Sveučilište u Zagrebu

Fakultet organizacije i informatike

Saša Mitrović

Metodološki okvir za izradu analitičkog informacijskog sustava temeljenog na metodama strojnog učenja u produkcijskom okružju

DOKTORSKI RAD

Varaždin, 2025.

Sveučilište u Zagrebu

Fakultet organizacije i informatike

SAŠA MITROVIĆ

Metodološki okvir za izradu analitičkog informacijskog sustava temeljenog na metodama strojnog učenja u produkcijskom okružju

DOKTORSKI RAD

Mentori:
prof. dr. sc. Neven Vrček
dr. sc. Valentina Janev

Varaždin, 2025.

University of Zagreb

Faculty of Organization and Informatics

Saša Mitrović

Methodological framework for developing analytical information system based on machine learning methods in a production environment

DOCTORAL THESIS

Supervisors:
Neven Vrček, PhD, full professor
Valentina Janev, PhD

Varaždin, 2025.

SAŽETAK

Rad predstavlja razvijeni automatizirani algoritam latencije temeljen na selekciji atributa skupa podataka koji skraćuje vrijeme predviđanja kod korištenja klasifikacijskih ili regresijskih metoda strojnog učenja u produkcijskom okružju. Glavni čimbenici razvijenog algoritma smanjenja latencije predviđanja su broj atributa u skupovima podataka, vrste atributa u skupovima podataka, postavke klasifikacijskih ili regresijskih metoda strojnog učenja, izvlačenje najznačajnijih atributa iz skupova podataka te broj predviđanja koji je moguće ostvariti u određenom vremenskom razdoblju – što je važna mjera u produkcijskom okružju. U radu, osim automatiziranog algoritma latencije, predstavljena je prilagođena verzija CRISP-DM metodologije za analitički informacijski sustav, odnosno za pretprocesiranje i obradu podataka s algoritmom latencije u analitičkom informacijskom sustavu. Rad odstupa od tradicionalnog vrednovanja modela strojnog učenja isključivo s mjerom postotka greške ili točnosti jer u velikom broju stvarnih situacija u produkcijskom okružju jedan posto razlike u točnosti predviđanja između različitih metoda strojnog učenja nema preveliki utjecaj ako se predviđanje može ostvariti u značajnije kraćem vremenu.

Ključne riječi: algoritam, automatizirano strojno učenje, automatizirani cjevovod, redukcija dimenzionalnosti, metodološki okvir

ABSTRACT

The paper will propose the development of an automated latency algorithm based on the selection of data set attributes that will shorten the prediction time when using classification/regression machine learning methods in a production environment. The main factors of the proposed prediction latency reduction algorithm are number of attributes in datasets, attribute types in datasets, settings of classification/regression machine learning methods, extraction of the most important attributes from datasets and number of predictions that can be achieved in a certain timespan - an important measure in the production environment. In addition to the automated latency algorithm, the paper will propose the development of a customized version of the CRISP-DM methodology for analytical information system, namely for pre-processing and data processing with the prediction latency algorithm in analytical information system. The paper seeks to move away from evaluating machine learning models solely with a measure of error or accuracy because in many real-world situations the one percent difference in prediction accuracy between different machine learning methods does not have much effect if the prediction can be achieved in a significantly shorter time.

Keywords: Algorithm, Automated machine learning, Automated pipeline, Dimensionality reduction, Methodological framework

PROŠIRENI SAŽETAK

Predloženi metodološki okvir za izgradnju analitičkog informacijskog sustava temeljenog na metodama strojnog učenja osmišljen i razvijen je kao automatizirani, skalabilni i domenski neovisan softverski sustav – naziva *MoziaisMl*, optimiziran za primjenu u produkcijskim okružjima s visokim zahtjevima za niskom latencijom i efikasnim izvođenjem modela. Primarni znanstveni doprinos očituje se u zamjeni tradicionalnog, ručnog procesa izgradnje modela strojnog učenja s algoritmom latencije koji istodobno integrira vlastito razvijene metode redukcije dimenzionalnosti – *Glasanja* i *Ruksaka* (engl. *Knapsack*).

Arhitektura sustava strukturirana je modularno s povezanim komponentama kako bi omogućila:

- integraciju heterogenih izvora podataka (relacijske baze podataka, CSV datoteke),
- automatiziranu i dinamičku pripremu podataka uključujući imputaciju nedostajućih vrijednosti, enkodiranje kategorijskih varijabli i kvantilnu transformaciju distribucija značajki,
- višekriterijsku selekciju značajki temeljenu na kombinaciji heurističkih (*Random Forest, XGBoost, Mutual Information, Recursive Feature Elimination*) i razvijenih metoda i optimizacijskih pristupa - glasanja i ruksaka, pri čemu se uzima u obzir važnost značajke i njezina varijanca kao funkcija troška,
- redukciju dimenzionalnosti kroz usporedbu standardnih tehnika - *PCA, t-SNE, UMAP,* Autoenkoderske mreže - s vlastito razvijenim pristupima optimizacije odabira značajki s ciljem minimizacije vremena izvođenja modela uz očuvanje prediktivne učinkovitosti,
- evaluaciju modela strojnog učenja primjenom *GridSearchCV* metode i *K-Fold* unakrsne validacije, uz dodatne metrike vrednovanja koje uključuju vrijeme izvođenja modela kao primarni kriterij, a točnost ili greška kao sekundarni, čime se uvodi koncept latentne evaluacije performansi u *AutoML* sustave, konkretno vlastito razvijenog sustava *MoziaisMl*.

Sustav *MoziaisMl* je razvijen u programskom jeziku Python, uz potporu znanstveno provjerenih biblioteka za strojno učenje – poglavito *scikit-learn* okružja, s ciljem primjene u stvarnim poslovnim scenarijima u kojima vremenska učinkovitost modela ima veću operativnu vrijednost od marginalnih poboljšanja u točnosti. Time se osigurava ne samo znanstveni doprinos u području automatiziranog modeliranja i optimizacije latentnih performansi, već i praktična iskoristivost u analitičkim informacijskim sustavima.

U svrhu znanstveno utemeljenog metodološkog pristupa razrade problema zamjene ručnog dizajna modela strojnog učenja, predloženi *MoziaisMl* sustav implementira cjelovit i modularan algoritam latencije, osmišljen prema načelima automatizacije, smanjenja vremenske složenosti, povećanja reproducibilnosti i evaluacije latentnih performansi. Sustav *MoziaisMl* je izgrađen s naglaskom na latencijsko optimizirane predikcije, gdje se vrijeme izvođenja modela vrednuje kao primarni kriterij, a točnost ili greška kao sekundarni, u skladu s potrebama stvarnih produkcijskih okružja u kojima operativna učinkovitost često nadilazi marginalne dobitke u prediktivnoj preciznosti.

U skladu s metodološkim pristupom zamjene ručno definiranih modela strojnog učenja automatiziranim pristupom, predloženi sustav *MoziaisMl* koncipiran je u skladu s fazama prilagođenog CRISP-DM metodološkog okvira, pri čemu se naglasak stavlja na automatizaciju postupaka pripreme podataka, reduciranje latencije izvođenja modela, te optimizaciju selekcije značajki u nižedimenzionalnom prostoru korištenjem vlastito razvijenih metoda.

Procesni tijek prvog dijela algoritma latencije unutar faze pripreme podataka obuhvaća sljedeće automatizirane korake:

- Detekcija i imputacija nedostajućih vrijednosti, pri čemu se dodatno kreiraju indikatorske značajke kako bi se informacijski potencijal praznih vrijednosti zadržao unutar modela.
- Razdvajanje složenih tekstualnih atributa na višestruke numerički interpretabilne stupce.
- Uklanjanje identifikacijskih varijabli temeljem unikatnosti vrijednosti po retku, čime se eliminira mogućnost prenaučenosti (engl. *overtraining*) modela.
- Usklađivanje strukture podataka između treniranog, validacijskog i testnog skupa eliminacijom međusobno nesinkroniziranih stupaca.
- Kategorizacija varijabli primjenom dinamičkog izbora između *One-Hot* i *Label* enkodiranja, ovisno o broju jedinstvenih vrijednosti i postavljenom pragovnom parametru.
- Kvantilna transformacija varijabli u uniformnu distribuciju radi ublažavanja utjecaja odstupanja i povećanja robusnosti modela.

Drugi dio algoritma latencije se odnosi na prilagođenu CRISP-DM fazu modeliranje i optimizacija s automatiziranim odabirom najboljeg modela strojnog učenja i njegovih

hiperparametara, dok u okviru trećeg dijela algoritma latencije, razvijene su i implementirane dvije vlastite metode redukcije dimenzionalnosti:

- Metoda Glasanja temelji se na agregaciji značajki koje su među najvažnijima prema četiri različite metode rangiranja (*Random Forest, XGBoost, Mutual Information, Recursive Feature Elimination*), pri čemu se odabire gornjih N% značajki s najviše glasova.
- Metoda Ruksaka (engl. *Knapsack algorithm*) predstavlja optimizacijski pristup koji uzima u obzir relativnu važnost značajki i njihovu varijancu kao oblik „troška“, a cilj je maksimizirati informacijski doprinos unutar unaprijed definiranog kapaciteta ruksaka veličine N (primjer, 10).

Sustav *MoziaisMl* podržava dvostruki modalitet primjene, odnosno klasifikacijske i regresijske zadatke, pri čemu se sve funkcionalnosti parametarski podešavaju (*task_type = „classification“* ili *„regression“*). Ovakav pristup omogućuje ne samo značajno ubrzanje faze dizajna modela i smanjenje vremena izvođenja u produkcijskim okružjima, već i zadržavanje ili poboljšanje prediktivne preciznosti zahvaljujući automatiziranim i replikabilnim postupcima unutar znanstveno validiranog metodološkog okvira.

Unutar faza modeliranje i optimizacija modela te evaluacija, u skladu s prilagođenim CRISP-DM metodološkim okvirom, razvijen je automatizirani postupak selekcije i vrednovanja modela koji nadilazi tradicionalni pristup temeljen isključivo na metričkoj točnosti, te integrira dimenziju vremenske učinkovitosti i latencijskog odziva nad izvornim skupom podataka te skupom podataka gdje su dvije vlastite metode redukcije dimenzionalnosti izvršile selekciju značajki odnosno smanjile dimenzionalnost skupa podataka. Cilj ovih faza je omogućiti automatizirano kreiranje, evaluaciju i selekciju modela strojnog učenja kako bi se potvrdile postavljene znanstvene hipoteze da brže vrijeme izvođenja modela u stvarnom produkcijskom okružju može imati veću operativnu vrijednost od marginalno bolje točnosti.

Postupak faza modeliranja i optimizacije te evaluacije uključuje sljedeće korake:

- Automatizirani odabir najboljeg modela i hiperparametara po točnosti i vremenu temeljem unaprijed definiranih parametarskih mreža i podržanih modela:
    - Podržani modeli su višestruki i raznoliki, čime se omogućuje metodološka fleksibilnost:
        - Za klasifikacijske zadatke su: *Decision Tree, Random Forest, SVC, XGBoost, GaussianNB, MLP, KNN, i Gradient Boosting*.

        - Za regresijske zadatke su: *Linear Regression, Lasso, Ridge, ElasticNet, SVR, Random Forest, XGBoost, MLP i Gradient Boosting.*
- Primjenu *GridSearchCV* metode, koja uz pomoć petostruke *K-Fold* unakrsne validacije osigurava robusnu procjenu generalizacijske sposobnosti modela.
- Sustavno bilježenje rezultata evaluacije najboljeg modela s hiperparametrima koje je proslijedila faza modeliranje i optimizacija modela, uključujući metapodatke o vremenu treniranja, vremenu predikcije te klasične metrike točnosti - *Accuracy* za klasifikaciju, srednje kvadratno odstupanje (*Root Mean Square Error*, RMSE) za regresiju.

Rezultati evaluacijskog procesa predloženog metodološkog okvira i ovog istraživanja, prošireni su integracijom latencijske ciljne funkcije, koja kvantificira kompromis između brzine izvršavanja modela (vremena predikcije) i njegove prediktivne točnosti. Ovakav pristup omogućuje primjenu višekriterijske optimizacije unutar metodološkog okvira automatiziranog strojnog učenja (*AutoML*) *MoziaisMl*, gdje se vlastito razvijene metode redukcije dimenzionalnosti i selekcije značajki, *Glasanja* i *Ruksaka*, nadovezuju na proces evaluacije kako bi se postigla optimalna kombinacija predikcijske preciznosti i izvedbene učinkovitosti u produkcijskim okružjima.

Rezultati istraživanja omogućuju višekriterijsko vrednovanje učinkovitosti razvijenog *AutoML* sustava *MoziaisMl* s algoritmom latencije. Ključne mjerne metrike korištene za kvantitativnu procjenu sustava *MoziaisMl* uključuju: točnost klasifikacije, RMSE za regresijske zadatke, vrijeme treniranja i predikcije kao mjera latencijske efikasnosti, broj odabranih značajki kao pokazatelj dimenzionalne složenosti modela, te reproduktivnost, koja je osigurana automatiziranim zapisom svih eksperimentalnih rezultata i konfiguracija.

Provedena je komparativna analiza između vlastito razvijenih metoda redukcije dimenzionalnosti – *Glasanja* i *Ruksaka* – te referentnih i prihvaćenih tehnika redukcije dimenzionalnosti: *PCA, t-SNE, UMAP* i Autoenkoderskih mreža. Usporedba se temeljila na sljedećim kriterijima: prediktivna točnost, broj odabranih značajki, vrijeme izvođenja modela te računalna složenost algoritma. Analizom rezultata utvrđeno je da vlastito razvijene metode postižu kompetitivne rezultate s povoljnim balansom između brzine izvođenja i očuvanja prediktivne preciznosti, u kontekstu primjene u različitim poslovnim domenama produkcijskog okružja.

Ovakav dizajn *MoziaisMl* sustava s algoritmom latencije predstavlja znanstveno validirano rješenje za smanjenje ljudske intervencije u razvoju modela strojnog učenja, istovremeno

osiguravajući visoku operativnu učinkovitost, skalabilnost i metodološku rigoroznost, čime se ostvaruje znanstveni doprinos u području automatiziranih analitičkih informacijskih sustava temeljenih na metodama strojnog učenja.

Zaključno, rezultati potvrđuju valjanost predloženog metodološkog okvira temeljenog na automatizaciji dizajna modela strojnog učenja, s naglaskom na optimizaciju latencije i redukciju dimenzionalnosti bez značajnog gubitka točnosti. Prednosti vlastitih metoda redukcije dimenzionalnosti, osobito u pogledu operativne skalabilnosti i fleksibilnosti, ukazuju na njihovu održivost i primjenjivost kako u znanstvenim istraživanjima tako i u stvarnim poslovnim produkcijskim okružjima.

EXTENDED ABSTRACT

The proposed methodological framework for developing an analytical information system based on machine learning techniques has been conceptualized and implemented as an automated, scalable, and domain independent software system, named *MoziaisMl*. It is specifically optimized for deployment in production environments characterized by stringent requirements for low latency and efficient model execution. The primary scientific contribution of this framework lies in replacing the traditional, manual model development process with a latency-oriented algorithm that simultaneously integrates two custom developed dimensionality reduction methods – the *Voting method* and the *Knapsack method*.

The system architecture is designed in self-contained modules with interchangeable components to support the following functionalities:

- integration of heterogeneous data sources (relational databases, CSV files),
- automated and dynamic data preprocessing, including imputation of missing values, categorical variable encoding, and quantile transformation of feature distributions,
- multi-criteria feature selection based on a combination of heuristic methods (*Random Forest, XGBoost, Mutual Information, Recursive Feature Elimination*) and custom developed optimization techniques — the voting-based consensus and the knapsack algorithm — where feature importance and variance are treated as cost functions,
- dimensionality reduction through the comparative analysis of standard techniques (PCA, t-SNE, UMAP, Autoencoders) and custom developed optimization-based feature selection methods, with the objective of minimizing model execution time while preserving predictive performance,
- evaluation of machine learning models with *GridSearchCV* and *K-Fold cross-validation*, extended with performance metrics that prioritize execution time as the primary evaluation criterion and accuracy or error as the secondary, thereby introducing the concept of latency-aware performance assessment in *AutoML* systems - specifically within custom developed *MoziaisMl* system.

The *MoziaisMl* system has been developed in the Python programming language, leveraging scientifically validated and externally referenced machine learning libraries - primarily the *scikit-learn* framework - with the objective of application in real-world business scenarios

where model execution efficiency holds greater operational value than marginal gains in predictive accuracy. This approach ensures not only a scientific contribution in the domain of automated modelling and latent performance optimization, but also practical applicability within analytical information systems.

For a scientifically grounded methodological approach to addressing the problem of replacing manual machine learning model design, the proposed *MoziaisMl* system implements a comprehensive and modular latency driven algorithm, designed in accordance with the principles of automation, reduction of temporal complexity, increased reproducibility, and latent performance evaluation. The *MoziaisMl* system is constructed with a focus on latency optimized predictions, where model execution time is assessed as the primary evaluation criterion and predictive accuracy or error as secondary, aligned with the requirements of real-world production environments in which operational efficiency often outweighs marginal improvements in predictive precision.

In accordance with the methodological approach aimed at replacing manually defined machine learning models with an automated strategy, the proposed *MoziaisMl* system is designed in alignment with the phases of an adapted CRISP-DM methodological framework. The system places particular emphasis on the automation of data preparation procedures, the reduction of model execution latency, and the optimization of feature selection within a lower-dimensional space through the application of custom developed methods.

The procedural flow of the first part of the latency algorithm within the data preparation phase comprises the following automated steps:

- Detection and imputation of missing values, whereby additional indicator features are generated to retain the informational value of null and empty entries within the modelling process.
- Decomposition of complex textual attributes into multiple numerically interpretable columns.
- Elimination of identifier variables based on their uniqueness per row, thereby reducing the risk of model overfitting.
- Structural alignment of training, validation and test datasets, achieved by removing mutually unshared columns to ensure schema consistency.

- Categorical encoding of variables, employing a dynamic selection between One-Hot and Label Encoding, based on the number of distinct values and a defined threshold parameter.
- Quantile transformation of variables into a uniform distribution, with the objective of mitigating the influence of outliers and enhancing model robustness.

The second part of the latency algorithm pertains to the customized CRISP-DM phase of modelling and optimization, which entails the automated selection of the most appropriate machine learning model along with its optimal hyperparameters. Within the third part of the latency algorithm, two proprietary dimensionality reduction methods have been developed and implemented:

- *The Voting method* relies on the aggregation of features ranked as most important according to four distinct machine learning methods (*Random Forest, XGBoost, Mutual Information, and Recursive Feature Elimination*). The top N% of features receiving the highest number of votes are selected for further modelling.
- *The Knapsack method* constitutes an optimization-based approach in which each feature's relative importance is treated as a "value" and its variance as a form of "cost". The objective is to maximize the total informational gain of the selected feature subset while respecting a predefined complexity budget (e.g., a knapsack capacity of N = 10).

The *MoziaisMl* system supports a dual mode application, encompassing both classification and regression tasks, wherein all functionalities are parametrically adjustable (i.e., *task_type = "classification" or "regression"*). This approach facilitates not only a significant acceleration of the model design phase and a reduction in execution time within production environments, but also the retention or improvement of predictive accuracy, owing to automated and replicable procedures embedded within a scientifically validated methodological framework.

Within the Modelling and Optimization and Evaluation phases — aligned with the adapted CRISP-DM methodological framework — an automated procedure for model selection and performance assessment has been developed, extending beyond the conventional accuracy-based evaluation paradigm. This framework integrates the dimension of execution time efficiency and latency responsiveness, applied to both the original dataset and to datasets transformed through two custom developed feature reduction methods, *Voting method* and the *Knapsack method*. The objective of these phases is to enable automated construction,

evaluation, and selection of machine learning models in accordance with the goal of confirming formulated scientific hypothesis: that reduced model execution time in real-world production environments may possess greater operational value than marginal improvements in predictive accuracy.

The Modelling and Optimization, and evaluation phases comprise the following sequential steps:

- Automated selection of the optimal model and its hyperparameters, based on predefined parameter grids and evaluation criteria that incorporate both predictive accuracy and execution time.
    - The framework supports a diverse set of models, providing methodological flexibility:
        - For classification tasks, supported models include: *Decision Tree, Random Forest, Support Vector Classifier (SVC), XGBoost, Gaussian Naive Bayes (GaussianNB), Multi-Layer Perceptron (MLP), K-Nearest Neighbors (KNN), and Gradient Boosting.*
        - For regression tasks, supported models include: *Linear Regression, Lasso, Ridge, ElasticNet, Support Vector Regressor (SVR), Random Forest, XGBoost, MLP, and Gradient Boosting.*
- Application of the *GridSearchCV* procedure combined with five-fold cross-validation (*K-Fold*), ensuring a robust estimation of each model's generalization capability.
- Systematic logging of evaluation results for the selected model and corresponding hyperparameters, as determined in the modelling and optimization phase. Logged metadata include: training time, prediction latency, and standard accuracy metrics - Accuracy for classification and Root Mean Square Error (RMSE) for regression.

The evaluation outcomes of the proposed methodological framework and this research have been extended through the integration of a latency-based objective function, which quantifies the trade-off between model execution speed (prediction time) and predictive accuracy. This approach facilitates the application of multi-objective optimization within the *MoziaisMl* automated machine learning (*AutoML*) framework, wherein the custom developed dimensionality reduction and feature selection methods - the *Voting method* and the *Knapsack method* - are incorporated into the evaluation process. The goal is to achieve an optimal balance

between predictive performance and operational efficiency, particularly within production-grade environments characterized by stringent latency constraints.

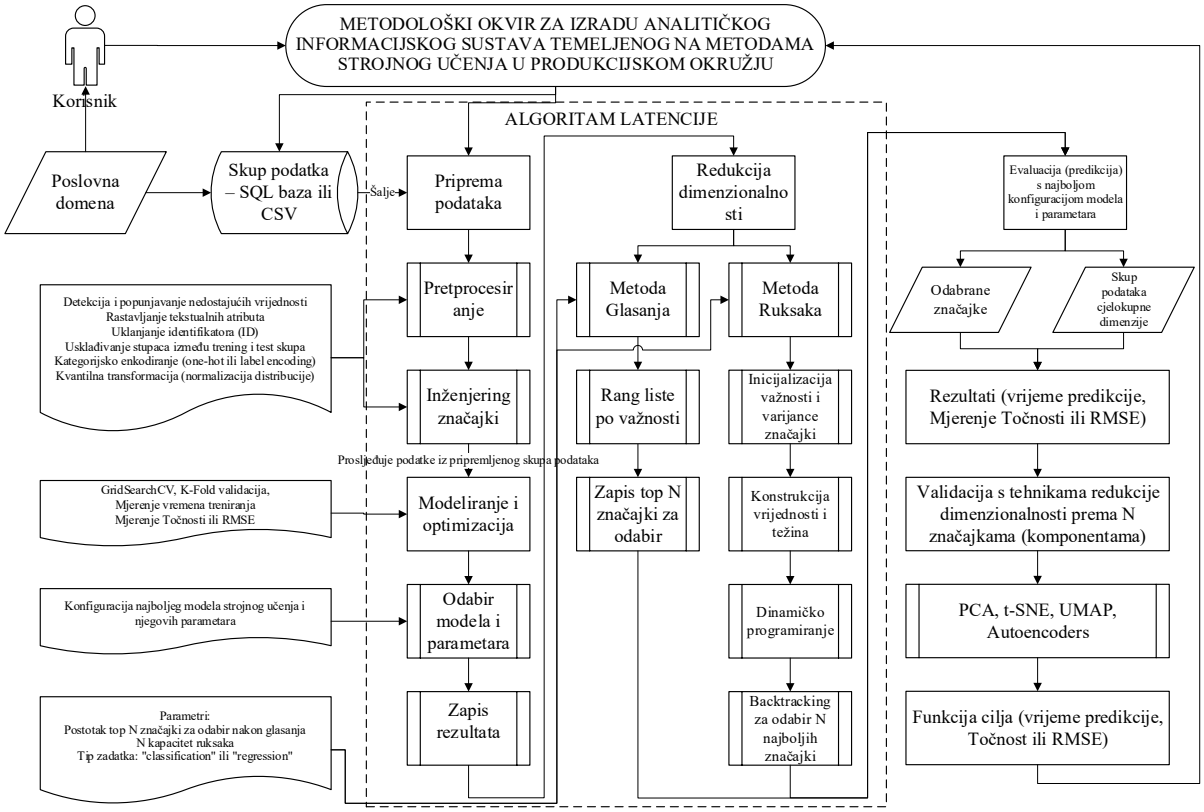
The research outcomes enable a multi-criteria evaluation of the performance of the developed *AutoML* system *MoziaisMl*, which incorporates a latency-optimized algorithm. The key performance metrics employed for the quantitative assessment of the *MoziaisMl* system include: classification accuracy, RMSE for regression tasks, training and prediction time as indicators of latency efficiency, the number of selected features as a measure of the model's dimensional complexity, and reproducibility, ensured through the automated logging of all experimental results and configuration parameters.

A comparative analysis was conducted between the proprietary dimensionality reduction methods - the *Voting method* and the *Knapsack method* - and widely accepted reference techniques for dimensionality reduction, including *PCA, t-SNE, UMAP*, and *Autoencoder* networks. The comparison was based on the following evaluation criteria: predictive accuracy, number of selected features, model execution time, and the computational complexity of the algorithm. The analysis of results indicates that the custom developed methods achieve competitive performance, demonstrating a favourable balance between execution speed and the preservation of predictive precision, particularly in the context of deployment within diverse business domains in production environments.

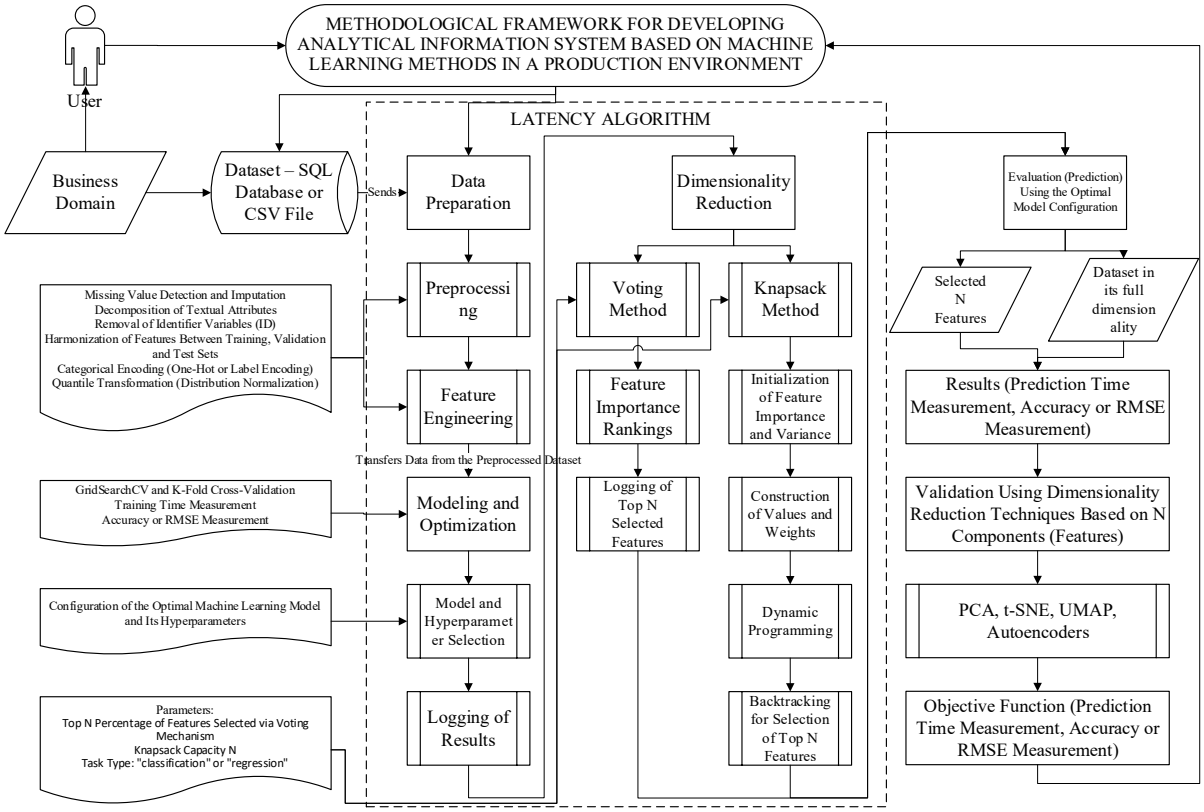
This design of the *MoziaisMl* system, incorporating a latency-based algorithm, represents a scientifically validated solution for reducing human intervention in the development of machine learning models, while simultaneously ensuring high operational efficiency, scalability, and methodological rigor. As such, it constitutes a substantive scientific contribution to the field of automated analytical information systems grounded in machine learning methodologies.

In conclusion, the results confirm the validity of the proposed methodological framework based on the automation of machine learning model design, with a specific focus on latency optimization and dimensionality reduction without significant loss of predictive accuracy. The advantages of the custom developed dimensionality reduction methods - particularly in terms of operational scalability and flexibility - demonstrate their sustainability and applicability both in scientific research and in real-world production environments.

GRAFIČKI SAŽETAK



GRAPHICAL ABSTRACT



SADRŽAJ

POPIS TABLICA

POPIS SLIKA

POPIS POJMOVA

MoziaisMl – (**M**)Metodološki (**o**)okvir (**z**)za (**i**)izradu (**a**)analitičkog (**i**)informacijskog (**s**)sustava temeljenog na metodama (**M**)strojnog (**l**)učenja. **M** i **l** dolaze od prijevoda riječi strojno i učenje u *machine* i *learning.*

CRISP-DM - CRoss Industry Standard Process for Data Mining

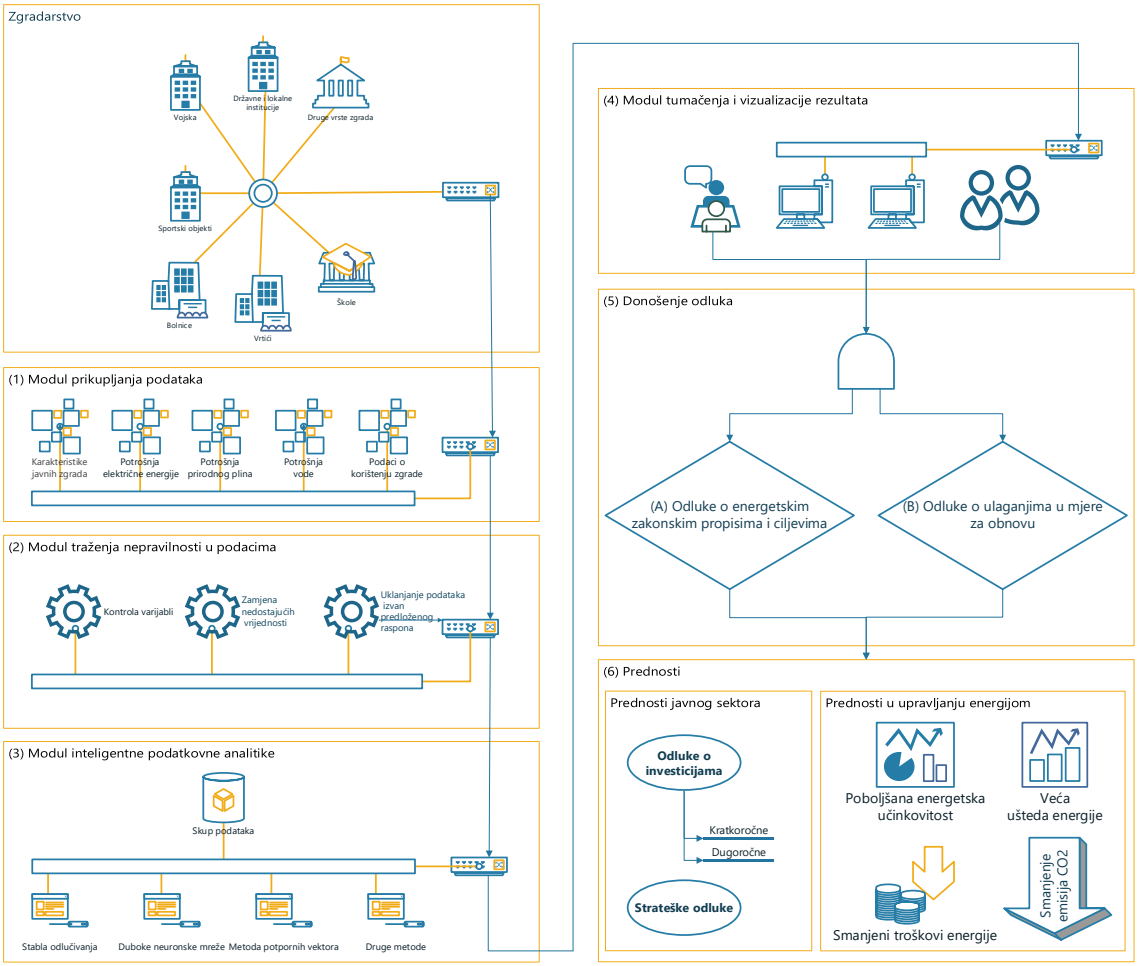
AutoML - Automated machine learning

1. UVOD

Uvodno poglavlje doktorskog rada predstavlja istraživački problem latencije i optimizacije vremena odnosno postizanja više brzine u analitičkim procesima, objašnjava potrebu za unaprjeđenjem metodološkog pristupa koji omogućuje bržu predikciju, smanjenje vremena izvođenja odnosno postizanja više brzine te optimizaciju modela strojnog učenja u produkcijskom okružju. Zatim, postavlja hipoteze koje su usmjerene na evaluaciju učinkovitosti predložene, prilagođene metodologije s ciljem smanjenja vremena odnosno postizanja više brzine kod predikcije nad podatcima. Poglavlje, također, ističe važnost istraživanja i znanstveni doprinos u kontekstu smanjenja dimenzionalnosti podataka i vremenske učinkovitosti, dok se na kraju poglavlja, predstavlja kratak pregled organizacije i sadržaja narednih poglavlja.

1.1. Problem istraživanja

Sve veća količina dostupnih podataka predstavlja značajan izazov u razvoju strojnog učenja, obzirom da tradicionalni pristupi razvoju modela strojnog učenja postaju sve složeniji i zahtijevaju značajne resurse u pogledu vremena, računalne snage i domenske ekspertize. Autori Zekić-Sušac, Mitrović i Has predložili su arhitekturu inteligentnog informacijskog sustava za energetsko upravljanje zgradama u javnom sektoru – MERIDA – metodološki temeljno na znanstvenim dostignućima istraživanja iz područja strojnog učenja, uz proširenje na integraciju principa poslovne inteligencije, s ciljem izrade predefiniranih ručno kreiranih modela strojnog učenja za prediktivnu analitiku. Arhitektura MERIDA sustava koncipirana je kao višeslojni model koji omogućuje analizu podataka od izvora do strateške odluke, uz podršku modela strojnog učenja. Arhitektura MERIDA sustava je prikazana na slici u nastavku[1].



Slika 1: Arhitektura MERIDA sustava (Izvor: [1])

Za modul inteligentne podatkovne analitike (3), svrha sljedećeg rada [2], autora Mitrović i Zekić-Sušac je bila pružiti metodološki utemeljenu i znanstveno rigoroznu analizu učinkovitosti različitih algoritama strojnog učenja koji se mogu primjenjivati u kontekstu modeliranja i optimizacije modela za energetsku učinkovitost zgrada unutar inteligentnog informacijskog sustava. Rad autora Mitrović i Zekić-Sušac sintetizira spoznaje iz znanstvene i stručne literature, pružajući pregled postojećih istraživačkih dostignuća (engl. *state of the art*), kao i smjernice za buduću primjenu naprednih algoritamskih metoda. Kroz sustavnu evaluaciju prediktivne uspješnosti, prednosti i ograničenja pojedinih algoritama – uključujući metode nadziranog i nenadziranog učenja – istraživanje se usredotočuje na kriterije njihove primjenjivosti u domeni energetske analitike. Kroz kritički osvrt autora Mitrović i Zekić-Sušac na izbor algoritama, rad doprinosi razvoju adaptivne i automatizirane komponente Modeliranje i optimizacija modela, sustava ovog doktorskog rada. Dobiveni rezultati i preporuke rada autora

Mitrović i Zekić-Sušac bili su korisni za razvoj algoritamskih rješenja i sustava u ovom doktorskom radu.[2] Osim MERIDA arhitekture, autori Tonković, Mitrović i Zekić-Sušac predstavili su prijedlog arhitekture koja se metodološki temelji na integraciji prediktivnih modela strojnog učenja u postojeći informacijski sustav zgradarstva javnog sektora putem reprezentacijskog prijenosa stanja (engl. *REpresentational State Transfer Application Programming Interface*, REST API). Cilj takve integracije jest omogućiti generiranje prognoznih modela i identifikaciju ključnih prediktora koji utječu na potrošnju prirodnog plina, čime se upraviteljima u javnom sektoru zgradarstva omogućuje donošenje podatkovno utemeljenih odluka o alokaciji resursa za rekonstrukciju onih zgrada koje imaju najveći potencijal za smanjenje potrošnje energenata. Predloženi pristup implementacije poslovno-inteligentnog sustava za energetsko upravljanje sastoji se od dvije glavne faze: (1) izgradnja web servisa (REST API-a) nad razvijenim modelima strojnog učenja te (2) ugradnja tog web servisa kao funkcionalnog modula unutar postojeće web-aplikacije informacijskog sustava. Tehnički zahtjevi za realizaciju navedene arhitekture sustava nisu visoki. Budući da suvremeni softverski okviri za strojno učenje omogućuju izvoz postojećih modela strojnog učenja u obliku REST API-a – uključujući i modele autora Mitrović, Zekić-Sušac, Has i Tonković (primjer, stabala odlučivanja iz radova autora [3, 4]). Ovakav REST API omogućuje smještanje (engl. *hosting*) modela na poslužitelj (engl. *server infrastructure*), pri čemu se samo rezultati predikcije distribuiraju prema korisnicima, čime se optimizira opterećenje mreže i povećava skalabilnost rješenja.[4]

Predložene arhitekture, s posebnim naglaskom na arhitekturu inteligentnog informacijskog sustava MERIDA[1], predstavljaju konceptualni, metodološki i funkcionalni okvir za transformaciju postojećeg ručnog, vremenski intenzivnog i ekspertno ovisnog procesa modeliranja strojnog učenja u sustav temeljen na principima automatiziranog strojnog učenja. Ovakav pristup omogućuje značajno smanjenje potreba za stručnim znanjima iz domene strojnog učenja te omogućuje korisnicima brzu, skalabilnu i ponovljivo točnu primjenu prediktivnih modela u produkcijskim operativnim okružjima. Automatizacijom cijelog procesa – od pripreme podataka, odabira i optimizacije modela, redukcije dimenzionalnosti značajki do evaluacije performansi – predstavlja se, u okviru ovog doktorskog rada razvijeni sustav *MoziaisMl*, koji je razvijen na temelju stečenih znanja kroz četverogodišnje sudjelovanje na projektu *Metodološki*

okvir za učinkovito upravljanje energijom s pomoću inteligentne podatkovne analitike (šifra: HRZZ-IP-2016-06-8350) voditeljice prof. dr. sc. Marijane Zekić-Sušac te omogućuje ne samo poboljšanje energetske učinkovitosti u javnom sektoru zgradarstva, već i širu primjenu u raznim poslovnim domenama koje zahtijevaju visoku razinu prediktivne pouzdanosti i minimalnu latenciju u donošenju odluka. Time je, ovim doktorskim radom, razvijen inteligentan, adaptivan i domenski neovisan sustav koji ubrzava donošenje strateških i operativnih odluka. Pritom *MoziaisMl* sustav jasno prikazuje, interpretira i dokumentira proces automatiziranog dizajna cjevovoda i evaluacije modela strojnog učenja te mjerljivo doprinosi vremenskoj učinkovitosti i konkurentnosti poslovnih procesa u stvarnim (produkcijskim) okružjima javnog i privatnog sektora. Smanjenjem latencije izmjeren je, kroz funkciju cilja, doprinos konkurentnosti poslovnih procesa u produkcijskim okružjima.

Radovi autora Mitrović, Zekić-Sušac, Has i Tonković [1-4] dovode do zaključka autora ovog doktorskog rada kako tradicionalni pristupi ručnog kreiranja modela strojnog učenja često rezultiraju financijski zahtjevnim procesima koji predstavljaju značajnu prepreku u široj primjeni strojnog učenja u različitim domenama. Kako bi se unaprijedila učinkovitost te pristup razvoju modela strojnog učenja bio jednostavniji i dostupniji većoj populaciji, novija istraživanja (više u [5-9]) usmjeravaju se na razvoj sustava utemeljenih na automatizaciji cijelog procesa razvoja modela strojnog učenja. Navedeni sustavi kroz modularnu arhitekturu omogućuju automatizirani tijek rada - od pripreme podataka i inženjerstva značajki do odabira modela, optimizacije hiperparametara i, na kraju, evaluacije predikcije nad podatcima. Navedeni sustavi bi u budućnosti trebali smanjiti potrebu za visokim stupnjem znanja i stručnosti odnosno ekspertizi domenskih stručnjaka te biti lako prilagodljivi za implementaciju u različitim domenama. [5-9] Nastavno, na prethodna novija istraživanja[5-9] autori Mitrović i Vrček predstavili su metodološki okvir koji se temelji na primjeni automatiziranih metoda strojnog učenja za predikciju emisije ugljikova dioksida (CO_2) u zgradama Republike Hrvatske. Predstavljeni metodološki okvir automatiziranih metoda strojnog učenja autora Mitrović i Vrček nije imao komponentu redukcije dimenzionalnosti skupa podataka. Zbog toga je vrijeme izvršavanja predikcije trajalo značajno duže, što je negativno utjecalo na vremensku učinkovitost. Također, predstavljeni okvir nije predstavljao sustav s cjelokupnim automatiziranim tijekom rada kakav je razvijen u sklopu ovog doktorskog rada.[10] Osim

primjene automatiziranih metoda strojnog učenja, autori Mitrović i Vrček proširuju domenu automatizacije s razvojem rješenja za evaluaciju klasifikacijskih zadataka s neuravnoteženim skupovima podataka primjenom prilagođenog CRISP-DM (engl. *CRoss Industry Standard Process for Data Mining*)[11] metodološkog okvira čime su željeli istražiti rizike od pristranosti modela uslijed disbalansa klasa.[12] Istraživanje rizika predstavljalo je bitnu evaluaciju za razvoj automatiziranog sustava strojnog učenja ovog rada jer je doprinijelo razumijevanju i razvoju automatizacije cijelog procesa *MoziaisMl*-a s unaprijeđenim generalizacijskim sposobnostima u uvjetima stvarnih, neuravnoteženih distribucija podataka različitih poslovnih domena u produkcijskim okružjima.

Nadalje, u području automatiziranih sustava strojnog učenja, pregledni rad prošlosti sadašnjosti i budućnosti automatiziranog strojnog učenja, autora Baratchi, et al. [5] ističe, među ostalim, izazove u primjeni sustava automatiziranog strojnog učenja na procesiranje skupova podataka s velikim brojem značajki, što dovodi do rasta prostora pretrage, značajnih računalnih zahtjeva i financijskih troškova. Sustavi automatiziranog strojnog učenja koriste metode optimizacije hiperparametara u velikim prostorima pretrage koji uključuju odabir algoritama, njihovu konfiguraciju te elemente pretprocesiranja. Međutim, nisu sve optimizacije hiperparametara i pristupi skalabilni na zahtjevne konfiguracijske procesne pristupe optimizaciji. Prethodno navedeno, postaje još izraženije u scenarijima kad se koriste tehnike validacije, poput *K-Fold* unakrsne validacije, koje dodatno povećavaju računalne zahtjeve i vrijeme izvršavanja. Obzirom kako sustavi automatiziranog strojnog učenja koriste heurističke pretrage i evolucijske algoritme, njihova efikasnost se značajno smanjuje kada raste dimenzionalnost podataka, zbog čega, kako autori Baratchi, et al. [5] ističu, su nužni inovativni pristupi u optimizaciji prostora pretrage kako bi se smanjili računalni troškovi i poboljšala izvedba modela strojnog učenja. [5]

Potaknuto navedenim problemima svih prethodno navedenih autora [1-10], rad predstavlja razvijeni automatizirani algoritam latencije temeljen na selekciji atributa skupa podataka koji skraćuje vrijeme predviđanja kod korištenja klasifikacijskih ili regresijskih metoda strojnog učenja u produkcijskom okružju. Pojam latencija, koji se u telekomunikacijama odnosi na kašnjenje u prijenosu informacija [13], ovdje se prenosi na domenu strojnog učenja, gdje označava vrijeme potrebno za evaluaciju modela

strojnog učenja. Produkcijsko okružje, u kontekstu primjene vlastito razvijenih metoda redukcije dimenzionalnosti, predstavlja operativni okvir u kojem se ispituje robusnost i generalizacijska sposobnost vlastito razvijenih metoda redukcije dimenzionalnosti na stvarnim podatkovnim skupovima različitih domena, pri čemu se provodi usporedna analiza s popularnim, tehnikama redukcije dimenzionalnosti - analizom glavnih komponenti (engl. *Principal Component Analysis*, PCA)[14-16], t-distribuirano stohastičko ugrađivanje susjeda (engl. *t-distributed Stochastic Neighbor Embedding,* t-SNE) [14-17], UMAP (engl. *Uniform Manifold Approximation and Projection*)[18] i autoenkoderskim neuronskim mrežama (engl. *Autoencoders*)[19]. Takav evaluacijski pristup omogućuje objektivnu validaciju učinkovitosti vlastito razvijenih metoda redukcije dimenzionalnosti u uvjetima stvarne (produkcijske) primjene, uz naglasak na njihovu brzinu vremena izvršavanja, prilagodljivost, skalabilnost i održavanje visoke prediktivne točnosti. Autori Mitrović i Vrček su razvili i predstavili prilagođenu CRISP-DM metodologiju, u kojoj je osigurana automatizira i sustavna analiza linearnih i nelinearnih metoda redukcije dimenzionalnosti kroz sve faze CRISP-DM[11] procesa. Autori Mitrović i Vrček su automatizaciju razvili na način da se dinamički određuje broj komponenti sukladno dimenzionalnoj strukturi različitih skupova podataka, zatim se integriraju metode redukcije dimenzionalnosti i klasifikacijske metode strojnog učenja. Rezultati istraživanja autora Mitrović i Vrček ukazuju da su metode PCA[14-16] i autoenkoderske neuronske mreže[19] učinkovite u linearnom odnosno nelinearnom prostoru, pri čemu metoda PCA[14-16] postiže visoku klasifikacijsku točnost uz znatno manje vrijeme izvedbe, dok autoenkoderske neuronske mreže[19] ukazuju, u prosjeku, na veću točnost u nelinearnim domenama uz kompromis u vremenskoj učinkovitosti.[20] Dodatno, metode t-SNE[14-17] i UMAP[18] uključene su u evaluacijski pristup ovog rada kao referentni vizualizacijski alati zbog svoje sposobnosti očuvanja lokalne i globalne strukture podataka pri malom broju dimenzionalnih komponenti, s ciljem da se u ovom istraživanju vrednuju kao metode redukcije dimenzionalnosti neovisno o broju komponenti, izvan njihove uobičajene primjene u vizualizaciji struktura podataka. [14-18] Također, rad se odmiče od vrednovanja modela strojnog učenja isključivo s mjerom postotka greške ili točnosti jer u velikom broju stvarnih situacija u produkcijskom okružju jedan posto razlike u točnosti ili greški predviđanja između različitih metoda strojnog učenja nema preveliki utjecaj ako se predviđanje može ostvariti u značajnije kraćem

vremenu, stoga je prvi po rangu, kriterij vrednovanja modela strojnog učenja, vrijeme inferencije odnosno brzina, a tek drugi točnost odnosno greška predikcije.

1.2. Cilj i svrha istraživanja

Cilj i svrha provedbe istraživanja bila je razvoj automatiziranog algoritma latencije za produkcijsku primjenu unutar prilagođene verzije CRISP-DM [11, 21, 22] metodološkog okvira razvoja informacijskog sustava temeljenog na metodama strojnog učenja, a što je rezultiralo vlastito razvijenim sustavom automatiziranog strojnog učenja, naziva *MoziaisMl*. Algoritam latencije primjenom razvijenih metoda i nizom slijednih koraka analizira i reducira dimenzionalnost ulaznog skupa podataka kako bi se dobio skup podataka za treniranje, validaciju i testiranje temeljem kojeg metode strojnog učenja rade brže uz neznatne gubitke u točnosti ili greški predviđanja. Učinak algoritma latencije prati se kroz 5. fazu prilagođene CRISP-DM metodologije (Evaluacija). Algoritam latencije upravlja izborom značajki za predviđanje metoda strojnog učenja u kraćem vremenskom okviru u odnosu na sustave automatiziranog strojnog učenja (engl. *automated machine learning, AutoML*) i na platforme za upravljanje operacionalizacijom modela strojnog učenja (engl. *Machine Learning Model Operationalization Management*, *MLOps*) prikazane tablicom u nastavku.

Sustav	Vrsta	Link
AdaNet	automatski odabir neuronske arhitekture (NAS)	Github
Advisor	optimizacija hiperparametara (HPO)	Github
AMLA	optimizacija hiperparametara (HPO), automatski odabir neuronske arhitekture (NAS)	Github
ATM	optimizacija hiperparametara (HPO)	Github
Auger	optimizacija hiperparametara (HPO)	Početna stranica
auptimizer	optimizacija hiperparametara (HPO), automatski odabir neuronske arhitekture (NAS)	Github
Auto-Keras	automatski odabir neuronske arhitekture (NAS)	Github
AutoML Vision	automatski odabir neuronske arhitekture (NAS)	Početna stranica

AutoML Video Intelligence	automatski odabir neuronske arhitekture (NAS)	Početna stranica
AutoML Natural Language	automatski odabir neuronske arhitekture (NAS)	Početna stranica
AutoML Translation	automatski odabir neuronske arhitekture (NAS)	Početna stranica
AutoML Tables	automatsko inženjerstvo značajki (AutoFE), optimizacija hiperparametara (HPO)	Početna stranica
AutoPyTorch	optimizacija hiperparametara (HPO), automatski odabir neuronske arhitekture (NAS)	Github
HyperGBM	optimizacija hiperparametara (HPO)	Github
HyperKeras	automatski odabir neuronske arhitekture (NAS)	Github
Hypernets	optimizacija hiperparametara (HPO), automatski odabir neuronske arhitekture (NAS)	Github
auto-sklearn	optimizacija hiperparametara (HPO)	Github
auto_ml	optimizacija hiperparametara (HPO)	Github
BayesianOptimization	optimizacija hiperparametara (HPO)	Github
BayesOpt	optimizacija hiperparametara (HPO)	Github
comet	optimizacija hiperparametara (HPO)	Početna stranica
DataRobot	optimizacija hiperparametara (HPO)	Početna stranica
DEvol	automatski odabir neuronske arhitekture (NAS)	Github
DeepArchitect	automatski odabir neuronske arhitekture (NAS)	Github
Determined	optimizacija hiperparametara (HPO), automatski odabir neuronske arhitekture (NAS)	Github
Driverless AI	automatsko inženjerstvo značajki (AutoFE)	Početna stranica
FAR-HO	optimizacija hiperparametara (HPO)	Github
H2O AutoML	optimizacija hiperparametara (HPO)	Github
HpBandSter	optimizacija hiperparametara (HPO)	Github
HyperBand	optimizacija hiperparametara (HPO)	Github
Hyperopt	optimizacija hiperparametara (HPO)	Github
Hyperopt-sklearn	optimizacija hiperparametara (HPO)	Github
Hyperparameter Hunter	optimizacija hiperparametara (HPO)	Github
Katib	optimizacija hiperparametara (HPO)	Github
MateLabs	optimizacija hiperparametara (HPO)	Github
Milano	optimizacija hiperparametara (HPO)	Github

MLJAR	automatsko inženjerstvo značajki (AutoFE), optimizacija hiperparametara (HPO), automatski odabir neuronske arhitekture (NAS)	Github
mlr3automl	optimizacija hiperparametara (HPO)	GitHub
nasbot	automatski odabir neuronske arhitekture (NAS)	Github
neptune	optimizacija hiperparametara (HPO)	Početna stranica
NNI	optimizacija hiperparametara (HPO), automatski odabir neuronske arhitekture (NAS)	Github
Oboe	optimizacija hiperparametara (HPO)	Github
Optunity	optimizacija hiperparametara (HPO)	Github
R2.ai	optimizacija hiperparametara (HPO)	Početna stranica
RBFOpt	optimizacija hiperparametara (HPO)	Github
RoBO	optimizacija hiperparametara (HPO)	Github
Scikit-Optimize	optimizacija hiperparametara (HPO)	Github
SigOpt	optimizacija hiperparametara (HPO)	Početna stranica
SMAC3	optimizacija hiperparametara (HPO)	Github
TPOT	automatsko inženjerstvo značajki (AutoFE), optimizacija hiperparametara (HPO)	Github
TransmogrifAI	optimizacija hiperparametara (HPO)	Github
Tune	optimizacija hiperparametara (HPO)	Github
Xcessiv	optimizacija hiperparametara (HPO)	Github
SmartML	optimizacija hiperparametara (HPO)	Github
MLBox	automatsko inženjerstvo značajki (AutoFE), optimizacija hiperparametara (HPO)	Github
AutoAI Watson	automatsko inženjerstvo značajki (AutoFE), optimizacija hiperparametara (HPO)	Početna stranica
AUtoML	automatizirano strojno učenje (AutoML)	Github
Optuna	optimizacija hiperparametara (HPO)	Github

Tablica 1: Razvijeni AutoML sustavi (Izvor:[23])

Prethodna tablica predstavlja pregled različitih *AutoML* sustava, pri čemu su klasificirani prema vrsti funkcionalnosti koje implementiraju – kao što su automatski odabir neuronskih arhitektura (NAS), optimizacija hiperparametara (HPO) i automatizirano inženjerstvo značajki (AutoFE). Proučavajući prikazane *AutoML* sustave, metodološki, većina navedenih sustava usmjerena je isključivo na maksimizaciju prediktivne točnosti modela, što se očituje kroz mjerne metrike poput postotka točnih

klasifikacija ili pogreške regresije. Međutim, u produkcijskim okružjima razlika od jednog postotnog boda u točnosti modela često je manje značajna u usporedbi s razlikom u vremenu potrebnom za dobivanje predikcija. U predmetnom istraživanju i ovom doktorskom radu smatra se kako bi znanstveni pristup vrednovanju *AutoML* sustava trebao obuhvatiti višekriterijsku analizu, u kojoj se vrijeme izvršavanja modela (latencija) postavlja kao primarni kriterij optimizacije, dok se prediktivna točnost vrednuje kao sekundarni, ali još uvijek važan kriterij. Ovakva promjena paradigme u vrednovanju omogućila je razvoj skalabilnog, adaptivnog i vremenski efikasnog *AutoML* sustava, naziva *MoziaisMl*, koji je prikladan za dinamične poslovne sustave i stvarna vremenska ograničenja. Time se omogućila implementacija modela koji ne samo da su precizni, nego i vremenski učinkoviti u kontekstu operativnog odlučivanja. Iako postoje značajni *AutoML* sustavi, kao što je vidljivo iz prethodne tablice, primjetno je da većina navedenih sustava ne uključuje komponentu vrednovanja vremenske složenosti predikcije modela, čime se propušta važan aspekt optimizacije za produkcijske uvjete. Cilj i svrha predmetnog istraživanja i ovog doktorskog rada je ostvarenje znanstvenog pomaka prema metodološkom dizajnu *AutoML* sustava u kojem je vrijeme izvršavanja eksplicitno integrirano u procese predikcijske preciznosti kao primarni kriterij optimizacije, dok se prediktivna preciznost vrednuje kao sekundarni kriterij.

U produkcijskom okružju kada predviđanje traje predugo, odmah se troškovi uvećavaju i duže vrijeme uskraćuje resurse koji bi se utrošili na rješavanje drugih problema u sačuvanom vremenu te tako nastaje oportunitetni trošak. Sastavljanje funkcije cilja predstavlja važan zadatak u razvoju algoritma latencije. Najvažniji, u jednoj ciljnoj funkciji – prvi parametar po rangu – je vrijeme izvršavanja i drugi parametar – utjecaj primjene algoritma latencije na točnost ili grešku metrika predviđanja. Moguće je bilo i uvođenje trećeg parametra – poslovnog učinka. Poslovni učinak se ne analizira u doktorskom radu jer poslovni učinak zavisi od domene svakog skupa podataka pa se zbog toga ne analizira za svaki skup podataka posebno. Funkcija cilja omogućuje kvantitativnu usporedbu alternativnih rješenja vlastito razvijenih metoda za redukciju dimenzionalnosti i najčešće korištenih i popularnih tehnika redukcije dimenzionalnosti, kroz modele strojnog učenja, po svakom skupu podataka. Funkcija cilja optimizira vrijeme izvršavanja (najvažniji, prvi, prioritet po rangu) i minimizira negativni utjecaj latencije algoritma na

točnost predviđanja. Funkcija ima višekriterijsku optimizaciju gdje se balansira između brzine i preciznosti odnosno točnosti predviđanja. Parametri su:

- T – vrijeme izvršavanja algoritma (manje je bolje)
- I – utjecaj latencije algoritma na točnost predviđanja (manje je bolje)
- α – težinski koeficijent koji određuje balans između brzine i preciznosti, gdje $0 \leq \alpha \leq 1$
- T_{min}, T_{max} – minimalno i maksimalno vrijeme izvršavanja u skupu podataka
- I_{min}, I_{max} – minimalni i maksimalni utjecaj latencije na točnost

Vrijeme izvršavanja (T) predstavlja prvi, po rangu, kriterij – cilj je minimizirati ga. Drugi kriterij je utjecaj latencije na točnost (I) – cilj je minimizirati negativni učinak latencije na točnost. Nastavno, parametar α (engl. *alpha*) omogućava balansiranje između vremena izvršavanja (engl. *execution time*) i preciznosti (engl. *accuracy impact*):

- Ako je α blizu 1 – veći je naglasak na vrijeme izvršavanja odnosno brzinu
- Ako je α blizu 0 – veći je naglasak na točnost ili grešku odnosno preciznost

Nadalje, kako bi parametri vremena izvršavanja (T) i utjecaja latencije na točnost (I) bili usporedivi potrebno je provesti normalizaciju kako bi vrijednosti parametara bile skalirane u rasponu (0, 1):

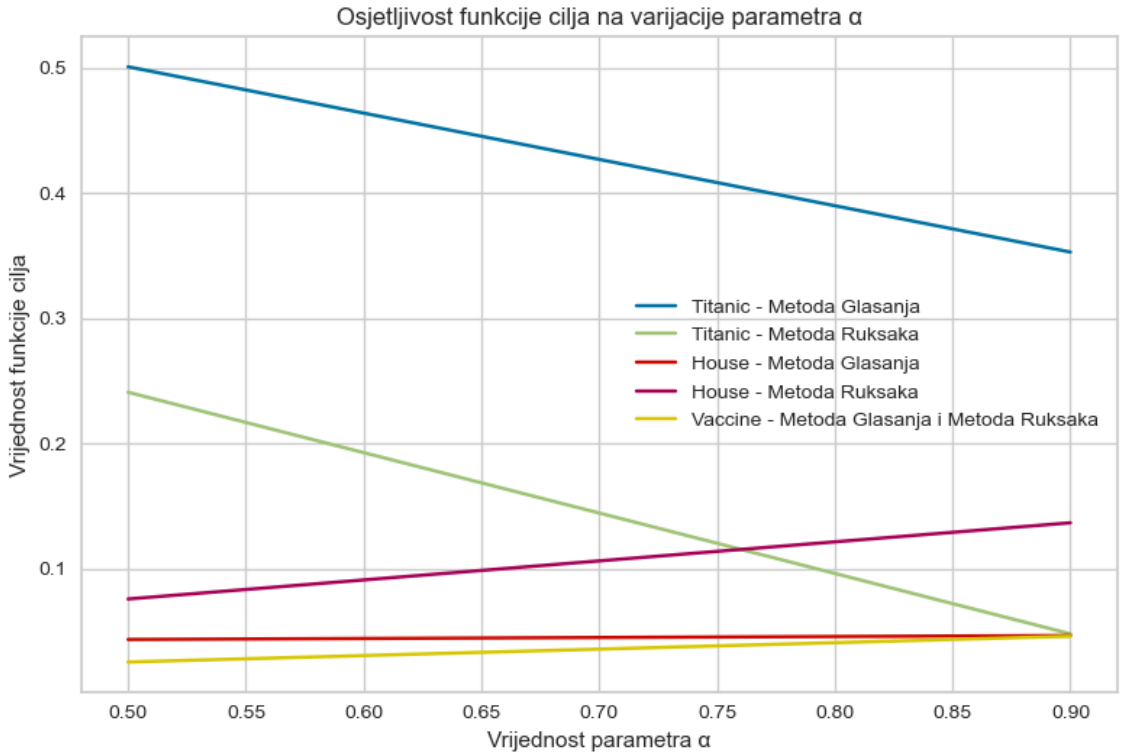
$$\begin{aligned} T_{norm} &= \frac{T - T_{\min}}{T_{\max} - T_{\min}} \\ I_{norm} &= \frac{I - I_{\min}}{I_{\max} - I_{\min}} \end{aligned} \tag{1}$$

Na temelju prethodne normalizacije, funkcija cilja se definirala kao linearna kombinacija normaliziranih vrijednosti:

$$F(T, I) = \alpha * T_{norm} + (1 - \alpha) * I_{norm} \tag{2}$$

U prethodno navedenoj funkciji cilja, manja vrijednost funkcije F(T,I) označava bolji balans između brzine i točnosti. Pomoću parametra α prilagođava se težina između prioriteta brzine (α bliže 1) i preciznosti (α bliže 0). Obzirom kako je prvi po rangu, kriterij, vrijeme izvršavanja odnosno brzina, autor ovog rada je parametar alpha (α) postavio na 0,7 (α = 0.7), s čime autor ističe prvi rang. Obrazloženje i grafički prikaz

odabira navedene vrijednosti 0,7 prikazano je kroz osjetljivost funkcije cilja na varijacije parametra alpha (α) u rasponu od 0,5 do 0,9 grafikonom na slici te objašnjeno kroz istraživački kontekst ovog doktorskog rada u nastavku.



Slika 2: Osjetljivost funkcije cilja na varijacije parametra alpha (α)

Iz dobivenog grafikona na prethodnoj slici vidljiv je prikaz osjetljivosti funkcije cilja na raspon $\alpha \in [0{,}5,\ 0{,}9]$ gdje se jasno prikazuju različita ponašanja *MoziaisMl* sustava ovisno o domeni skupa podataka i primjeni razvijenih metoda redukcije dimenzionalnosti – *Glasanja* i *Ruksaka*. U pojedinim slučajevima (primjer, Titanic – metoda Ruksaka), vrijednost funkcije cilja opada s porastom α, što znači da vrijeme izvršavanja prevladava u ukupnoj procjeni i ima znatno povoljniji omjer u usporedbi s gubicima u točnosti. U drugim slučajevima (primjer, House Prices – metoda Ruksaka), vrijednost funkcije cilja raste s porastom α, što upućuje na to da vrijeme izvršavanja nosi viši optimizacijski trošak u odnosu na marginalne dobitke u preciznosti. U slučaju metode *Glasanja* i *Ruksaka* za skup podataka Vaccine i metode *Glasanja* za House Prices skup podataka, gdje funkcija cilja ostvaruje, zanemariv, blagi rast, što implicira visok stupanj ravnoteže između vremenske učinkovitosti i stabilnosti predikcija unutar cjelokupnog raspona vrijednosti α.

Odabir vrijednosti $\alpha = 0,7$ utemeljen je na kontekstu primjene gdje je vremenska učinkovitost prioritet, ali ne na štetu točnosti predikcija. Vrijednost 0,7, unutar istraživanja ovog doktorskog rada, omogućuje uravnotežen kompromis, pri čemu 70% težine funkcije cilja dolazi iz domene vremena izvođenja, dok se 30% odnosi na očuvanje preciznosti.

Zaključno, osjetljivost funkcije cilja u odnosu na parametar α potvrđuje njenu primjenjivost i stabilnost unutar *MoziaisMl* sustava. Prikazana analiza, prethodnom slikom, opravdava izbor vrijednosti $\alpha = 0,7$ s aspekta operativne racionalnosti, pritom demonstrira robusnost i generalizacijsku sposobnost metodološkog pristupa u evaluaciji kompromisa između brzine i preciznosti s dobivenim optimizacijskim vrijednostima funkcije cilja u različitim poslovnim domenama.

1.3. Hipoteze istraživanja

U skladu s prethodno navedenim ciljem i svrhom istraživanja određuju se sljedeće hipoteze, s kojima se procjenjuje valjanost pretpostavke cilja i svrhe istraživanja.

Hipoteze:

H1. Prilagođena verzija CRISP-DM metodologije s ugrađenim algoritmom latencije za analitički informacijski sustav u produkcijskom okružju značajno brže, za više od 15%, izvršava predviđanja nad podatcima nego neprilagođena verzija CRISP-DM metodologije,

H2. Prilagođena verzija CRISP-DM metodologije s ugrađenim algoritmom latencije za analitički informacijski sustav u produkcijskom okružju ne smanjuje za i više od 4% točnost metrika predviđanja modela strojnog učenja nego neprilagođena verzija CRISP-DM metodologije.

Prva hipoteza koristila se za provjeru vremena izvršavanja odnosno brzine predviđanja, dok se druga hipoteza koristila za provjeru točnosti ili greške odnosno preciznosti. Izrađena prilagođena verzija CRISP-DM metodološkog okvira informacijskog sustava temeljenog na metodama strojnog učenja potvrđuje ili odbacuje obje hipoteze i tako određuje pouzdanost i primjenu razvijenog analitičkog

informacijskog sustava u produkcijskom okružju. Hipoteze su provjeravane na skupovima podataka klasifikacije i skupovima podataka regresije.

U skladu s problemima prethodnih istraživanja prva hipoteza (H1) polazi od pretpostavke da će primjena prilagođene CRISP-DM metodologije s ugrađenim algoritmom latencije rezultirati značajnim ubrzanjem procesa izvođenja predikcija u odnosu na neprilagođenu verziju CRISP-DM metodologije. Prethodno navedena tvrdnja temelji se na činjenici da *AutoML* sustavi, premda automatizirani, često se suočavaju s visokim računalnim zahtjevima i produženim vremenom izvođenja, osobito pri analizi visokodimenzionalnih skupova podataka[5]. Algoritam latencije razvijen u okviru ovog istraživanja, koristeći vlastite metode redukcije dimenzionalnosti – metoda *Glasanja* i metoda *Ruksaka* – smanjuje broj ulaznih značajki na optimalan podskup koji zadržava visoku informativnost odabranih značajki, čime se značajno smanjuje vrijeme potrebnog izvođenja modela. S obzirom na algoritam latencije, H1 procjenjuje da bi vrijeme izvođenja predikcije u produkcijskom okružju moglo biti smanjeno za više od 15%, čime se osigurava operativna prednost u stvarnim poslovnim sustavima produkcijskog okružja gdje latencija ima izravan utjecaj na donošenje odluka.

Druga hipoteza (H2) provjerava znanstveno utemeljenu pretpostavku da navedeno smanjenje vremenske složenosti neće uzrokovati značajan gubitak u prediktivnoj točnosti modela. Drugim riječima, polazi se od tvrdnje da prilagođena CRISP-DM metodologija s integriranim algoritmom latencije neće smanjiti točnost modela za više od 4% u usporedbi s neprilagođenom metodologijom. Ova hipoteza odgovara na pitanje ovog istraživanja u području primjene redukcije dimenzionalnosti – hoće li eliminacija značajki rezultirati neželjenim gubitkom točnosti.

1.4. Znanstveni doprinos istraživanja

Znanstveni doprinos se očituje u vidu doprinosa metodologiji za izradu analitičkog informacijskog sustava temeljenog na metodama strojnog učenja u produkcijskom okružju, doprinosa metodologiji koja integrira nedostajuće aspekte

klasičnih pristupa faza CRISP-DM-a te omogućuje precizniju sintezu teorijskih spoznaja ovog istraživanja kroz:

1. razvijeni algoritam latencije temeljen na pretprocesiranju skupa podataka i selekciji atributa skupa podataka kroz redukciju dimenzionalnosti
2. razvijenu prilagođenu verziju CRISP-DM metodologije za analitički informacijski sustav, odnosno za analizu podataka s algoritmom latencije predviđanja u analitičkom informacijskom sustavu u produkcijskom okružju.

Predložena prilagođena verzija CRISP-DM metodologije unapređuje analitički informacijski sustav u produkcijskom okružju nudeći dizajnersko konceptualno rješenje koje može poboljšati prilagodljivost sustava u produkcijskom okružju i može bitno dovesti do novih razvojnih okvira koji mogu značajno poboljšati razinu automatizacije analitičkih informacijskih sustava temeljenih na metodama strojnog učenja u produkcijskim okružjima. Postoje značajna istraživanja u smjeru u kojem ide cilj i svrha istraživanja ovog rada jer se smatra da se smanjenjem atributa/značajki može napraviti iskorak prema autonomnoj inteligenciji[24].

Razvijeni *MoziaisMl* sustav operacionalizira višekomponentni konceptualni okvir koji je metodološki utemeljen na načelima automatizacije, replikabilnosti i minimizacije vremenske složenosti. *MoziaisMl* sustav se sastoji od razvijenih komponenti, od kojih svaka komponenta obavlja precizno utvrđene funkcionalnosti, usklađene s ciljem razvoja automatiziranog algoritma latencije za produkcijsku primjenu unutar prilagođene verzije CRISP-DM metodologije.

U okviru komponente za automatiziranu pripremu podataka provodi se cjelovita transformacija ulaznih skupova podataka s ciljem osiguravanja njihove analitičke upotrebljivosti i kompatibilnosti sa zahtjevima algoritama strojnog učenja. Primjenom algoritamski definiranih postupaka detekcije i imputacije nedostajućih vrijednosti (*NaN*, *NA*, N/A, *NULL*, prazna vrijednost), uz istodobno kreiranje dodatnih binarnih indikatorskih varijabli (*_was_missing*), omogućuje se zadržavanje semantičke informacije o odsutnosti podataka, čime se umanjuje gubitak informacijske vrijednosti. Daljnji algoritamski definirani postupci uključuju identifikaciju i uklanjanje duplikata radi očuvanja strukturnog integriteta skupa podataka, kao i tekstualnu dekompoziciju složenih atributa u više stupaca, čime se omogućuje jednostavnija analiza složenih

tekstualnih informacijskih elemenata enkodiranjem takvih elemenata u više numerički interpretabilnih, za algoritme strojnog učenja, značajki.

U okviru komponente za redukciju dimenzionalnosti značajki implementirana je vlastito razvijena metoda *Glasanja* koja konsolidira rezultate kroz mehanizam glasovanja odnosno kolektivnog odlučivanja metoda *RandomForest (RF)[25, 26]*, *XGBoost (XGB)[27]*, *Mutual Information (MI)[28, 29]* i *Recursive feature elimination (RFE)[30]*, osiguravajući time robusniji skup značajki za niskolatentnu evaluaciju bez značajnog kompromisa u predikciji preciznosti. Zatim vlastito razvijena metoda *Ruksaka* je optimizacijski algoritam temeljen na paradigmama kombinatorne optimizacije algoritma ruksaka (engl. *Knapsack algorithm*), pri čemu se svaka značajka promatra kao objekt s atribuiranom vrijednošću (važnost značajke) i težinom (varijanca značajke), a selekcija se provodi unutar unaprijed definiranog kapaciteta složenosti modela odnosno ukupne definirane maksimalne težine (varijance) značajki koje se mogu smjestiti u ruksak. Pristup vlastito razvijene metode *Ruksaka* omogućuje balansiranje između predikcijske učinkovitosti i latentne složenosti modela. Rezultat komponente je redukcija dimenzionalnosti uz očuvanje prediktivne moći, čime se doprinosi ukupnom cilju *MoziaisMl* sustava – konstrukciji modela s minimiziranom latencijom i maksimalno moguće očuvanom točnošću, što je osobito važno u kontekstu automatiziranog strojnog učenja u okruženjima s velikim količinama podataka (engl. *Big Data*) i potrebom za brzim predikcijama u produkcijskim okružjima.

U kontekstu predmetnog istraživanja, automatizacija *MoziaisMl* sustava se ne odnosi na generički koncept automatske primjene algoritama strojnog učenja, već na konkretno razvijene, programskim kodom, module i procedure koji su dizajnirani s ciljem smanjenja potrebe za ručnom intervencijom u svim fazama podatkovne analize, sukladno fazama prilagođenog CRISP-DM metodološkog okvira. Automatizirani su sljedeći procesi: pretprocesiranje i priprema podataka (uključujući detekciju i imputaciju nedostajućih vrijednosti, razdvajanje tekstualnih atributa, dinamički odabir tehnike enkodiranja kategorijskih varijabli te kvantilnu transformaciju distribucije značajki), selekcija značajki u dimenzionalno reduciranom prostoru putem razvijenih metoda *Glasanja* i *Ruksaka*, kao i modeliranje i optimizacija hiperparametara pomoću metode *GridSearchCV*[31] s višestrukom *K-Fold*[32, 33] unakrsnom validacijom.

MoziaisMl sustav također automatski bilježi metapodatke, uključujući vrijeme treniranja, vrijeme inferencije kod predikcije i metrike evaluacije, te omogućuje iterativnu evaluaciju klasifikacijskih i regresijskih modela u dvostrukom modalitetu (*task_type* = „*classification*“ ili „*regression*“). Kroz te mehanizme, *MoziaisMl* u potpunosti uklanja potrebu za ručnim kodiranjem faza eksperimenta, čime omogućuje replikabilnu, vremenski optimiziranu i znanstveno validiranu analizu podataka, u skladu s postavljenim ciljem i svrhom istraživanja vezanim uz latencijsku učinkovitost, redukciju dimenzionalnosti i generalizacijsku sposobnost predikcijskih modela. Stoga se pojam „automatizirano“ u ovom radu odnosi na sustavno programsko izvršavanje modularnih koraka koji su tradicionalnim analizama podataka zahtijevali ručnu konfiguraciju, parametrizaciju i heurističku analizu.

1.5. Struktura istraživanja i rada

Drugo poglavlje daje pregled relevantne literature i istraživanja odnosno analize literature povezanom s temom doktorskog rada.

Treće poglavlje predstavlja teorijske osnove četiri korištene i popularne metode redukcije dimenzionalnosti –PCA[14-16], t-SNE[14-17], UMAP[18] i autoenkoderske neuronske mreže[19].

Četvrto poglavlje predstavlja razvijeni automatizirani algoritam latencije te na kojim principima se temelji.

Peto poglavlje predstavlja razvijenu metodologiju prema prilagođenom CRISP-DM metodološkom okviru s vlastito razvijenim algoritmom latencije koji uključuje razvijene metode za pretprocesiranje podataka i razvijene metode redukcije dimenzionalnosti.

Šesto poglavlje donosi rezultate provedenog istraživanja upotrebe algoritma latencije s razvijenim metodama za pretprocesiranje podataka i razvijenim metodama, *Glasanja* i *Ruksaka,* redukcije dimenzionalnosti kroz prilagođenu verziju CRISP-DM metodološkog okvira. Osim rezultata upotrebe razvijenih metoda provodi se i analiza generalizacijskih sposobnosti usporednom analizom učinkovitosti razvijenih metoda

Glasanja i *Ruksaka* redukcije dimenzionalnosti s popularnim metodama redukcije dimenzionalnosti - PCA[14-16], t-SNE[14-17], UMAP[18] i autoenkoderskim neuronskim mrežama [19].

Zaključno poglavlje donosi zaključke o cilju, realizaciji provedenog istraživanja ovog rada te rezultatima razvijenih metoda za pretprocesiranje podataka i razvijenih metoda redukcije dimenzionalnosti kroz prilagođenu verziju CRISP-DM metodološkog okvira te potvrđivanje hipoteza.

2. PREGLED RELEVANTNE LITERATURE U PODRUČJU AUTOMATIZIRANOG STROJNOG UČENJA, REDUKCIJE DIMENZIONALNOSTI I METODOLOŠKIH OKVIRA

Poglavlje pregled relevantne literature i istraživanja predstavlja provedenu analizu znanstvenih i stručnih radova te ostalih publikacija izravno povezanih s temom istraživanja – automatizirano strojno učenje, redukcija dimenzionalnosti i CRISP-DM - u doktorskom radu. Svrha provođenja analize postojećih istraživanja u području automatiziranog strojnog učenja, redukcije dimenzionalnosti i primijenjenih metodoloških okvira bila je utvrđivanje relevantnosti, za temu istraživanja, korištenih metoda, ciljeva, rezultata, izazova i primjene. Zatim, naglašavaju se ograničenja istraživanja analiziranih znanstvenih i stručnih radova te ostalih publikacija, predstavljaju se neistražena područja istraživanja, te postoje li prostori za dodatna istraživanja u području latencije.

2.1. Pretraga relevantne literature

Predmet pregleda literature je istraživanje i analiziranje pristupa razvoju i dizajnu *AutoML* sustava, uključujući arhitekturu i integraciju modula za inženjerstvo značajki, redukciju dimenzionalnosti, modeliranju i optimizaciji hiperparametra te evaluaciju i implementaciju korištenjem izvornog, proširenog ili prilagođenog CRISP-DM okvira. Istraživanje će se provesti na temelju postojećih znanstvenih članaka pretragom sljedećih tehničkih baza podataka:

- SCOPUS,
- Web of Science.

Pretraga prethodno navedenih baza podataka provedena je postavljanjem upita koji sadrže ključne riječi predmetnog područja teme istraživanja. Prvi upit (U1) traži radove koji analiziraju korištenje izvorne, proširene ili prilagođene CRISP-DM metodologije s *AutoML* sustavima. Drugi upit (U2) traži radove koji obrađuju temu razvoja sustava zasnovanih u kontekstu automatiziranog strojnog učenja te se odnose na

cjeloviti (engl. *full-lifecycle*) razvojni proces ili cjevovod (engl. *pipeline*) strojnog učenja, dok treći upit (U3) traži radove koji obrađuju teme inženjerstva značajki, redukcije dimenzionalnosti, optimizacije hiperparametara i odabira modela strojnog učenja. Upiti se nalaze u nastavku:

U1. (automl OR "automated ML" OR "automated machine learning") AND ("CRISP-DM" OR "CRISP DM" OR "CRoss Industry Standard Process for Data Mining" OR "CRISP ML(Q)" OR "CRISP ML")

U2. (AutoML OR "automated ML" OR "automated machine learning") AND ("end-to-end" OR "full-lifecycle" OR "pipeline" OR "automated workflow")

U3. (AutoML OR "automated ML" OR "automated machine learning") AND ("model development lifecycle" OR "hyperparameter tuning" OR "feature engineering" OR "feature selection" OR "dimensionality reduction" OR "reducing dimensionality")

Osim prethodno navedenih tehničkih baza znanstvenih članaka za istraživanje u radu su se koristili i svi ostali dostupni izvori (poput disertacija, raspoloživih studija slučaja) putem interneta koji su obrađivali izvorne, proširene ili prilagođene CRISP-DM metodologije, *AutoML* sustave, inženjerstvo značajki, redukciju dimenzionalnosti, optimizaciju hiperparametara i odabir modela strojnog učenja.

Najviše dobivenih rezultata, kroz tri upita, bilo je u *Web of Science* bazi znanstvenih radova i tehničke literature, 710, dok je 629 radova bilo u *SCOPUS* bazi. Točan broj dobivenih rezultata prikazan je tablicom u nastavku.

Baza	U1	U2	U3
SCOPUS	92	299	238
Web of Science	6	380	324

Tablica 2: Dobiveni rezultati postavljenih upita po bazama

S obzirom da se istraživano područje bavi utvrđivanjem pristupa razvoju i dizajnu sustava za automatizaciju cjelovitog procesa strojnog učenja, uključujući arhitekturu i

integraciju modula za inženjerstvo značajki, redukciju dimenzionalnosti, modeliranju i optimizaciji hiperparametra te evaluaciju i implementaciju korištenjem izvornog, proširenog ili prilagođenog CRISP-DM okvira, posebno su analizirani i indeksirani članci autora na tu temu, a samo mali broj članaka je uklonjen iz početnih rezultata upita jer nisu bili dovoljno povezani s predmetnom temom istraživanog područja.

Dobiveni rezultati dodatno su pregledani i međusobno uspoređeni kako bi se uklonili znanstveni članci, konferencijski članci i ostali radovi koji su indeksirani u obje znanstvene baze. Također, treba istaknuti da upit vraća samo rezultate koji pokrivaju u potpunosti područje istraživanja koje je predmet istraživanog područja. Prethodno navedeni zaključci se temelje na pregledanim ključnim riječima i naslovima radova unutar dobivenih rezultata.

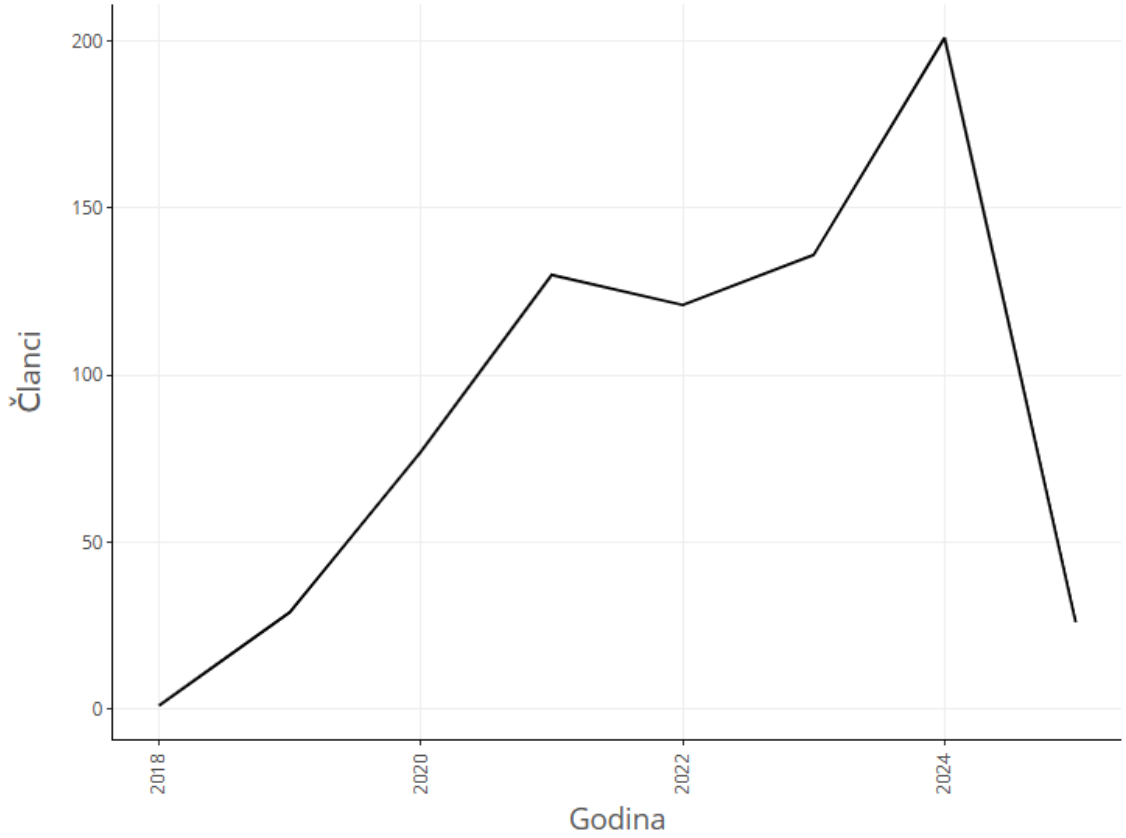
Nakon temeljite analize ključnih riječi i naslova znanstvenih radova, provedena je deduplikacija članaka koji su istovremeno indeksirani u obje odabrane znanstvene baze podataka. Ovim procesom osigurana je točnost i jedinstvenost skupa podataka znanstvenih članaka, čime su eliminirani redundantni zapisi koji bi mogli utjecati na valjanost rezultata istraživanja. Nakon završne filtracije, utvrđeno je da ukupan broj članaka koji ispunjavaju kriterije za daljnju analizu iznosi 725. Ovaj korak predstavlja ključnu fazu u osiguravanju metodološke dosljednosti istraživanja, čime se osigurava da bibliometrijska analiza obuhvati samo one znanstvene radove koji pružaju značajne i neponovljene doprinose unutar istraživanog područja.

Vremensko razdoblje obuhvaćenih članaka bilo je od 2018. godine do 2025. godine. Prosječna stopa rasta produkcije u prethodnom navedenom vremenskom razdoblju 59,27%, ukupno je bilo 2.981 autora s objavama radova iz 526 izvora, međunarodna suradnja na pisanju znanstvenih članaka iznosila je 16,14%, prosječan broj koautora na napisanim člancima je 5,25, svi obuhvaćeni članci imaju 1.905 autorovih ključnih riječi (deskriptora, engl. *Descriptors*), koje omogućuju brže pretraživanje i klasifikaciju radova u znanstvenim bazama podataka, starost svakog znanstvenog članka je 2,67 godine, a njihova prosječna citiranost 14,86. S obzirom na veliki broj dobivenih rezultata, daljnja analiza se vršila prema sažetcima te broju citata radova na temelju kojih je izdvojena literatura koja je poslužila kao podloga za donošenje zaključka o stanju i prazninama (engl. *research gap*) istraživanog područja. Na slici, u nastavku, su prikazane

osnovne informacije o reduciranim člancima nakon pregleda ključnih riječi i naslova radova.

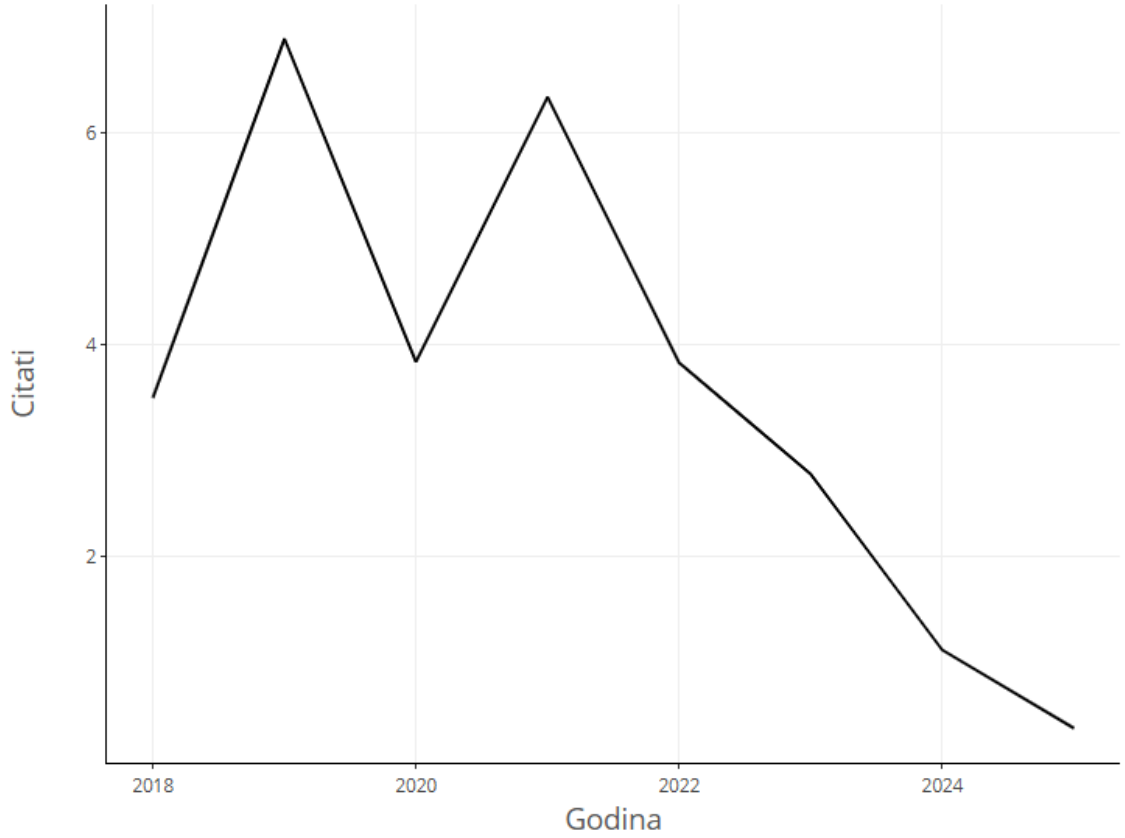
Slika 3: Osnovne informacije o člancima

Godišnja znanstvena produkcija prikazuje kvantitativni prikaz dinamike objavljivanja znanstvenih radova unutar vremenskog razdoblja od 2018. do 2025. godine. Navedena metrika omogućuje uvid u evoluciju istraživačkog područja kroz analizu broja publikacija objavljenih svake godine, od 2018. do 2025., čime se mogu identificirati trendovi u znanstvenoj produkciji istraživanog područja, periodi intenzivnog istraživanja te moguće stagnacije ili opadanja interesa za tematiku istraživanog područja. Godišnja znanstvena produkcija prikazana je, slikom, u nastavku.



Slika 4: Godišnja znanstvena produkcija

Na temelju analize godišnje znanstvene produkcije, vidljivo je da je u razdoblju od 2018. do 2021. zabilježen značajan porast broja znanstvenih publikacija povezanih s istraživanim područjem, što ukazuje na povećani interes istraživača te moguću ekspanziju istraživanja tematike istraživanog područja pristupa razvoju i dizajnu sustava za automatizaciju cjelovitog procesa strojnog učenja, uključujući arhitekturu i integraciju modula za inženjerstvo značajki, redukciju dimenzionalnosti, modeliranju i optimizaciji hiperparametra te evaluaciju i implementaciju korištenjem izvornog, proširenog ili prilagođenog CRISP-DM okvira. U razdoblju od 2021. do 2022. zamjećuje se blaga stagnacija znanstvene produkcije, dok od 2022. do 2024. godine ponovno je zabilježen rast znanstvene produkcije istraživanog područja. Važno je napomenuti da pad u znanstvenoj produkciji zabilježen za 2025. godinu se mora tumačiti s oprezom, budući da godina još nije završila te su podatci nepotpuni, što, trenutno, rezultira smanjenim prikazom broja publikacija. U nastavku je slikovni prikaz prosječnog broja citata po godini.

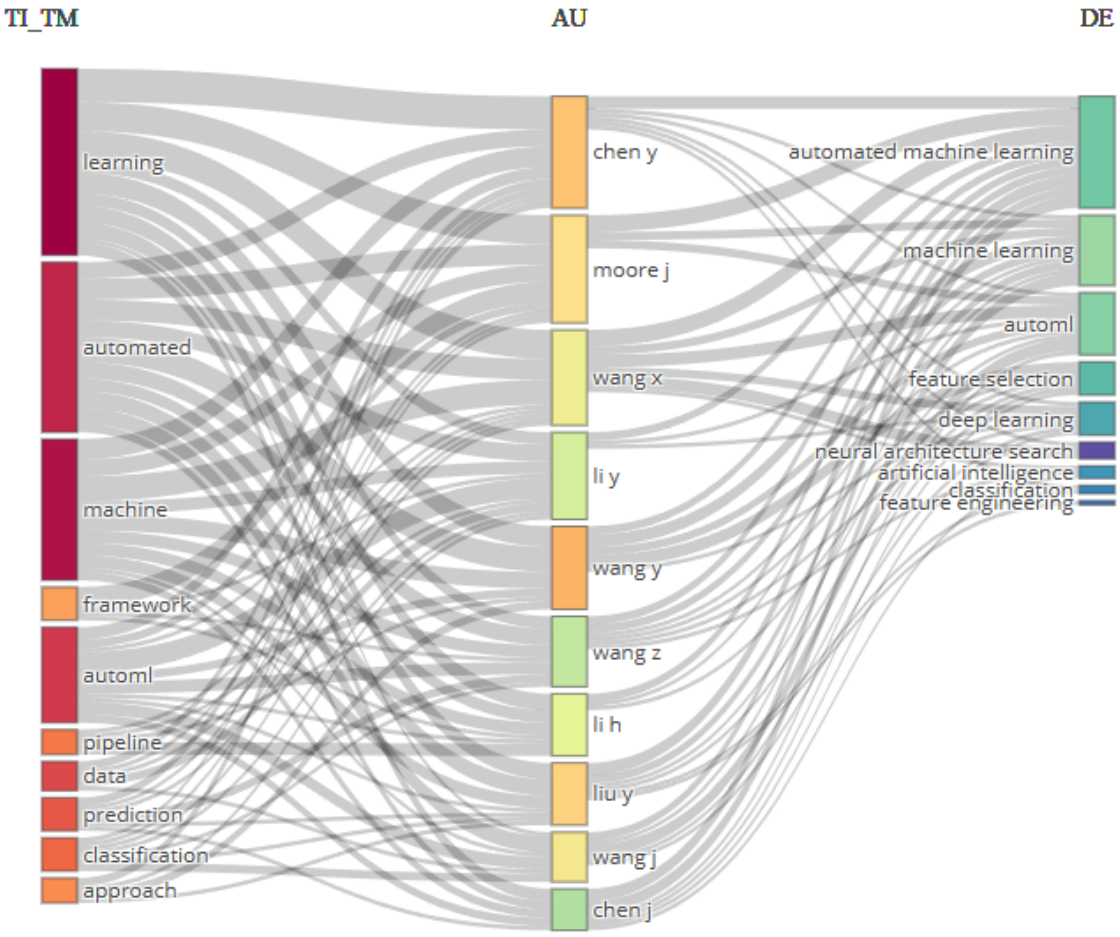


Slika 5: Prosječan broj citata po godini

Na temelju slikovnog prikaza prosječnog broja citata po godini može se uočiti dinamika citiranosti znanstvenih radova unutar analiziranog razdoblja od 2018. do 2025. godine. U početnim godinama, od 2018. do 2020., primjetan je uzlazni trend, gdje broj citata doseže svoj vrhunac, što sugerira da su radovi objavljeni u tom razdoblju imali značajan znanstveni odjek. Nakon početnog rasta, u 2020. dolazi do blagog pada citiranosti, koji se zatim ponovno stabilizira i dostiže novi vrhunac u 2021. godini. Nakon 2021. godine primjetan je kontinuirani pad prosječnog broja citata, koji postaje izraženiji prema 2024. godini. Nadalje, radovi objavljeni u posljednjim godinama, 2024. i 2025., razumljivo imaju niži prosječan broj citata jer nije prošlo dovoljno vremena da ostvare značajniji znanstveni doseg.

U nastavku je prikazan dijagram s tri polja: TI_TM, AU i DE. Polje TI_TM (engl. (engl. *Title Thematic Mapping*) predstavlja tematske izraze i naslova dokumenata te je kombinacija dva sljedeća polja: naslova (engl. *Title*, TI) koje označava naslov rada, odnosno fraze i ključne riječi koje autori koriste u naslovima svojih znanstvenih publikacija i tematsko mapiranje (engl. *Thematic Mapping*, TM), koje se odnosi se na

analizu tematskih obrazaca, pri čemu se fraze iz naslova grupiraju i analiziraju kako bi se identificirali dominantni istraživački trendovi. Ostala dva polja AU - autori i DE – ključne riječi deskriptori. U dijagramu s tri polja međusobno su povezana, prethodno navedena tri polja, kako bi se prikazale relacije tematskih izraza i vizualiziralo koji autori (AU) koriste određene tematske izraze u naslovima svojih radova (TI_TM) te koje ključne riječi (engl. *Author Keywords, Descriptors,* DE) su povezane s tematskim izrazima iz naslova (TI_TM). [34] Za parametre je odabrano 10 tematskih izraza, autora i ključnih riječi, dok je slikovni prikaz u nastavku. [34]



Slika 6: Dijagram tri polja

Analizom rezultata vidljivo je da izraz *automated* unutar naslova radova (TI_TM) dominira s 41 pojavljivanjem, dok *AutoML*, *framework* i *pipeline* također imaju značajnu prisutnost – 23, 8 i 6, respektivno. Ove tematske fraze nemaju *incoming flow*, što znači da nisu generirane iz drugih analiziranih entiteta, već su direktno korištene u naslovima radova. S druge strane, imaju *outgoing flow* prema autorima, što sugerira da su različiti

autori koristili iste tematske izraze pri formuliranju naslova svojih radova. U srednjem dijelu dijagrama prikazani su autori (AU) koji su objavili radove s prethodno navedenim tematskim izrazima u naslovima. Autori poput Chen Y., Moore J., Wang X., i Li Y. povezani su s tematskim izrazima iz TI_TM, što upućuje na njihov doprinos znanstvenoj produkciji unutar istraživanog područja. Povezanost, prikazanih autora s dijagrama, s više tematskih izraza i ključnih riječi sugerira interdisciplinarnost i širok istraživački doseg unutar domene automatiziranog strojnog učenja, što potvrđuje pojavljivanje od 41 i 23 puta *Automated Machine Learning* i *AutoML* ključnih riječi, respektivno.

U desnom dijelu dijagrama prikazane su ključne riječi autora (engl. *Descriptors*, DE), koje autori dodjeljuju svojim radovima zbog bržeg pretraživanja i klasifikacije radova u znanstvenim bazama podataka. Ključne riječi poput *automated machine learning* (27 pojavljivanja, 10 autora), *AutoML* (15 pojavljivanja, 9 autora) i *feature selection* (8 pojavljivanja, 6 autora) jasno ukazuju na dominantne istraživačke teme.

Analiza dijagrama tri polja otkriva bitan obrazac za istraživačko područje - konzistentnost istraživačkih tema. Suštinski, izrazi korišteni u naslovima radova visoko koreliraju s ključnim riječima koje autori sami dodjeljuju svojim radovima, a to ukazuje na snažnu usklađenost istraživačkih interesa i njihove terminološke reprezentacije u znanstvenim bazama podataka – SCOPUS i WoS.

Nadalje, Analiza najrelevantnijih izvora publikacije u okviru istraživanog područja omogućila je identifikaciju znanstvenih časopisa koji su najzastupljeniji za predmetnu tematiku.

Rbr.	Izvor publikacije	Broj članaka
1	IEEE ACCESS	16
2	SCIENTIFIC REPORTS	10
3	APPLIED SCIENCES (SWITZERLAND)	8
4	REMOTE SENSING	7
5	DIAGNOSTICS	6
6	ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE	6
7	IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE	6
8	PLOS ONE	6
9	APPLIED SOFT COMPUTING	5
10	BIOINFORMATICS	5

Tablica 3: Najrelevantniji izvori publikacija

Na temelju prikazane tablice zaključuje se da je časopis *IEEE Access* najproduktivniji izvor, s 16 objavljenih radova, što ukazuje na njegovu značajnu ulogu u objavljivanju istraživanja povezanih s predmetnom tematikom. Slijedi časopis *Scientific Reports* s 10 radova, koji je poznat po širokom spektru multidisciplinarnih istraživanja, dok su *Applied Sciences (Switzerland)* i *Remote Sensing* također visoko rangirani, s 8 odnosno 7 radova. Časopisi poput *Diagnostics*, *Engineering Applications of Artificial Intelligence*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PLOS ONE*, *Applied Soft Computing* i *Bioinformatics*, svaki s 5 ili 6 objavljenih radova, dodatno naglašavaju interdisciplinarnost istraživanog područja, pokrivajući područja umjetne inteligencije, biomedicinskih znanosti i naprednih algoritama za obradu podataka.

Na temelju najrelevantnijih izvora publikacija, distribucija članaka ukazuje na raznoliku prirodu istraživanja unutar analizirane domene te sugerira da se relevantni znanstveni rezultati objavljuju u publikacijama, koji kombiniraju tehničke, primijenjene i teoretske aspekte istraživanja. Nadalje, prisutnost publikacija iz područja umjetne inteligencije, procesiranja podataka i biomedicine sugerira da analizirano područje istraživanja ima široku primjenu u različitim područjima znanosti - područje prirodnih znanosti, područje tehničkih znanosti i područje biomedicine i zdravstva.

U nastavku, analiza najcitiranijih lokalnih izvora unutar područja istraživanja identificira znanstvene publikacije koje su unutar analiziranog skupa 725 članaka najčešće citirane. Navedena metrika pruža uvid u temeljne izvore znanstvenih spoznaja i omogućuje razumijevanje dominantnih publikacijskih izvora koji su oblikovali istraživanje unutar specifičnog područja.[34]

Rbr.	**Izvor publikacije**	**Broj članaka**
1	ARXIV	695
2	THE JOURNAL OF MACHINE LEARNING RESEARCH (JMLR)	468
3	LECTURE NOTES IN COMPUTER SCIENCE (LNCS)	352
4	ADVANCED NEUROLOGY	344
5	SPRING SER CHALLENGE	313
6	PROCEEDINGS OF IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION	267

7	PROCEEDINGS OF MACHINE LEARNING RESEARCH	263
8	AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE	199
9	MACHINE LEARNING	186
10	IEEE ACCESS	176

Tablica 4: Najcitiraniji lokalni izvori

Prema prethodno prikazanoj tablici, *arXiv* je najcitiraniji izvor, s ukupno 695 citata, što sugerira da su *pre-print/pre-release* radovi, često dostupni prije formalne objave u znanstvenim časopisima, od velike važnosti za predmetno istraživačko područje. Slijedi *The Journal of Machine Learning Research (JMLR)* s 468 citata. Također, *Lecture Notes in Computer Science (LNCS)* s 352 citata potvrđuje svoju relevantnost kao značajan izvor konferencijskih radova i akademskih istraživanja u području računalnih znanosti.

Nadalje, prisutnost publikacija poput *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition* (267 citata) i *Proceedings of Machine Learning Research* (263 citata) naglašava važnost konferencijskih radova u prijenosu novih istraživačkih otkrića u području strojnog učenja. Visoka citiranost publikacija *AAAI Conference on Artificial Intelligence* (199 citata) i *Machine Learning* (186 citata) ukazuje na stabilnu poziciju časopisa i konferencija u generiranju temeljnih znanstvenih spoznaja. Na kraju, časopis *IEEE Access* (176 citata) dodatno potvrđuje značaj interdisciplinarnih publikacija koje omogućuju široku diseminaciju rezultata predmetnog istraživačkog područja.

U nastavku je prikazan Bradfordov zakon distribucije znanstvene literature određuje jezgrene izvore (engl. *core sources*) koji objavljuju najveći broj radova unutar predmetnog istraživačkog područja. Metoda Bradfordov zakon distribucije kategorizira izvore u zone prema njihovoj produktivnosti – središnja zona (Zone 1) sadrži najrelevantnije časopise koji objavljuju najveći broj radova na određenu temu, a u predmetnoj analizi znanstvene literature, u središnjoj zoni se nalaze 73 časopisa.[34, 35]

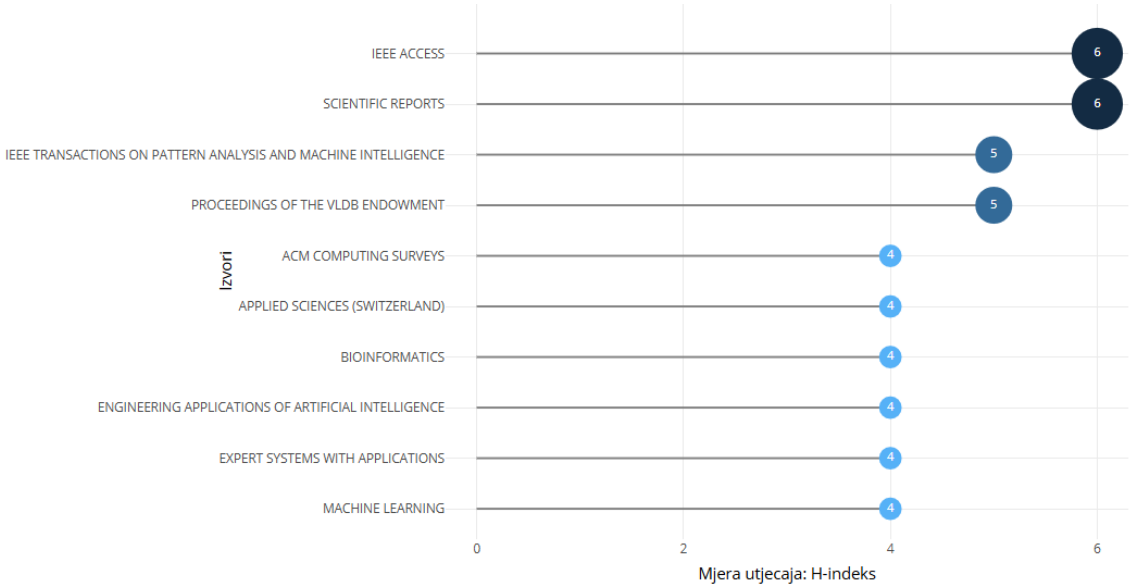
Rbr.	**Izvor (engl. Source, SO)**	**Rang**	**Frekvencija**	**Kumulativna frekvencija**	**Zona**
1	IEEE ACCESS	1	16	16	Zona 1

2	SCIENTIFIC REPORTS	2	10	26	Zona 1
3	APPLIED SCIENCES (SWITZERLAND)	3	8	34	Zona 1
4	REMOTE SENSING	4	7	41	Zona 1
5	DIAGNOSTICS	5	6	47	Zona 1
6	ENGINEERING APPLICATIONS OF ARTIFICIAL INTELLIGENCE	6	6	53	Zona 1
7	IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE	7	6	59	Zona 1
8	PLOS ONE	8	6	65	Zona 1
9	APPLIED SOFT COMPUTING	9	5	70	Zona 1
10	BIOINFORMATICS	10	5	75	Zona 1

Tablica 5: Prvih 10 izvora prema Bradfordovom zakonu

U prethodno navedenoj tablici, izvor (engl. *Source*, SO) je naziv znanstvenog časopisa u kojem su objavljeni radovi, rang (engl. *Rank*) je rang časopisa prema broju objavljenih radova, gdje 1 označava najproduktivniji izvor, frekvencija (engl. *Freq*) je broj radova objavljenih u tom časopisu u sklopu analiziranog skupa podataka, kumulativna frekvencija (engl. *cumFreq*) pokazuje kako se radovi akumuliraju kroz rangirane izvore, omogućujući analizu distribucije produktivnosti te zone (engl. *Zone*), prema Bradfordovoj distribuciji, grupiraju časopise (izvore) prema produktivnosti,, gdje Zona 1 predstavlja najproduktivnije izvore koji zajedno sadrže najveći broj objavljenih radova.[34, 35]

Zatim, vizualizacija lokalnog utjecaja izvora (engl. *Sources' Local Impact*) prema H-indeksu omogućuje procjenu znanstvenog doprinosa pojedinih publikacijskih kanala unutar predmetnog istraživačkog područja. Mjera H-indeks izvora temelji se na broju radova objavljenih u određenom časopisu i broju njihovih citata, pri čemu viša vrijednost H-indeksa ukazuje na veću vidljivost i utjecaj radova objavljenih u tom izvoru unutar analiziranog skupa podataka. Slikom, u nastavku, prikazana je analiza lokalnog utjecaja izvora prema H-indeksu. [34]



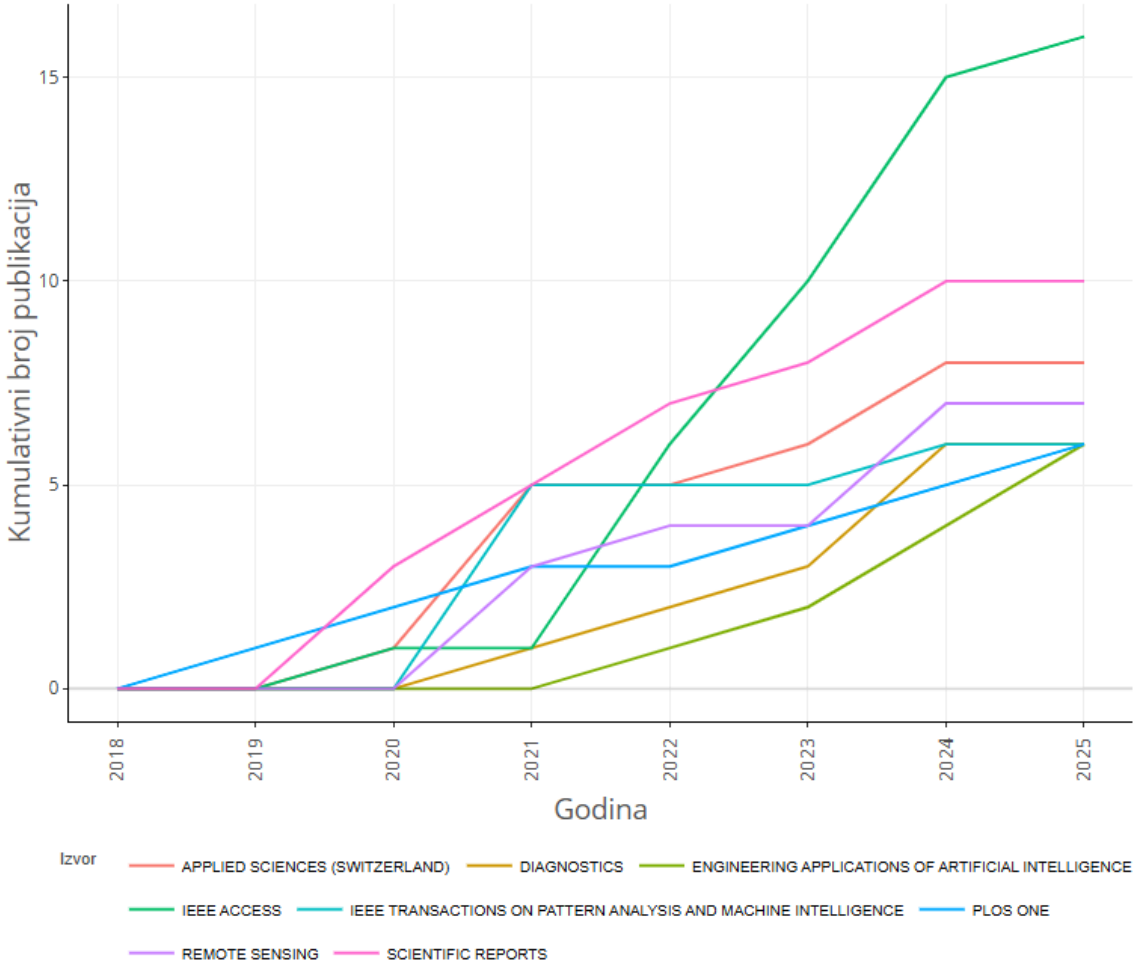
Slika 7: Analiza lokalnog utjecaja izvora prema H-indeksu

Prema prikazanoj slici, najveći lokalni H-indeks (H = 6) imaju časopisi *IEEE Access* i *Scientific Reports*, što sugerira da radovi objavljeni u ovim izvorima imaju najveći utjecaj u okviru analiziranog skupa 725 publikacija. Časopisi *IEEE Transactions on Pattern Analysis and Machine Intelligence* te *Proceedings of the VLDB Endowment* slijede s H = 5, što ukazuje na njihovu značajnu ulogu u objavljivanju radova koji su široko citirani. Nadalje, časopisi *ACM Computing Surveys, Applied Sciences (Switzerland)* i *Bioinformatics* imaju H-indeks 4, što sugerira da su njihovi radovi relevantni, ali s nešto manjim brojem citiranih publikacija u usporedbi s vodećim izvorima.

Nadalje, prisutnost časopisa *Engineering Applications of Artificial Intelligence*, *Expert Systems with Applications* i *Machine Learning*, također s H = 4, ukazuje na interdisciplinarnu prirodu istraživačkog područja te na širok raspon časopisa koji doprinose razvoju tematike predmetnog istraživanja. Rezultati pokazuju kako su visoko citirani radovi unutar analiziranog skupa članaka objavljeni u kombinaciji multidisciplinarnih časopisa, poput, *Scientific Reports* i *IEEE Access* te specijaliziranih izvora iz područja umjetne inteligencije, strojnog učenja i informacijskih sustava.

U nastavku, prikazana slika produkcije znanstvenih izvora kroz vrijeme (engl. *Sources' Production over Time*) pruža uvid u kumulativni broj publikacija u pojedinim znanstvenim časopisima unutar analiziranog skupa podataka tijekom vremenskog razdoblja od 2018. do 2025. godine. Navedena analiza omogućuje prepoznavanje

trendova u znanstvenoj produkciji i identifikaciju najutjecajnijih publikacijskih izvora u specifičnom istraživačkom polju.[34]



Slika 8: Produkcija znanstvenih izvora kroz vrijeme

Na temelju slike, primjećuje se kontinuirani rast broja publikacija u gotovo svim časopisima, pri čemu su pojedini izvori zabilježili znatno veći porast znanstvene produkcije u odnosu na druge. Časopis *IEEE Access* pokazuje najizraženiji rast, posebno u razdoblju nakon 2021. godine, što upućuje na njegovu sve veću važnost u objavljivanju istraživanja unutar analiziranog predmetnog istraživačkog područja. Sličan trend rasta vidljiv je i u časopisima *Scientific Reports* te *IEEE Transactions on Pattern Analysis and Machine Intelligen*ce, što ukazuje na njihovu stabilnu prisutnost u diseminaciji znanstvenih rezultata istraživačkog područja. Ostali časopisi - *Applied Sciences (Switzerland), Engineering Applications of Artificial Intelligenc*e i *Remote Sensing*, također pokazuju konzistentan rast produkcije, ali s nešto umjerenijom dinamikom u odnosu na vodeće izvore. Vidljivo je da su određeni časopisi, primjerice *Diagnostics* i

PLOS ONE, zabilježili razdoblja stagnacije, gdje se njihov kumulativni broj objavljenih radova nije značajno mijenjao u određenim godinama, što može upućivati na specifične promjene u fokusu istraživanja ili strategijama objavljivanja.

Rast popularnosti časopisa *IEEE Access* i *Scientific Reports* može se djelomično objasniti njihovim otvorenim pristupom radovima (engl. *open access*), što omogućuje širu vidljivost i dostupnost objavljenih radova i vjerojatno povećava njihovu citiranost.

Analiza najrelevantnijih autora (engl. *Most Relevant Authors*) prikazana je u nastavku, prikazujući broj objavljenih radova po autoru te njihovu frakcioniziranu vrijednost. Navedena analiza omogućila je identifikaciju vodećih istraživača unutar predmetnog znanstvenog područja i pružila uvid u njihov znanstveni doprinos.

Frakcijsko atributiranje raspoređuje zaslugu za objavljene radove među koautorima. Umjesto da se svakom autoru pripiše puni broj radova u kojima je sudjelovao, broj publikacija se frakcionira prema broju koautora. Primjer, ako rad ima dva autora, svaki od njih dobiva 0,5 zasluge za taj rad; ako ih ima pet, svaki autor dobiva 0,2, itd. Navedena metoda pruža precizniju sliku o stvarnom doprinosu svakog autora znanstvenoj produkciji.[36] Najproduktivniji autori su prikazani tablicom u nastavku.

Autori	**Članci**	**Frakcijsko atribuiranje**
WANG Y	12	2,49
CHEN Y	10	1,25
LIU Y	10	1,79
MOORE J	10	1,76
WANG J	10	2,05
WANG X	10	1,72
LI H	9	1,39
LI Y	9	1,73
WANG Z	9	1,67
CHEN J	8	1,07

Tablica 6: Prikaz najrelevantnijih autora

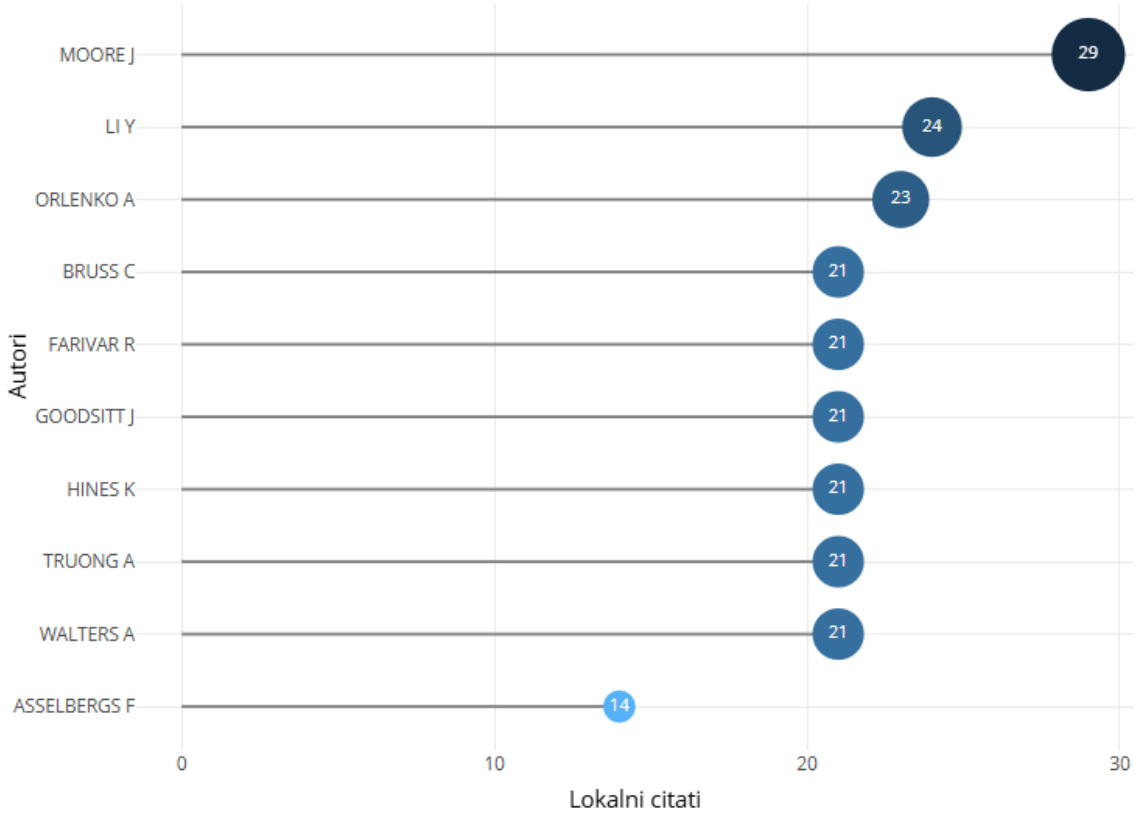
Najproduktivniji autor u analiziranom skupu radova je Wang Y, koji je objavio 12 radova, s frakcioniziranom vrijednošću 2,49, što ukazuje na njegov značajan znanstveni doprinos, čak i kada se uzme u obzir podjela zasluga među koautorima. Drugim riječima, najproduktivniji autor Wang Y ima 1 rada s 1 koautorom, 1 rad s 2 koautora, 2 rad s 3 koautora, 3 rad s 4 koautora, 1 rad s 5 koautora, 1 rad s 7koautora, 1 rad s 9 koautora, 1

rad s 11 koautora i 1 rad s 12 koautora te ima doprinos od 2,49 samostalna rada koji se dobio formulom[36]:

$$\begin{aligned} \textit{frakcijsko atributiranje} = &\left(1\times\frac{1}{2}\right)+\left(1\times\frac{1}{3}\right)+\left(2\times\frac{1}{4}\right)+\left(3\times\frac{1}{5}\right)+\left(1\times\frac{1}{6}\right)+\left(1\times\frac{1}{8}\right) \\ &+\left(1\times\frac{1}{10}\right)+\left(1\times\frac{1}{12}\right)+\left(1\times\frac{1}{13}\right)=2,49 \end{aligned} \tag{3}$$

Autori poput Chen Y, Liu Y, Moore J, Wang J i Wang X objavili su 10 radova, ali imaju različite frakcionizirane vrijednosti, što sugerira različit stupanj suradnje i koautorskih mreža. Na primjer, Wang J (2,05) ima višu frakcioniziranu vrijednost u usporedbi s Chen Y (1,25), što znači da je surađivao u radovima s manjim brojem koautora. Nadalje, autori Li H, Li Y, Wang Z i Chen J imaju po 9 ili 8 radova, ali njihove frakcionizirane vrijednosti variraju između 1,07 i 1,73, što također ukazuje na različitu strukturu suradnje što se tiče koautorstva.

Analiza najcitiranijih lokalnih autora (engl. *Most Local Cited Authors*) prikazuje autore čiji su radovi unutar analiziranog skupa 725 radova najviše citirani. Navedena analiza je pružila uvid u utjecaj pojedinih istraživača unutar predmetnog istraživačkog područja te omogućila identifikaciju ključnih autora koji su oblikovali istraživačko područje. Indikator najcitiranijih lokalnih autora je posebno vrijedan jer pruža uvid u lokalnu citiranost, odnosno u to koliko su radovi određenih autora referencirani unutar istog skupa radova. Razlika između lokalnih citata i ukupne citiranosti može biti ključna u razumijevanju unutarnje povezanosti istraživačke zajednice, gdje visoko lokalno citirani autori često predstavljaju temeljne reference ili pionire predmetnog istraživačkog područja. Slikom, u nastavku, prikazana je analiza najcitiranijih lokalnih autora.[34]



Slika 9: Lokalni citati autora

Prema slici, Moore J je najcitiraniji autor unutar analiziranog skupa radova, s 29 lokalnih citata, što sugerira njegov značajan znanstveni utjecaj i učestalost referenciranja njegovih radova od strane drugih istraživača u ovom istom skupu od 725 radova. Autori Li Y (24 citata) i Orlenko A (23 citata) također imaju visoku razinu citiranosti, što ukazuje na njihovu prepoznatljivost i relevantnost unutar predmetnog istraživačkog područja. Nadalje, autori poput Bruss C, Farivar R, Goodsitt J, Hines K, Truong A i Walters A imaju po 21 citat, dok Asselbergs F ima 14 lokalnih citata, što i dalje predstavlja značajan znanstveni doprinos unutar analiziranog predmetnog istraživačkog područja.

Tablicom, u nastavku, prikazuju se rezultati analize produkcije autora kroz vrijeme (engl. *Authors' Production over Time*), pružajući uvid u znanstvenu aktivnost autora u različitim godinama te njihov znanstveni utjecaj mjeren brojem citata. Navedena analiza omogućila je praćenje dinamike objavljivanja radova, njihovog akademskog dosega i utvrđivanje dominantnih istraživačkih trendova povezanih s pojedinim autorima.[34]

Autor	Godina	Frekvencija	Ukupni broj citata	Prosječni broj citata godišnje
WANG Y	2021	3	132	26,4
WANG Y	2022	4	62	15,5
WANG Y	2023	3	33	11
WANG Y	2024	2	2	1
CHEN Y	2020	1	21	3,5
CHEN Y	2021	2	192	38,4
CHEN Y	2024	7	3	1,5
LIU Y	2020	1	85	14,167
LIU Y	2021	1	9	1,8
LIU Y	2022	3	18	4,5

Tablica 7: Analiza produkcije autora kroz vrijeme

Stupci u tablici predstavljaju sljedeće atribute: Autore – ime autora čija se znanstvena produkcija analizira, Godina – godina u kojoj su autorovi radovi objavljeni, Frekvencija (engl. *Freq*) – broj radova koje je autor objavio u određenoj godini, Ukupni broj citata (engl. *Total Citations*, TC) – broj citata koje su ti radovi dobili do trenutka analize, što ukazuje na njihov znanstveni utjecaj i Prosječni broj citata godišnje (engl. *Total Citations per Year*, TCpY) – omjer ukupnog broja citata i proteklog vremenskog razdoblja od objave, koji omogućuje procjenu dinamike citiranosti radova (što je veća vrijednost, to je rad brže i učestalije citiran).[34]

Podatci, u tablici, ukazuju da je Wang Y najproduktivniji autor, s objavama u više godina - 2021., 2022., 2023. i 2024.. Radovi autora Wang Y iz 2021. godine imaju 132 citata, što uzrokuje relativno visok prosječni broj citata godišnje od 26,4, dok su kasniji radovi iz 2022. i 2023. godine manje citirani, ali i dalje održavaju značajan znanstveni utjecaj. Radovi autora Wang Y iz 2024. imaju najmanji broj citata (2 citata, prosječni broj citata godišnje je 1), što je razumljivo jer su najnoviji i još nisu imali dovoljno vremena za akumulaciju citata. Chen Y ima slične rezultate, pri čemu su njegovi radovi iz 2021. godine najcitiraniji (192 citata, prosječni broj citata godišnje je 38,4), dok su radovi iz 2024. još uvijek slabo citirani (prosječni broj citata godišnje je 1,5). Autor Liu Y, iako manje produktivan u odnosu na autore Wang Y i Chen Y, ima rad iz 2020. godine koji je akumulirao 85 citata (prosječni broj citata godišnje je 14,167), što ukazuje na njegov

dugoročni akademski utjecaj. Radovi autora Liu Y iz 2021. i 2022. godine imaju znatno manje citata, što znači da su manje utjecajni.

Rezultati analize produkcije autora kroz vrijeme omogućili su praćenje evolucije znanstvene produkcije pojedinih autora, analizu njihovog utjecaja kroz broj citata i procjenu trajnog akademskog doprinosa u istraživačkom području. Također, omogućili su razlikovanje autora koji imaju kontinuirani znanstveni utjecaj od autora čiji radovi brzo dosegnu vrhunac citiranosti, ali s vremenom gube na relevantnosti.

Tablica, u nastavku, prikazuje produktivnost autora (engl. *Author Productivity*) prema Lotkinom zakonu (engl. *Lotka's Law*), koji opisuje distribuciju znanstvene produkcije među istraživačima. Lotkin zakon tvrdi da većina autora objavljuje samo jedan rad, dok manji broj istraživača objavljuje više radova, pri čemu produktivnost opada eksponencijalno. Ova analiza omogućuje bolje razumijevanje raspodjele istraživačke aktivnosti unutar određenog znanstvenog područja.[34, 37]

Broj napisanih članaka	**Broj autora**	**Proporcija autora**
1	2545	0,854
2	274	0,092
3	72	0,024
4	40	0,013
5	15	0,005
6	12	0,004
7	8	0,003
8	6	0,002
9	3	0,001
10	5	0,002
12	1	0

Tablica 8: Analiza produktivnosti autora prema Lotkinom zakonu

Stupci u tablici predstavljaju sljedeće atribute: Broj napisanih članaka – predstavlja broj radova koje je autor napisao, Broj autora – označava koliko autora je objavilo određeni broj radova i Proporcija autora – predstavlja relativni udio autora koji su objavili određeni broj radova u odnosu na ukupan broj autora.[34, 37]

Podatci, u tablici potvrđuju Lotkin zakon[37], pri čemu se vidi da velika većina autora (85,4%) objavljuje samo jedan rad (2.545 autora). Nadalje, 9,2% autora objavilo je dva rada (274 autora), dok se broj autora rapidno smanjuje kako raste broj objavljenih

radova. Samo 72 autora (2,4%) objavila su tri rada, a još manji broj autora (manje od 1%) ima pet ili više objavljenih radova. Najproduktivniji autor Wang Y u analiziranom skupu radova objavio je 12 radova, što dodatno potvrđuje karakterističnu eksponencijalnu raspodjelu znanstvene produktivnosti.

Tablica, u nastavku, prikazuje rezultate analize lokalnog utjecaja autora (engl. *Authors' Local Impact*) unutar skupa predmetnog istraživanog područja znanstvenih publikacija. Navedena analiza omogućila je identifikaciju najutjecajnijih istraživača na temelju različitih bibliometrijskih indikatora koji procjenjuju produktivnost, utjecaj i konzistentnost znanstvene produkcije autora. [34]

Autor	**H-indeks**	**G-indeks**	**M-indeks**	**Ukupni broj citata**	**Broj publikacija**	**Prva godina publikacije**
MOORE J	7	10	0,875	692	10	2018
WANG Y	7	12	1,4	229	12	2021
LI Y	6	8	1,2	72	9	2021
WANG C	6	8	0,857	121	8	2019
ZHANG Y	6	7	1,2	61	7	2021
LI H	5	8	0,714	64	9	2019
LI S	5	8	1,25	180	8	2022
LIU H	5	7	1	143	7	2021
LIU Z	5	7	0,833	83	7	2020
TSAMARDINOS I	5	8	0,833	139	8	2020

Tablica 9: Analiza lokalnog znanstvenog utjecaja autora

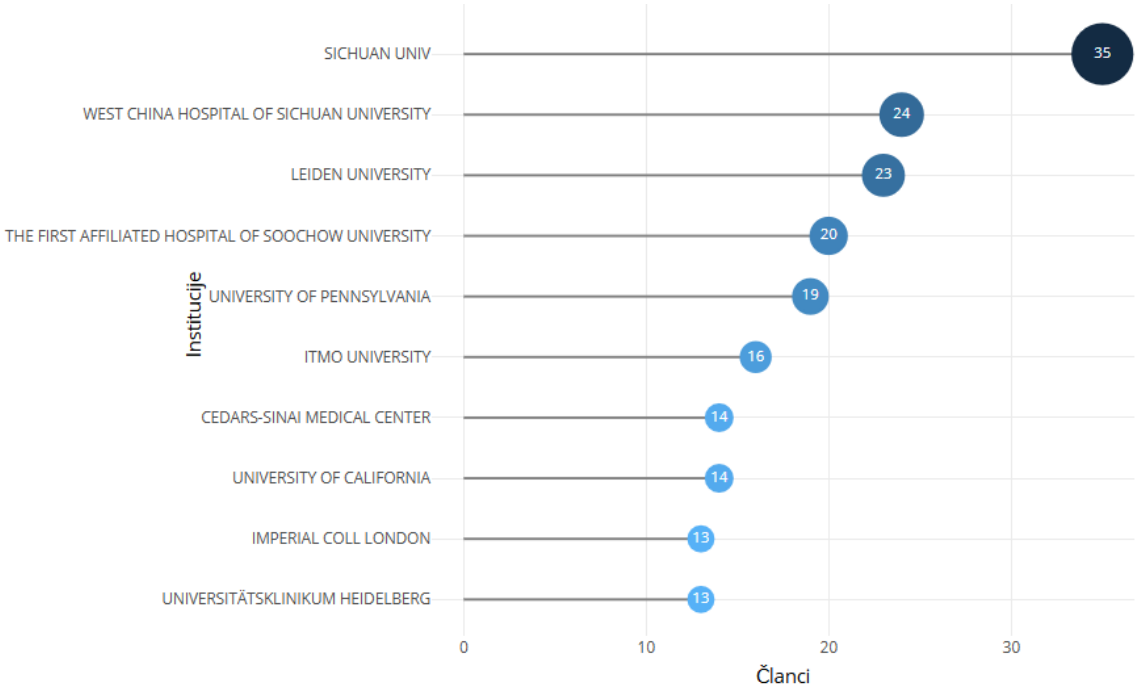
Stupci u tablici predstavljaju sljedeće atribute: Autor – ime istraživača čiji se znanstveni utjecaj analizira, H-indeks (engl. *h_index*) – mjeri produktivnost i utjecaj autora na temelju broja radova i njihovih citata. Autor s H-indeksom *n* ima najmanje n radova koji su citirani najmanje *n* puta. Viši H-indeks označava veći znanstveni utjecaj. Nadalje, G-indeks (engl. *g_index*) – poboljšana verzija H-indeksa koja uzima u obzir distribuciju citata. G-indeks *n* znači da su *n* najcitiranijih radova zajedno dobili najmanje n^2 citata. Viši G-indeks sugerira da autor ima nekoliko vrlo utjecajnih radova. Zatim, M-indeks (engl. *m_index*) – odnosi se na prosječnu godišnju stopu rasta H-indeksa, računajući se kao H-indeks podijeljen s brojem godina od prve godine publikacije (PY_start). Veće vrijednosti ukazuju na brži znanstveni rast i kontinuiranu produktivnost.

Ukupni broj citata (engl. *Total Citations*, TC) – ukupan broj citata koje su autorovi radovi primili unutar analiziranog skupa publikacija, što je direktan indikator njihovog znanstvenog utjecaja, Broj publikacija (engl. *Number of Publications*, NP) – ukupan broj objavljenih radova autora unutar analiziranog skupa i prva godina publikacije (PY_start) – prva godina u kojoj je autor objavio rad unutar analiziranog skupa podataka.[34]

Prema rezultatima iz, prethodne, tablice analiza lokalnog znanstvenog utjecaja autora, Autor Moore J ima najviši ukupni broj citata (692) i najveći H-indeks (7), što ukazuje na njegov značajan dugoročni utjecaj u analiziranom predmetnom znanstvenom području. Također, njegov G-indeks (10) pokazuje da je među njegovim radovima nekoliko visokocitiranih radova, dok M-indeks (0,875) sugerira stalnu i konzistentnu znanstvenu produktivnost od 2018. godine. Slične rezultate imaju autori Wang Y (H-indeks je 7, Ukupni broj citata je 229, Broj publikacija je 12) i Li Y (H-indeks je 6, Ukupni broj citata je 72, Ukupni broj citata je 9) te su također visoko produktivni autori, ali s manjim ukupnim brojem citata, što ukazuje na noviji doprinos znanstvenoj zajednici. Autor Wang Y ima relativno visok M-indeks (1,4), što sugerira brzu akumulaciju znanstvenog utjecaja kroz vrijeme. Autori Liu H i Liu Z imaju niže H-indekse (5), ali značajan broj citata (143 i 139, respektivno), što upućuje na prisustvo nekoliko radova s iznimno visokom citiranošću, no manji broj ukupnih publikacija. Autor Tsamardinos I ima sličan profil s visokim M-indeksom (0,833), što ukazuje na konzistentan znanstveni rast.

S analizom lokalnog znanstvenog utjecaja autora utvrdili su se ključni istraživači u predmetnom istraživačkom području te identificirali najcitiraniji znanstveni radovi ovog istraživačkog područja.

U nastavku su, prikazane slikom, najrelevantnije institucije (engl. *Most Relevant Affiliations*). Analiza najrelevantnijih institucija omogućila je identifikaciju akademskih i istraživačkih ustanova koje imaju najveći doprinos znanstvenoj produkciji unutar analiziranog skupa radova. Navedena analiza daje uvid u geografske i institucionalne centre istraživačkih aktivnosti te ističe vodeće institucije u znanstvenom području predmetnog istraživanja.[34]



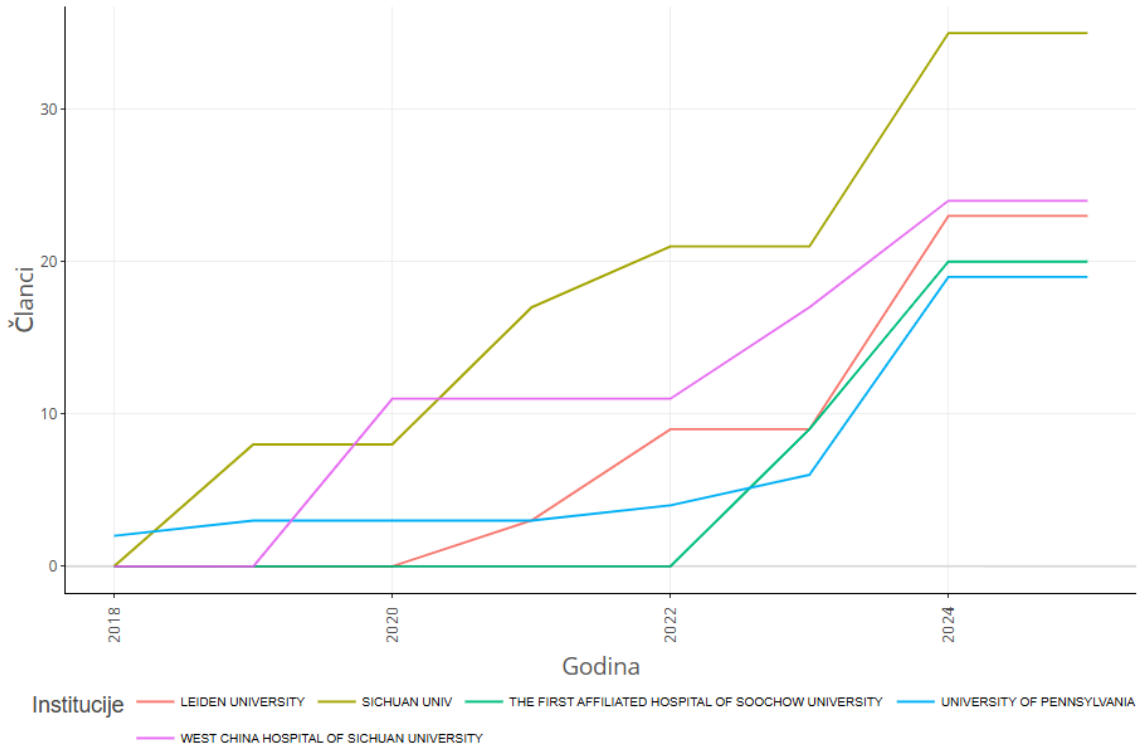
Slika 10: Najrelevantnije institucije

Na temelju rezultata iz prethodne slike, *Sichuan University* prednjači s ukupno 35 objavljenih radova, što ukazuje na njegovu dominantnu ulogu u analiziranom skupu radova. Slijedi *West China Hospital of Sichuan University* s 24 objavljena rada, što sugerira da je ova institucija ključna za medicinska i interdisciplinarna istraživanja povezana s temom istraživačkog područja. Nadalje, *Leiden University* (23 rada), *The First Affiliated Hospital of Soochow University* (20 radova) i *University of Pennsylvania* (19 radova) imaju značajan znanstveni doprinos, potvrđujući svoju relevantnost u istraživačkom prostoru predmetnog istraživačkog područja.

Među ostalim visoko rangiranim institucijama nalaze se *ITMO University* (16 radova), *Cedars-Sinai Medical Center* (14 radova), *University of California* (14 radova), *Imperial College London* (13 radova) i *Universitätsklinikum Heidelberg* (13 radova). Prethodno navedene institucije predstavljaju aktere u razvoju istraživanja predmetnog područja, gdje se može primijetiti kombinacija tehničkih sveučilišta, medicinskih centara i multidisciplinarnih istraživačkih ustanova.

Zatim slijedi, na slici u nastavku, analiza rezultata produkcije radova institucija kroz vrijeme (engl. *Affiliations' Production over Time*), omogućujući identifikaciju znanstvenih centara s najintenzivnijom istraživačkom aktivnošću u različitim godinama. Navedena analiza pokazala se ključnom za razumijevanje vremenske dinamike

istraživanja, prepoznavanje vodećih institucija u znanstvenom području predmetnog istraživanja te praćenje razvoja institucionalnog doprinosa znanstvenoj produkciji područja istraživanja ovog rada. [34]



Slika 11: Analiza produkcije radova institucija kroz vrijeme

Prema prikazanoj slici, *Sichuan University* se ističe, u odnosu na ostale institucije, znanstvenom produkcijom, s 35 objavljenih radova u 2024. i 2025. godini, dok je u prethodnim godinama (2022. i 2023.) imao manji, ali značajan broj publikacija (21 rad). Ovaj trend sugerira stalni rast istraživačke aktivnosti i povećanje doprinosa institucije *Sichuan University* znanstvenom području predmetnog istraživanja.

Nadalje, *West China Hospital of Sichuan University* ima 24 objavljena rada u 2024. i 2025., što ukazuje na kontinuirani doprinos medicinskih istraživanja u ovom kontekstu predmetnog istraživanja. Također, *Leiden University* održava stabilan broj publikacija (23 rada u 2024. i 2025.), što sugerira konzistentan istraživački doprinos i održivu akademsku produktivnost u području predmetnog istraživanja. Institucija *The First Affiliated Hospital of Soochow University* također pokazuje stabilnu produkciju od 20 radova u 2024. i 2025., što upućuje na njegovu važnu ulogu u istraživanjima vezanim s medicinom i zdravstvenim znanosti te područjem predmetnog istraživanja.

Tablicom, u nastavku, prikazani su rezultati analize zemalja korespondirajućih autora (engl. *Corresponding Author's Countries*) odnosno glavnih autora za komunikaciju tijekom objave rada, a s analizom je omogućena identifikacija geografskog porijekla vodećih istraživača u objavljenim radovima. Navedena analiza daje uvid u dominantne istraživačke centre, razinu međunarodne suradnje i distribuciju znanstvene produktivnosti predmetnog istraživanja među zemljama. [34]

Zemlja	**Broj radova**	**Postotak radova**	**Radovi iz jedne zemlje (SCP)**	**Radovi s međunarodnom suradnjom (MCP)**	**Postotak radova s međunarodnom suradnjom (MCP %)**
Sjedinjene Američke Države	128	17,7	107	21	16,4
Kina	123	17	103	20	16,3
Njemačka	58	8	52	6	10,3
Indija	36	5	33	3	8,3
Brazil	22	3	18	4	18,2
Kanada	21	2,9	17	4	19
Grčka	21	2,9	18	3	14,3
Ujedinjeno Kraljevstvo	18	2,5	9	9	50
Koreja	17	2,3	16	1	5,9
Nizozemska	14	1,9	13	1	7,1

Tablica 10: Analiza zemalja korespondirajućih autora

Stupci u tablici predstavljaju sljedeće atribute: Zemlja (engl. *Country*) – zemlja kojoj pripada korespondirajući autor (glavni autor za komunikaciju tijekom objave rada), Broj radova (engl. *Articles*) – ukupan broj znanstvenih publikacija u kojima je korespondirajući autor iz navedene zemlje, Postotak radova (engl. *Articles %*) – udio radova iz određene zemlje u odnosu na ukupan broj analiziranih publikacija, izražen u postotcima, Radovi iz jedne zemlje (engl. *Single Country Publications*, SCP) – broj radova u kojima su svi autori iz iste zemlje, bez međunarodne suradnje, Radovi s međunarodnom suradnjom (engl. *Multiple Country Publications*, MCP) – broj radova u kojima su autori iz više zemalja, što ukazuje na međunarodnu znanstvenu suradnju, Postotak radova s međunarodnom suradnjom (engl. *Multiple Country Publications*, MCP

%) – udio radova s međunarodnom suradnjom u odnosu na ukupan broj radova iz određene zemlje.[34]

Rezultati iz prethodne tablice, pokazuju da Sjedinjene Američke Države imaju najveći broj radova s korespondirajućim autorima (128 radova, 17,7% svih publikacija), pri čemu je većina tih radova nacionalna (SCP je 107), dok je 21 rad nastao kroz međunarodnu suradnju (MCP % je 16,4%). Kina (123 radova, 17%) također ima visok broj korespondirajućih autora, s dominantnim udjelom nacionalnih istraživanja (SCP je 103, MCP je 20, MCP % je 16,3%), što ukazuje na relativno nisku razinu međunarodne suradnje u usporedbi s nekim drugim zemljama. S druge strane, Ujedinjeno Kraljevstvo pokazuje iznimno visok MCP % (50%), što znači da je polovica radova u kojima su britanski istraživači korespondirajući autori nastala kroz međunarodnu suradnju, što ukazuje na otvorenost prema međunarodnoj suradnji. Brazil (MCP % je 18,2%), Grčka (MCP % je 14,3%) i Kanada (MCP % je 19%) također pokazuju relativno visoku razinu međunarodne suradnje, dok Njemačka (MCP % je 10,3%) i Indija (MCP % je 8,3%) imaju značajan broj radova, ali uz nisku stopu međunarodne suradnje.

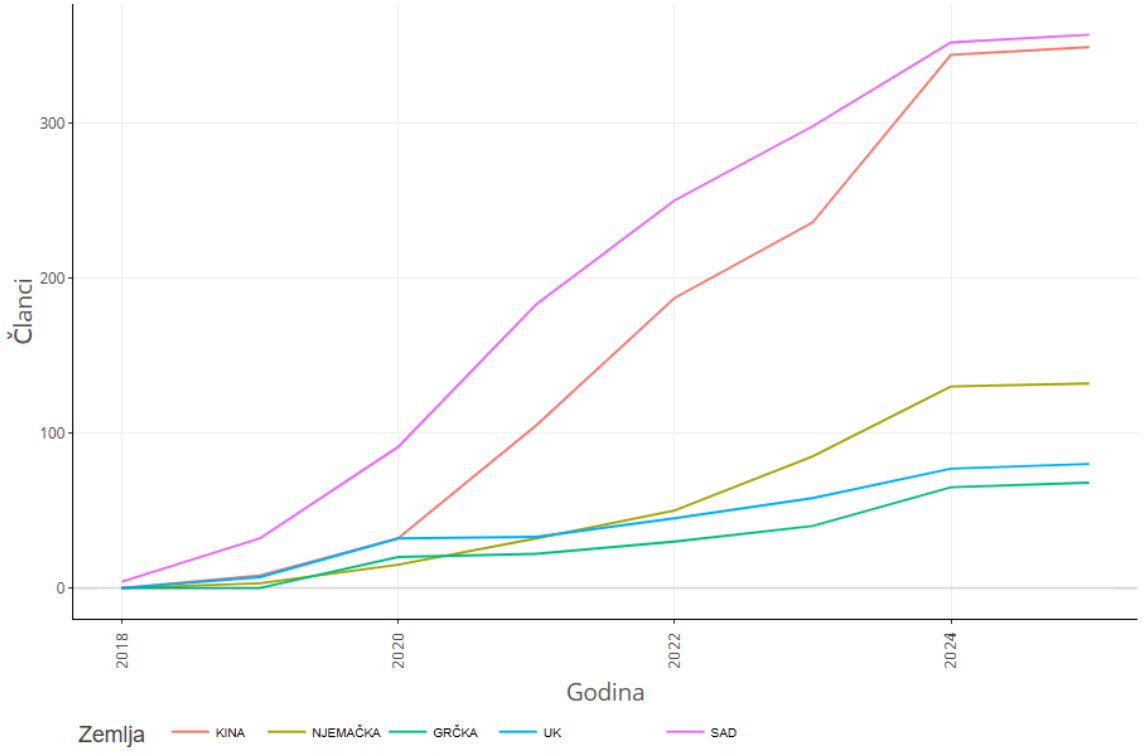
Tablicom, u nastavku, prikazani su rezultati analize znanstvene produkcije radova po zemljama (engl. *Countries' Scientific Production*). Analizom se omogućilo uvid u geografsku raspodjelu istraživačkih aktivnosti unutar analiziranog znanstvenog područja predmetnog istraživanja. Navedena analiza identificira zemlje s najvišom znanstvenom produkcijom i služi za razumijevanje dominantnih istraživačkih centara na globalnoj razini.[34]

Zemlja	**Broj radova**
Sjedinjene Američke Države	357
Kina	353
Njemačka	133
Ujedinjeno Kraljevstvo	80
Grčka	68
Brazil	66
Indija	58
Kanada	56
Italija	40
Španjolska	40

Tablica 11: Analiza znanstvene produkcije po zemljama

Rezultati iz tablice ukazuju da su Sjedinjene Američke Države i Kina vodeće zemlje po broju objavljenih znanstvenih radova, s 357 i 353 publikacije, respektivno. Gotovo izjednačena distribucija, između prethodno navedene dvije zemlje, potvrđuje njihovu vodeću ulogu u globalnoj znanstvenoj produkciji te sugerira na intenzivna istraživanja u analiziranom predmetnom području ovog istraživanja. Zemlje poput Njemačke (133 radova), Ujedinjenog Kraljevstva (80 radova) i Grčke (68 radova) također pokazuju značajnu istraživačku aktivnost. Brazil (66 radova), Indija (58 radova), Kanada (56 radova), Italija (40 radova) i Španjolska (40 radova) također ostvaruju relevantan znanstveni doprinos, pri čemu se vidi široka međunarodna zastupljenost istraživača iz različitih dijelova svijeta. Prikazana distribucija broja radova, iz prethodne tablice, upućuje na globalni karakter znanstvenih istraživanja, gdje uz vodeće zemlje poput SAD-a i Kine, značajnu ulogu imaju i europske i južnoameričke države.

Slijedi slika, u nastavku, koja pruža uvid u dinamiku znanstvene produkcije po zemljama tijekom analiziranog vremenskog razdoblja od 2018. do 2025. godine. Navedena analiza omogućila je identifikaciju trendova rasta u znanstvenoj produkciji te usporedbu tempa rasta znanstvene produkcije među različitim zemljama.



Slika 12: Znanstvena produkcija po zemljama kroz vrijeme

Na temelju prethodne slike, vidljivo je da Sjedinjene Američke Države (SAD) i Kina prednjače u znanstvenoj produkciji, s kontinuiranim i značajnim rastom broja publikacija. Sjedinjene Američke Države (SAD) su započele s manjim brojem publikacija u 2018. godini (4 rada), ali je od 2020. godine primjetan značajan porast, dosežući vrhunac od 357 radova u 2025. Sličan trend bilježi i Kina, čija se znanstvena produkcija povećava iz godine u godinu, od 32 rada u 2020. do 349 radova u 2025., pri čemu je rast posebno izražen nakon 2021. godine. Njemačka, Ujedinjeno Kraljevstvo (UK) i Grčka također pokazuju pozitivan trend rasta, ali s nižom stopom u usporedbi s vodećim zemljama. Njemačka je doživjela kontinuiran porast, od 15 radova u 2020. do 132 u 2025., dok Ujedinjeno Kraljevstvo (UK) raste od 7 publikacija u 2019. do 80 u 2025. Grčka, iako s manjim apsolutnim brojevima, bilježi stabilan rast znanstvene produkcije, od 20 radova u 2020. do 68 u 2025.

Prethodnom slikom je jasno vidljiva dominacija Kine i Sjedinjenih Američkih Država (SAD), s oštrim uzlaznim trendovima u njihovoj znanstvenoj produkciji. Njemačka također pokazuje progresivan rast, dok su Ujedinjeno Kraljevstvo (UK) i Grčka u stabilnom, ali nešto sporijem porastu.

Zatim, tablicom, u nastavku, prikazani su rezultati analize najcitiranijih zemalja (engl. *Most Cited Countries*), omogućivši uvid u ukupni broj citata (engl. *Total Citations*, TC) i prosječan broj citata po radu (engl. *Average Article Citations*) za svaku zemlju. Navedena analiza pružila je dublje razumijevanje utjecaja znanstvene produkcije na globalnoj razini te je pomogla u identifikaciji istraživačkih zemalja s najvišom akademskom vidljivošću.[34]

Zemlja	Ukupni broj citata (TC)	Prosječan broj citata po radu
Sjedinjene Američke Države	3268	25,50
Kina	1372	11,20
Hong Kong	1028	342,70
Njemačka	907	15,60
Indija	522	14,50
Australija	324	27,00
Kanada	250	11,90
Ujedinjeno Kraljevstvo	249	13,80
Grčka	236	11,20
Nizozemska	177	12,60

Tablica 12: Analiza najcitiranijih zemalja

Prema prikazanim rezultatima u tablici, Sjedinjene Američke Države imaju najviši ukupni broj citata (TC je 3268), što potvrđuje njihov dominantan znanstveni utjecaj. Također, prosječan broj citata po radu (25,50) ukazuje na visoku kvalitetu i prepoznatljivost istraživanja koja potječu iz Sjedinjenih Američkih Država. Kina (TC je 1372) također ima visok broj ukupnih citata, ali s nešto nižim prosječnim brojem citata po radu (11,20), što može ukazivati na veliku produkciju radova, ali s nešto manjim pojedinačnim utjecajem. Među rezultatima, ističe se Hong Kong, koji pokazuje iznimno visok prosječan broj citata po radu (342,70), iako s manjim ukupnim brojem citata (1028). Ova anomalija s Hong Kong-om ukazuje na nekoliko vrlo visoko citiranih radova, koji značajno podižu prosjek citiranosti. Zemlje poput Njemačke (TC je 907, prosjek 15,60), Indije (TC je 522, prosjek 14,50) i Australije (TC je 324, prosjek 27,00) također imaju solidan znanstveni utjecaj, pri čemu se Australija ističe relativno visokim prosječnim brojem citata po radu, što ukazuje na usmjerenost na visokokvalitetna istraživanja. Kanada (TC je 250, prosjek 11,90), Ujedinjeno Kraljevstvo (TC je 249, prosjek 13,80), Grčka (TC je 236, prosjek 11,20) i Nizozemska (TC je 177, prosjek 12,60) također sudjeluju u globalnoj znanstvenoj produkciji, ali s nešto nižim ukupnim brojem citata. Međutim, njihov prosječan broj citata po radu ukazuje na stabilan znanstveni doprinos.

Rezultati, iz prethodne tablice, su dali procjenu znanstvenog utjecaja pojedinih zemalja, ne samo na temelju ukupnog broja citata, već i kroz prosječnu citiranost po radu, što je pokazatelj kvalitete i vidljivosti znanstvene produkcije. Sjedinjene Američke

Države i Kina dominiraju ukupnom citiranošću, dok Hong Kong, Australija i Njemačka pokazuju značajan utjecaj kroz prosječnu citiranost pojedinačnih radova.

U sljedećoj analizi, kroz tablicu u nastavku, prikazuju se rezultati analize najglobalnije citiranih dokumenata (engl. *Most Global Cited Documents*), omogućivši identifikaciju znanstvenih radova s najvećim utjecajem u analiziranom predmetnom znanstvenom istraživačkom području. Navedeni rezultati služe za razumijevanje temeljnih referenci unutar istraživanog područja, procjenu znanstvene vidljivosti pojedinih radova te analizu trendova u citiranosti. [34]

Rad (Prvi autor, Godina, Izvor)	**Jedinstveni identifikator rada (DOI)**	**Ukupan broj citata (TC)**	**Citati po godini (TC per Year)**	**Normalizirani broj citata**
HE X, 2021, KNOWL BASED SYST	10.1016/j.knosys.2020.106622	1022	204,40	32,26
ALAA A, 2019, PLOS ONE	10.1371/journal.pone.0213653	349	49,86	7,23
OLSON R, 2019, AUTOMATED MACHINE LEARNING: METHODS, SYSTEMS, CHALLENGES	10.1007/978-3-030-05318-5_8	336	48,00	6,96
JAVAID M, 2022, INT J INTELL NETW	10.1016/j.ijin.2022.05.002	328	82,00	21,43
LE T, 2020, BIOINFORMATICS	10.1093/bioinformatics/btz470	246	41,00	10,67
PALEYES A, 2023, ACM COMPUT SURV	10.1145/3533378	226	75,33	27,06
DUNN A, 2020, NPJ COMPUT MATER	10.1038/s41524-020-00406-3	207	34,50	8,98
CHEN Z, 2021, NUCLEIC ACIDS RES	10.1093/nar/gkab122	172	34,40	5,43

ZOELLER M, 2021, J ARTIF INTELL RES	Nema	168	33,60	5,30
TURNER R, 2020, PR MACH LEARN RES	Nema	146	24,33	6,33

Tablica 13: Analiza najglobalnije citiranih dokumenata

Stupci u tablici predstavljaju sljedeće atribute: Rad – identificiran po prvom autoru, godini objave i izvoru objave. Zatim, Jedinstveni identifikator rada (engl. *Digital Object Identifier* , DOI) – jedinstveni identifikator znanstvenog rada koji omogućuje trajnu poveznicu na izvornu publikaciju. Neki radovi nemaju DOI (označeno kao Nema). Nadalje, Ukupan broj citata (engl. *Total Citations*, TC) – broj puta koliko je određeni rad citiran u drugim znanstvenim publikacijama. Zatim, Citati po godini (engl. *Total Citations per Year,* TC per Year) – prosječan broj citata koje rad prima svake godine od objave. Pokazatelj omogućava procjenu dinamike citiranosti – radovi s visokim *TC per Year* brzo stječu znanstvenu vidljivost i postaju referentni u polju. Na kraju, Normalizirani broj citata (engl. *Normalized Total Citations*) – omjer ukupnog broja citata rada u odnosu na prosječnu citiranost radova u istom području. Navedena metrika omogućuje usporedbu utjecaja radova neovisno o njihovom vremenu objave, čime se izbjegava pristranost prema starijim, dulje citiranim radovima.[34]

Iz rezultata prikazane, prethodne, tablice, najcitiraniji rad u analiziranom skupu radova je *He X, 2021, KNOWLEDGE-BASED SYSTEMS*, s 1.022 citata i prosječnom godišnjom citiranošću od 204,40, što ga čini iznimno utjecajnim unutar znanstvene zajednice. Također, ima najviši normalizirani broj citata (32,26), što znači da je njegova citiranost znatno iznad prosjeka u polju. Drugi radovi, poput *Alaa A, 2019, PLOS ONE* (349 citata, Citati po godini su 49,86, Normalizirani broj citata je 7,23) i *Olson R, 2019, AUTOMATED MACHINE LEARNING METHODS* (336 citata, Citati po godini su 48,00, Normalizirani broj citata je 6,96), također imaju visok znanstveni utjecaj, ali s nižim normaliziranim brojem citata u odnosu na vodeći rad.

Nadalje, rad *Dunn A, 2020, NPJ COMPUT MATER* (207 citata, Citati po godini su 34,50, normalizirani broj citata je 8,98) pokazuje višu normaliziranu vrijednost (8,98), što sugerira veći relativni utjecaj u svom području u odnosu na prosječne publikacije. Također, radovi *Javaid M, 2022, INT J INTELL NETW* (328 citata Citati po godini su

82,00, Normalizirani broj citata je 21,43) *i Paleyes A, 2023, ACM COMPUT SURV* (226 citata, Citati po godini su 75,33 Normalizirani broj citata je 27,06) ističu se visokom prosječnom godišnjom citiranošću, što znači da su brzo postali relevantni unutar istraživačke zajednice.

Radovi bez jedinstveni identifikator rada (DOI) - *ZOELLER M, 2021, J ARTIF INTELL RES* i *TURNER R, 2020, PR MACH LEARN RES*, također su značajni, no njihova vidljivost i dostupnost možda je ograničena zbog nepostojanja jedinstvenog identifikatora rada (DOI-a), što možda utječe na njihovu, smanjenu, citiranost.

Sljedećom tablicom prikazani su rezultati analize najcitiranijih lokalnih dokumenata (engl. *Most Local Cited Documents*) u ovom skupu od 725 radova. Navedena analiza omogućila je prepoznavanje znanstvenih radova koji imaju najveći utjecaj unutar analiziranih 725 radova u ovom poglavlju (lokalna citiranost) u usporedbi s njihovim globalnim utjecajem.[34]

Stupci u tablici predstavljaju sljedeće atribute: Rad – identificiran po prvom autoru, godini objave i izvoru objave. Zatim, Jedinstveni identifikator rada (engl. *Digital Object Identifier* , DOI) – jedinstveni identifikator znanstvenog rada koji omogućuje trajnu poveznicu na izvornu publikaciju. Neki radovi nemaju DOI (označeno kao Nema). Nadalje, Godina (engl. *Year*) – godina objave rada, predstavlja bitan parametar za razumijevanje dinamike citiranosti kroz vrijeme. Zatim, Lokalni citati (engl. *Local Citations*) – ukupan broj citiranosti u ovom skupu od 725 radova. Nadalje, Globalni citati (engl. *Global Citations*) – ukupan broj citata koje je rad primio u svim znanstvenim publikacijama izvan lokalnog skupa odnosno izvan ovog skupa od 725 radova. Pokazatelj globalnih citata ukazuje na širi znanstveni utjecaj rada. Nadalje, Omjer lokalnih i globalnih citata u postotku (LC/GC Ratio %) – postotak koji pokazuje koliko su citati rada koncentrirani unutar ovog skupa od 725 radova. Visok omjer ukazuje na to da je rad iznimno važan u predmetnom istraživačkom polju, dok nizak omjer sugerira širi globalni utjecaj. Normalizirani lokalni citati (engl. *Normalized Local Citations*) – normalizirani broj citata rada unutar ovog skupa od 725 radova, omogućuje usporedbu između radova objavljenih u različitim godinama i istraživačkim poljima. Normalizirani globalni citati (engl. *Normalized Global Citations*) – normalizirani broj globalnih citata koji omogućuje usporedbu između različitih radova na temelju njihovog šireg akademskog dosega.[34]

Rad (Prvi autor, Godina, Izvor)	**Jedinstveni identifikator rada (DOI)**	**Godina**	**Lokalni citati**	**Globalni citati**	**LC/GC Omjer (%)**	**Normalizirani lokalni citati**	**Normalizirani globalni citati**
TRUONG A, 2019, PROC INT C TOOLS ART	10.1109/ICTAI.2019.00209	2019	21	120	17,50	20,30	2,49
ORLENKO A, 2020, BIOINFORMATICS	10.1093/bioinformatics/btz796	2020	14	35	40,00	22,00	1,52
LI Y, 2021, PROC VLDB ENDOW	10.14778/3476249.3476270	2021	12	13	92,31	37,14	0,41
ORLENKO A, 2018, BIOCOMPUT-PACSYM	Nema	2018	9	28	32,14	1,00	1,00
DAFFLON J, 2020, HUM BRAIN MAPP	10.1002/hbm.25028	2020	9	29	31,03	14,14	1,26

YAKOVLEV A, 2020, PROC VLDB ENDOW	10.14778/3415478.3415542	2020	9	41	21,95	14,14	1,78
MANDUCHI E, 2020, BMC BIOINFORMATICS	10.1186/s12859-020-03755-4	2020	6	12	50,00	9,43	0,52
ESTEVEZ-VELARDES, 2019,	Nema	2019	5	10	50,00	4,83	0,21
TURNER R, 2020, PR MACH LEARN RES	Nema	2020	5	146	3,42	7,86	6,33
IKEMURA K, 2021, J MED INTERNET RES	10.2196/23458	2021	5	51	9,80	15,48	1,61

Tablica 14: Prikaz analize najcitiranijih lokalnih dokumenata

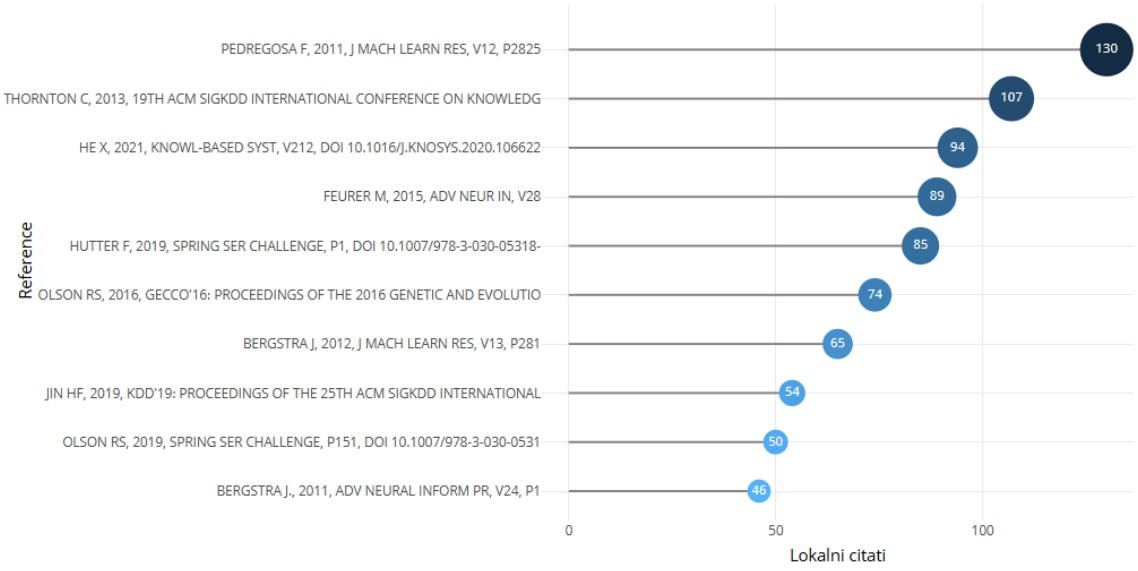
Prema prikazanim podatcima iz prethodne tablice, rad *Truong A, 2019, PROC INT C TOOLS ART* ima najveći broj lokalnih citata (21) i globalnih citata (120), s LC/GC omjerom od 17,50%, što ukazuje na značajnu prisutnost u globalnom akademskom korpusu, ali i značajan utjecaj unutar analiziranog skupa podataka. Navedeni rad, također,

ima najvišu normaliziranu lokalnu citiranost (20,30) i normaliziranu globalnu citiranost (2,49), što znači da je među najutjecajnijima u lokalnom i globalnom kontekstu.

Rad *Li Y, 2021, PROC VLDB ENDOW* ima veći omjer lokalnih u odnosu na globalne citate (92,31%), što sugerira iznimnu važnost unutar ovog skupa od 725 radova, ali ograničen globalni doseg (samo 13 globalnih citata). Slično tome, *rad Estevez-Velarde S, 2019* ima 50% LC/GC omjer, što znači da je pola njegovih citata došlo iz ovog skupa od 725 radova, što ukazuje na specijalizirani utjecaj unutar predmetnog istraživačkog područja. S druge strane, rad *Turner R, 2020, PR MACH LEARN RES* ima vrlo visoku globalnu citiranost (146 citata), ali nisku lokalnu citiranost (5 citata), s niskim LC/GC omjerom (3,42%). Prethodno navedeno sugerira da je rad izuzetno utjecajan na globalnoj razini, ali nije toliko često citiran unutar ovog skupa od 725 radova.

Analiza najcitiranijih lokalnih dokumenata omogućila je razumijevanje znanstvene vidljivosti pojedinih radova u lokalnom i globalnom kontekstu. Radovi s visokim LC/GC omjerom su bitni u predmetnom istraživačkom području ovog rada, dok su radovi s visokim globalnim citiranjem široko prepoznati unutar globalne znanstvene zajednice. Analiza najcitiranijih lokalnih dokumenata služila je za identifikaciju ključnih referenci za daljnje istraživanje.

Slijedom navedenog, u nastavku se analiziraju reference. Slika, u nastavku, prikazuje rezultate analize najcitiranijih lokalnih referenci (engl. *Most Local Cited References)* u analizi predmetnog istraživačkog područja, time omogućivši identifikaciju ključnih izvora koji su najčešće citirani unutar ovog skupa od 725 znanstvenih radova. Navedena analiza pruža uvid u fundamentalne radove koji oblikuju istraživačko područje te ukazuje na najutjecajnije znanstvene publikacije unutar određene zajednice istraživača.

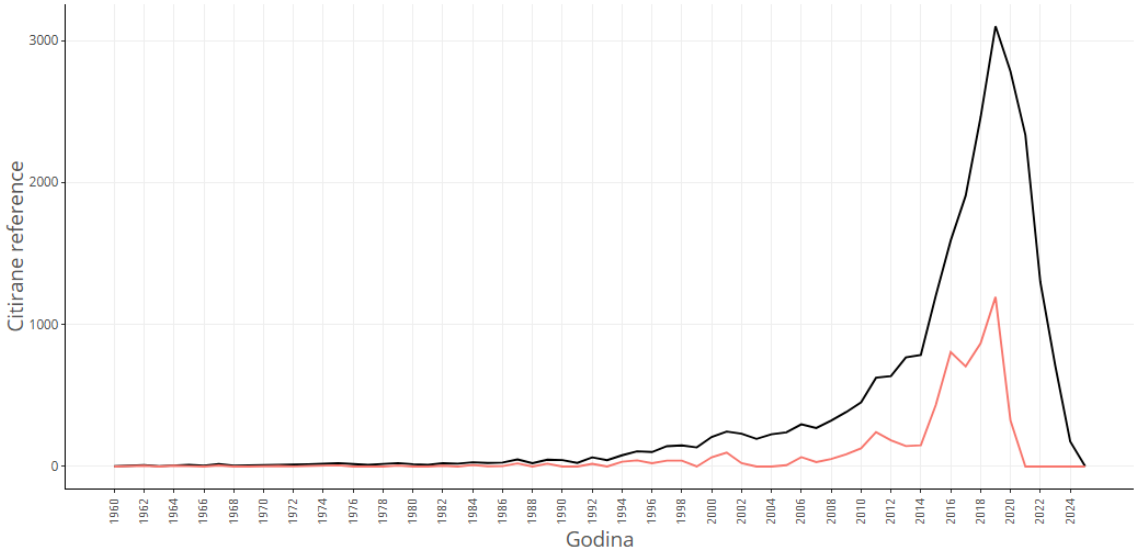


Slika 13: Najcitiranije lokalne reference

Prema prikazanoj slici, najcitiranija lokalna referenca je rad *Pedregosa F, 2011, J Mach Learn Res, V12, P2825*, koji je citiran 130 puta unutar ovog skupa od 725 znanstvenih radova, što upućuje na njegov središnji značaj u predmetnoj istraživačkoj domeni. Navedeni rad Pedregose i autora je *Scikit-learn: Machine Learning in Python[38]*, a *Scikit-learn* predstavlja temeljni metodološki i tehnički doprinos strojnom učenju, stoga je ključna referenca za veliki broj radova unutar ovog skupa od 725 znanstvenih radova, pri čemu predstavlja ključnu referencu i ovog rada.

Slijedi rad *Thornton C, 2013, 19TH ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING*, koji je citiran 107 puta, što potvrđuje njegov visok akademski utjecaj u području strojnog učenja. Također, rad *He X, 2021, KNOWLEDGE-BASED SYSTEMS*, koji je citiran 94 puta, ukazuje na nedavno objavljeni rad koji brzo stječe znanstvenu prepoznatljivost. Radovi *Feurer M, 2015, ADV NEUR IN* (89 citata) i *Hutter F, 2019, SPRING SER CHALLENGE* (85 citata) također su među najosnovnijim referencama u ovom istraživačkom području, što sugerira na njihovu ulogu u oblikovanju istraživanja unutar ovog analiziranog skupa od 725 znanstvenih radova. Ostale visoko citirane reference su *Olson RS, 2016, GECCO'16* (74 citata), *Bergstra J, 2012, J Mach Learn Res* (65 citata) te *Bergstra J, 2011, ADV NEURAL INFORM PR* (46 citata), pokazuju dugoročan utjecaj na istraživanja u području strojnog učenja, modeliranja i optimizacije modela.

U nastavku slijedi spektroskopija referenci (engl. *Reference Spectroscopy*), odnosi se na analizu godina objave referenci citiranih unutar skupa analiziranih 725 znanstvenih radova. Navedena metoda omogućuje razumijevanje vremenske distribucije citiranih izvora, prepoznavanje dominantnih perioda akademskog utjecaja i identifikaciju razvoja istraživačkih trendova kroz godine.[34]

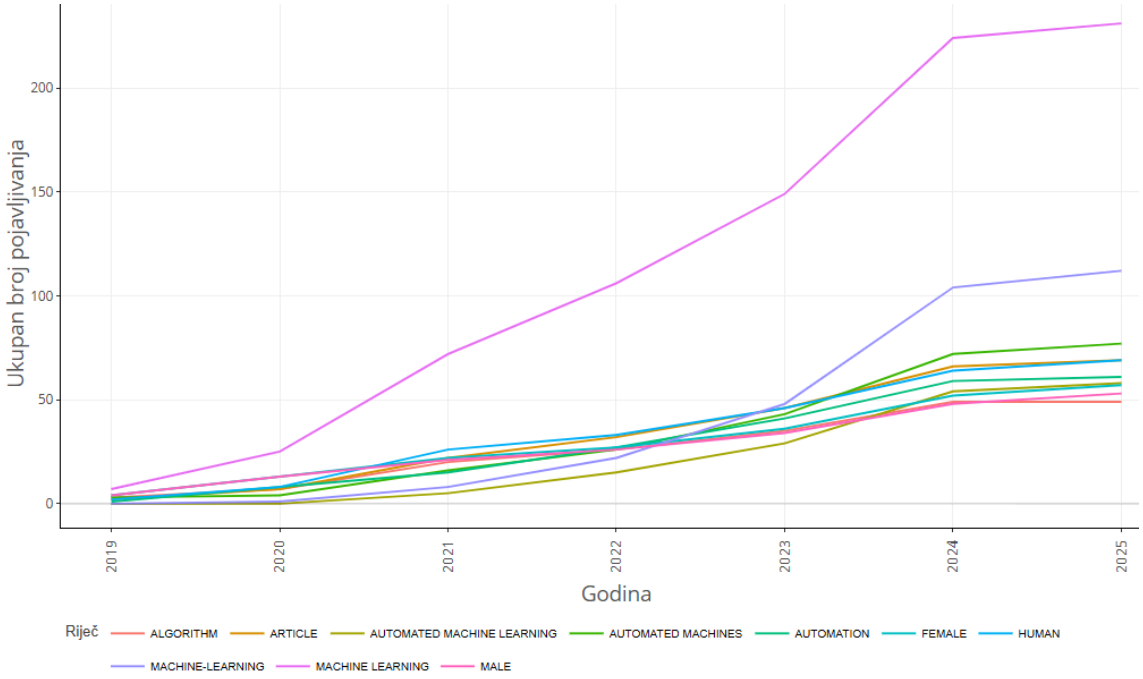


Slika 14: Spektroskopija referenci

Prethodna slika prikazuje broj citiranih referenci po godini objave (crna linija) i odstupanje od petogodišnjeg medijana citata (crvena linija). Prema slici, dominantni period znanstvene citiranosti započinje oko 1990. godine. Crna linija na slici (ukupni broj citiranih referenci) prikazuje eksponencijalni rast citiranosti nakon 1990-ih, dok crvena linija (odstupanje od petogodišnjeg medijana) pokazuje razdoblja kada su određene godine bile posebno citirane u odnosu na prethodnih pet godina, dok nagli rast citiranosti započinje oko 2005. godine. Najveći broj citiranih referenci pripada periodu između 2015. i 2020. godine, dok je vrhunac je postignut 2019. godine (3.101 citirana referenca), nakon čega slijedi blagi pad 2020. godine (2.783 citata). Ova razdoblja imaju pozitivna odstupanja od petogodišnjeg i ukupnog medijana, što ukazuje na visoku akademsku relevantnost radova iz tog perioda. Nakon 2021. godine dolazi do naglog smanjenja citiranosti, tako 2022. godina bilježi negativno odstupanje od -1.160, dok 2024. i 2025. godine pokazuju izrazito niske vrijednosti citiranosti (176 i 2 citata, respektivno) - ukazuje na sporiju integraciju novijih radova, budući da noviji radovi još nisu dovoljno često citirani.[34]

Rezultati su se pokazali korisnima za razumijevanje ciklusa akademske produkcije, predviđanje budućih trendova te optimizaciju istraživačkih strategija prilikom izbora ključnih referenci za područje predmetnog istraživanja.

Nadalje, prateći trendove pojavljivanja pojmova kroz vrijeme, slika, u nastavku, prikazuje učestalost određenih ključnih pojmova u znanstvenoj literaturi kroz razdoblje od 2019. do 2025. godine. Navedena analiza omogućuje identifikaciju trendova u korištenju određenih pojmova, čime se može pratiti razvoj interesa istraživačke zajednice za određene teme.



Slika 15: Pojavljivanje ključnih pojmova kroz vrijeme

Iz prikazane slike, pojam *MACHINE LEARNING* bilježi značajan rast kroz godine, od samo 7 pojavljivanja 2019. godine do 231 u 2025., a pojam *MACHINE-LEARNING*, koji je prisutan u nešto manjoj mjeri, također raste, s 0 pojavljivanja 2019. na 112 u 2025. Rezultati, prethodno navedenih pojmova, ukazuju na kontinuirano povećanje interesa za temu strojnog učenja, s posebnim naglaskom na varijacije u pisanju izraza.

Nadalje, postoji rastući interes za *AUTOMATED MACHINE LEARNING* odnosno *AutoML*. Pojam *AUTOMATED MACHINE LEARNING* nije bio značajno prisutan prije 2021. godine, ali od 2021. do 2025. bilježi rast s 5 na 58 pojavljivanja. Ovaj trend sugerira

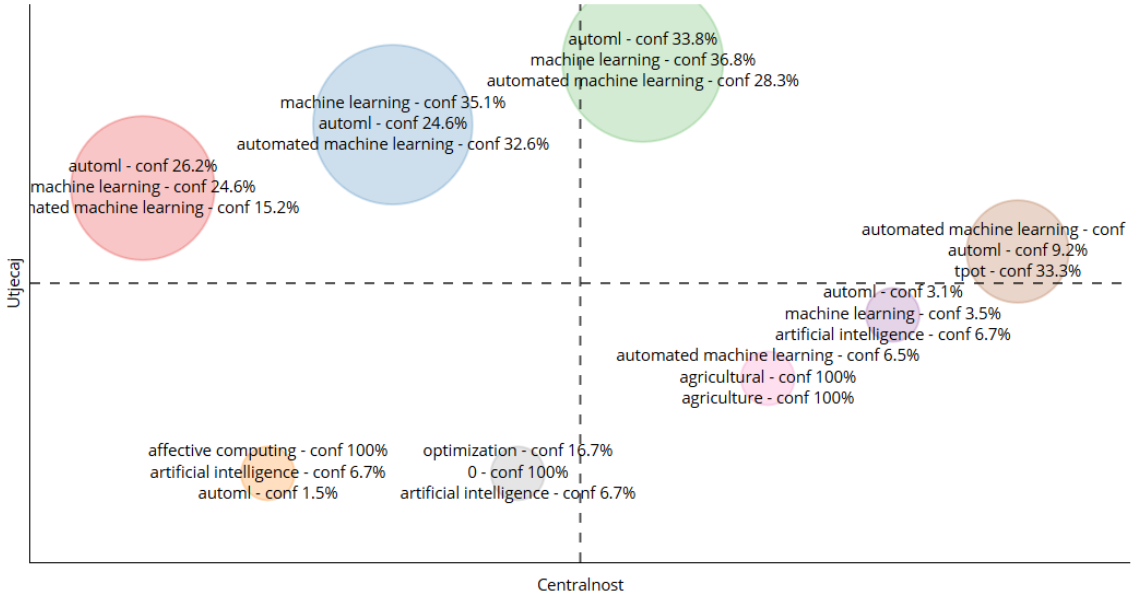
da se *AutoML* sve više prepoznaje kao važna poddisciplina strojnog učenja, čime raste njegova relevantnost u znanstvenim radovima.

Osim *AutoML*-a, postoji stabilan rast pojmova *AUTOMATION* i *AUTOMATED MACHINES*. Pojam *AUTOMATION* pokazuje kontinuirani rast, od 2 pojavljivanja 2019. godine do 61 u 2025., što ukazuje na sve veću prisutnost automatizacije kao koncepta u znanstvenoj literaturi. Zatim, pojam *AUTOMATED MACHINES* bilježi sličan trend, povećavajući se s 3 pojavljivanja 2019. godine na 77 u 2025. Navedeni rezultati ukazuju na to da su koncepti automatizacije i automatiziranih sustava postali sve značajniji u istraživanjima, kada je riječ o njihovoj vezi sa strojnim učenjem.

Na kraju, pojam *ALGORITHM* također pokazuje rastući trend, pojavljuje se konzistentno u znanstvenim radovima, s 3 pojavljivanja 2019. godine, dosežući 49 u 2025. Stabilan rast pojma *ALGORITHM* potvrđuje kontinuirani istraživački interes za teorijske i praktične aspekte algoritama, što je temelj strojnog učenja i *AutoML* sustava poput *MoziaisMl* sustava.

Rezultati ključnih pojmova ukazuju na eksponencijalni rast interesa za teme povezane sa strojnim učenjem, automatizacijom i algoritmima u znanstvenim radovima. Pojam *MACHINE LEARNING* i *MACHINE-LEARNING* ostaje dominantan pojam, dok pojmovi *AUTOMATION* i *AUTOMATED MACHINE LEARNING* bilježe stabilan rast, sugerirajući povećano istraživačko zanimanje za ove koncepte. Navedeni rezultati su ključni za razumijevanje razvoja istraživačkih prioriteta, što je korisno za identifikaciju budućih trendova oko *AutoML*-ova i ključnih tema povezanih s *AutoML*-ovima.

U nastavku je prikazana analiza klasteriranja pomoću bibliografske povezanosti (engl. *Clustering by Coupling*). Navedena analiza temelji se na bibliografskoj povezanosti dokumenata (engl. *document coupling*), gdje se istražuju znanstveni radovi koji dijele iste reference. Jedinica analize su rad/dokument (engl. *Unit of Analysis = Documents*), a povezanost među dokumentima mjeri se prema zajedničkim referencama (engl. *Coupling measured by References*). Kao mjera utjecaja koristi se rezultat globalnih citata (engl. *Global Citation Score*), dok su klasteri označeni prema ključnim riječima autora (engl. *Cluster Labeling = Authors' Keywords*).[34]



Slika 16: Klasteri bibliografske povezanosti

Klasteri bibliografske povezanosti pokazuju da su ključne istraživačke teme u ovom skupu od 725 znanstvenih radova usmjerene na strojno učenje i automatizirano strojno učenje (engl. *AutoML*), pri čemu pojedini klasteri dominiraju i postižu značajan akademski utjecaj. Klaster *automl - conf 33.8% machine learning - conf 36.8% automated machine learning - conf 28.3%* sadrži najveći broj radova (85) i najvišu akademsku citiranost (utjecaj je 3,05), što ukazuje na njegovu centralnu ulogu u istraživačkom području. Sličan značaj ima i *klaster machine learning - conf 35.1% automl - conf 24.6% automated machine learning* s 81 radom i utjecajem od 2.644, potvrđujući da su teme povezane s primjenom *AutoML*-a u strojnome učenju visoko relevantne i široko istraživane – što time potvrđuje i relevantnost predmetnog istraživanja ovog rada.

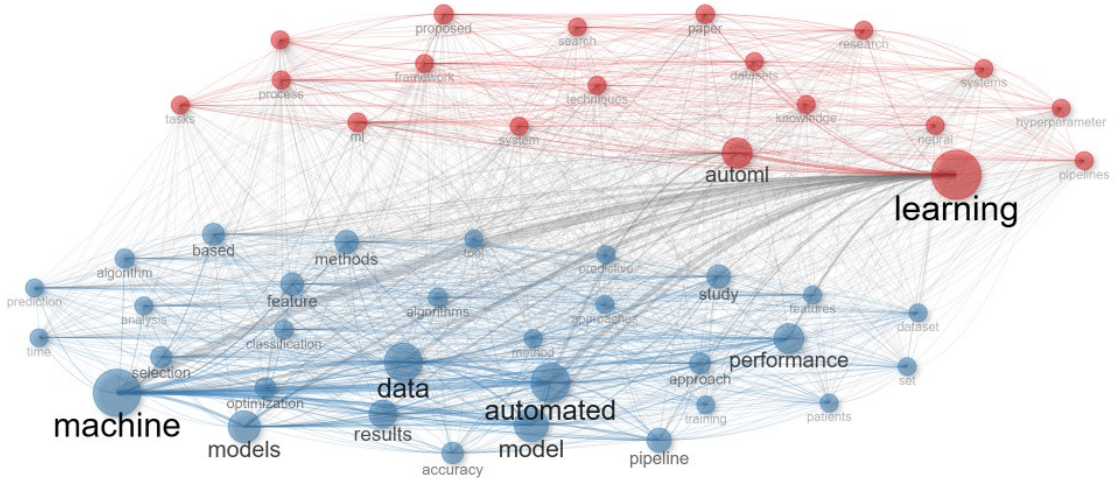
S druge strane, specijalizirani klasteri *poput automated machine learning - conf 15.2% automl - conf 9.2%* i *automated machine learning - conf 6.5% agricultural - conf 10%* imaju manji broj radova (16 i 7, respektivno), ali zadržavaju visok akademski utjecaj (2.555 i 1.491, respektivno). Prethodno navedeno sugerira da, iako ove teme nisu dominantne u široj znanstvenoj zajednici, one i dalje imaju važnu ulogu u specijaliziranim istraživačkim područjima, posebice u kontekstu automatizacije procesa u poljoprivredi i drugim domenama primjene strojnog učenja.

Na suprotnoj strani spektra slike nalaze se klasteri *affective computing - conf 100%* i *optimization - conf 16.7%*, koji imaju nisku frekvenciju (4 rada) i umjeren

akademski utjecaj (0.337 i 0.346, respektivno), što ukazuje na njihov periferni status unutar znanstvene zajednice. Navedeni klasteri predstavljaju specifične niše koje su manje istražene, ali mogu imati budući razvojni potencijal unutar šireg konteksta strojnog učenja.

Prethodni slikovni prikaz potvrđuje ovu analizu, gdje klasteri s visokom centralnošću i velikim akademskim utjecajem zauzimaju gornji desni kvadrant, dok se manje istraženi i specijalizirani klasteri nalaze na periferiji vizualizacije slike. Konkretno, *AutoML* i strojno učenje dominiraju kao temeljne istraživačke paradigme, dok optimizacija, afektivno računarstvo (engl. *affective computing*) i poljoprivredne primjene ostaju u nišnim područjima s mogućim potencijalom za budući rast. Navedeni rezultati pružaju uvid u strukturu znanstvene produkcije, omogućivši identificiranje dominantnih trendova i prepoznavanju područja u kojima postoji prostor za daljnji akademski doprinos.

Nadalje, analiza ko-učestalosti pojmova (engl. *Co-occurrence Network Analysis*) temelji se na frekvenciji i međusobnoj povezanosti ključnih riječi u sažecima znanstvenih radova. Koristeći jednostavne pojmove (engl. *Unigrams*) kao jedinicu analize, ova mrežna vizualizacija omogućila je identifikaciju najčešće korištenih pojmova i njihovih semantičkih veza unutar sažetaka predmetnog istraživačkog područja. Prikazani rezultati u mrežnoj karti, na slici u nastavku, pružaju uvid u strukturnu povezanost koncepta u području strojnog učenja i automatizacije.[34]



Slika 17: Mreža ko-učestalosti pojmova iz sažetaka

Prema slici, analizirani pojmovi grupirani su u dva glavna klastera (prikazana crvenom i plavom bojom na mrežnom grafu u slici), što sugerira postojanje dviju dominantnih tematskih podskupina u sažetcima znanstvenih radova:

1. Prvi klaster (crvena grupa) – *Learning* i *AutoML* su središnji pojmovi
    - Čvor *learning* odnosno učenje ima najvišu vrijednost posrednosti[1] (engl. *betweenness centrality*), 0,117, što ukazuje na njegovu ključnu ulogu u povezivanju ostalih koncepata.
    - Pojam *automl* odnosno automatizirano strojno učenje također zauzima važnu poziciju s vrijednosću posrednosti od 0,03, što pokazuje njegovu rastuću prisutnost u istraživačkom diskursu.
    - Ovaj klaster uključuje povezane pojmove poput *datasets, framework, process, research, hyperparameter* i *techniques* koji upućuju na fokus istraživanja na razvoj automatiziranih metodologija u strojnome učenju.
    - Semantičke veze između pojmova sugeriraju da je istraživačka zajednica usmjerena na razvoj automatiziranih procesa i optimizaciju modela strojnog učenja kroz *AutoML* sustave.
2. Drugi klaster (plava grupa) – Machine i Models su središnji pojmovi
    - Ključni čvorovi *machine* i *models* dominiraju ovim klasterom, pri čemu pojam machine ima najvišu vrijednost *PageRank*-a[2] (0,046), što ukazuje na njegov dominantan akademski utjecaj.
    - Ostali povezani pojmovi uključuju *algorithms, optimization, accuracy, classification i methods* što dovodi do zaključka da se istraživanja unutar ovog klastera primarno bave razvojem, evaluacijom i optimizacijom modela strojnog učenja.
    - Pojmovi *automated*, *performance* i *results* dodatno potvrđuju da istraživanja povezana s ovim klasterom uključuju analizu performansi

[1] *Betweenness centrality* otkriva koji su pojmovi važni za povezivanje različitih istraživačkih tema. Viskoa vrijednost znači da pojam služi kao veza između tematskih grupa. [34]

[2] *PageRank* pokazuje koji su pojmovi dominantni i najutjecajniji u istraživanju odnosno identificira temeljne koncepte. Visoka vrijednost predstavlja da je pojam ključan i često povezan. [34]

algoritama, usporedbe metoda i poboljšanja u domeni prediktivnog modeliranja.

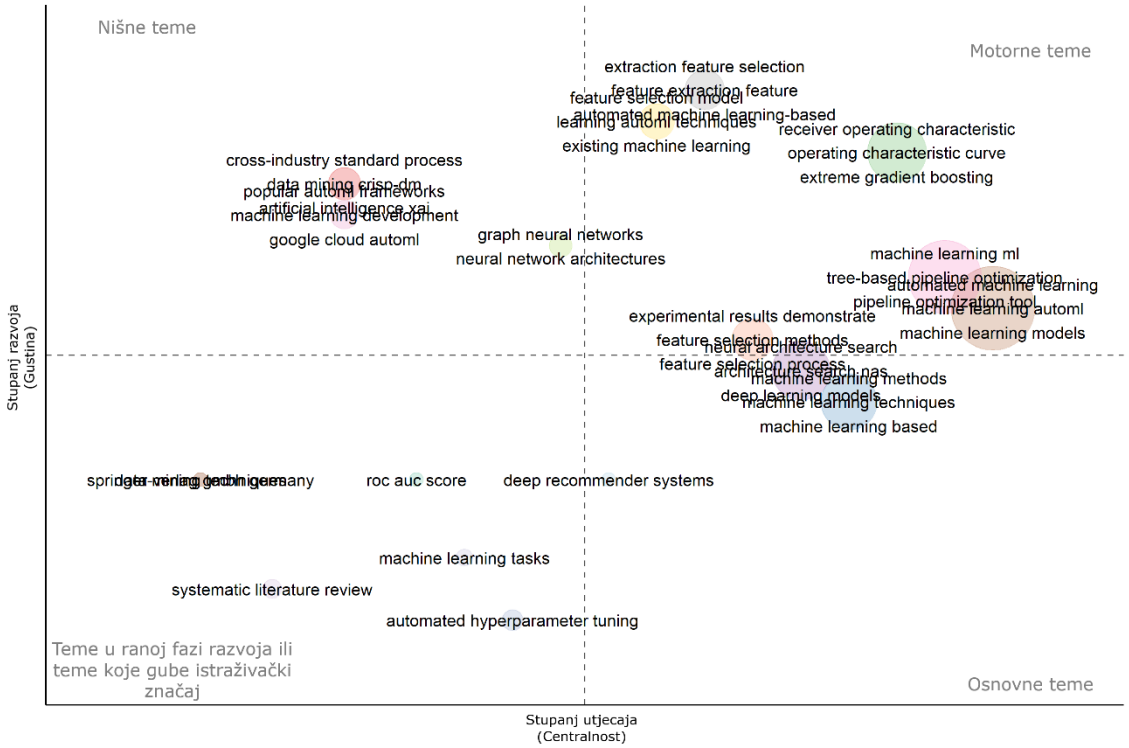
Mrežna vizualizacija ukazuje na snažnu povezanost između dvaju klastera, pri čemu se *learning* i *machine* javljaju kao ključni mostovi između dviju tematskih skupina. Ova struktura sugerira da se istraživanja unutar domene strojnog učenja paralelno razvijaju u dva glavna pravca:

1. Jedan pravac usmjeren je na *AutoML*, automatizaciju i optimizaciju hiperparametara, čime se nastoji poboljšati učinkovitost modela uz smanjenje ljudske intervencije.
2. Drugi pravac fokusira se na razvoj modela strojnog učenja, optimizaciju algoritama i poboljšanje prediktivne točnosti, pri čemu su tradicionalne metode strojnog učenja još uvijek dominantne.

Pravac u kojem se istraživači fokusiraju na brzinu odnosno vrijeme izvršavanja kao prvi parametar po rangu važnosti pa zatim prediktivna točnost – nije uočen.

Navedena analiza pokazuje da istraživački radovi u domeni strojnog učenja pokrivaju dvije međusobno povezane, ali tematski diferencirane skupine istraživanja. Dok prvi klaster istražuje automatizaciju i optimizaciju modela, drugi je fokusiran na tradicionalne metode strojnog učenja i njihove performanse. Analiza je identificirala dominantne trendove i pronašla potencijalno područje za daljnje istraživanje unutar ovih tematskih skupina – brzina odnosno vrijeme izvršavanja kao prvi rang važnosti ispred prediktivne točnosti.

Nadalje, prikazana je tematska mapa (engl. *Thematic Map*) koja koristi analizu složenih pojmova (engl. *Trigrams*) iz sažetaka znanstvenih radova kako bi vizualizirala struktura i prikazali razvojni trendovi istraživačkih tema. Mapa prikazuje četiri kvadranta, koji odgovaraju različitim kategorijama istraživačkih tema, pri čemu su os X (Stupanj utjecaja - centralnost) i os Y (Stupanj razvoja - gustina) ključni pokazatelji važnosti i zrelosti određene teme u istraživačkom području.[34]



Slika 18: Tematska mapa složenih pojmova

Na slici su prikazane[34]:

1. Motorne teme (Gornji desni kvadrant, engl. *Motor Themes*) – Ključne istraživačke domene u razvoju
    - Motorne teme odlikuju se visokom centralnošću i visokom gustoćom, što znači da su temeljna područja istraživanja s intenzivnim razvojem i velikim utjecajem na druge teme.
    - Najznačajnije teme uključuju:
        - *receiver operating characteristic, operating characteristic curve, extreme gradient boosting* - navedene teme ukazuju na široku upotrebu ROC analize i naprednih metoda optimizacije modela strojnog učenja.
        - *machine learning ml, pipeline optimization, automated machine learning* - potvrđuje se trend automatizacije i optimizacije strojnog učenja, čime se smanjuje potreba za ručnim podešavanjem hiperparametara.

- Zaključuje se kako su prikazane teme dobro definirane, stabilne i ključne za budući razvoj istraživanja u strojnome učenju i automatiziranom procesiranju podataka.

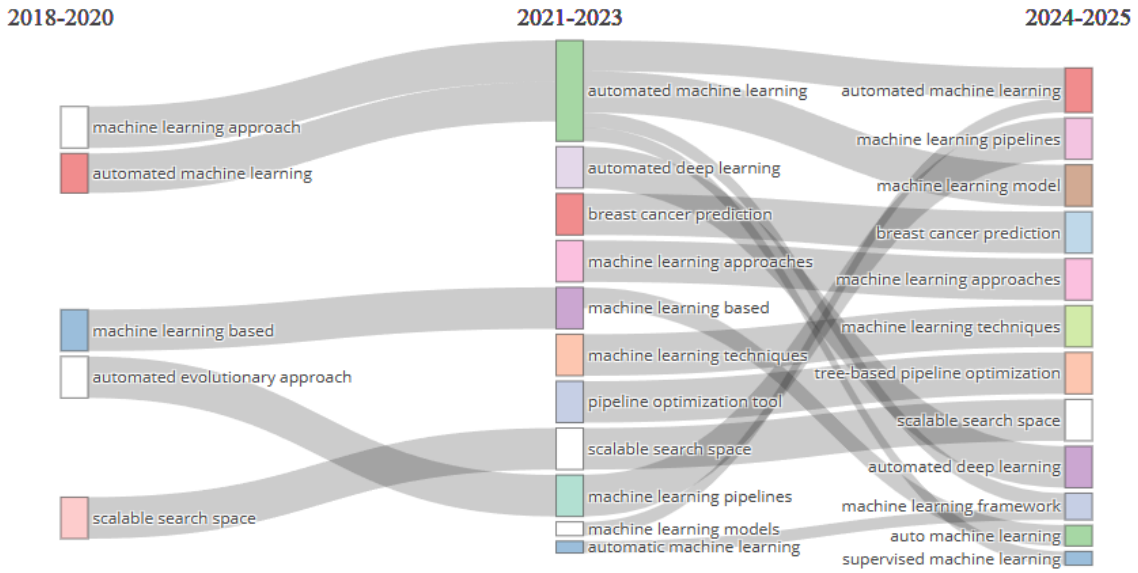
2. Osnovne teme (Donji desni kvadrant, engl. *Basic Themes*) – Široko istraživane, ali ne nužno inovativne
    - Osnovne teme imaju visoku centralnost, ali nisku gustoću, što znači da su čvrsto povezane s drugim temama, ali još uvijek nisu u potpunosti razvijene. Teme su:
        - *machine learning methods, deep learning architectures, feature selection methods* - temeljne metode strojnog učenja i arhitekture dubokog učenja ostaju ključni elementi istraživanja.
        - *experimental results demonstrate, feature selection techniques, neural architecture search* - fokus na optimizaciju modela i poboljšanje prediktivne točnosti.
        - Zaključuje se kako su prikazane teme već etablirane, ali će se vjerojatno razvijati dalje, posebice u kontekstu poboljšanja učinkovitosti modela.
3. Nišne teme (Gornji lijevi kvadrant, engl. *Niche Themes*) – Specijalizirana, ali manje relevantna istraživanja
    - Niske centralnosti i visoke gustoće, nišne teme su visoko specijalizirane i izolirane od šireg istraživačkog područja. Najistaknutije teme su:
        - *cross-industry standard process, popular automl frameworks, artificial intelligence development* - ove teme sugeriraju specifične domene primjene *AutoML*-a i standardizaciju postupaka.
        - Zaključuje se kako prikazane iako su važne u određenim kontekstima, navedene teme nisu široko povezane s drugim istraživanjima i vjerojatno će ostati specijalizirane.

4. Teme u ranoj fazi razvoja ili teme koje gube istraživački značaj (Donji lijevi kvadrant, engl. *Emerging or Declining Themes*) – Tematska područja u nastajanju ili u padu
    - Ove teme imaju nisku centralnost i nisku gustoću, što znači da su ili u ranoj fazi razvoja ili gube istraživački značaj. Najvažnije teme ovdje su:
        - *systematic literature review, machine learning tasks, automated hyperparameter tuning* - Ove teme sugeriraju istraživačke sinteze i optimizaciju hiperparametara, no njihova niska centralnost može ukazivati na to da nisu dominantne u aktualnom istraživačkom području.
        - *data mining techniques, roc auc score, deep recommender systems* - nekada su bile značajne, ali sada su manje utjecajne.
        - Zaključuje se kako ove teme mogu predstavljati buduće trendove, no postoji mogućnost da su istraživači preusmjerili fokus na novije pristupe.

Tematska mapa prikazuje dvije dominantne istraživačke teme – *AutoML* i optimizacija modela kao ključne motorne teme te temeljne metode strojnog učenja i značajki kao osnovne teme. Istovremeno, specijalizirane nišne teme i potencijalno opadajući istraživački trendovi pružaju uvid u buduće istraživačke smjerove i identificiraju područja koja su izgubila akademsku relevantnost. Navedena analiza omogućila je identifikaciju ključnih istraživačkih trendova i priliku za daljnji doprinos unutar dinamičnog polja strojnog učenja, predlaganjem okvira razvoja informacijskog sustava temeljenog na metodama strojnog učenja za vlastito razvijeni sustav automatiziranog strojnog učenja, naziva *MoziaisMl* gdje je razvijen algoritam latencije koji stavlja brzinu odnosno vrijeme izvršavanja kao prvi rang važnosti ispred prediktivne točnosti.

Na kraju, prikazana je analiza tematske evolucije. U bibliometrijskoj analizi omogućuje razumijevanje promjena u fokusu istraživanja tijekom vremena, identificirajući razvojne puteve ključnih koncepata i njihovih međusobnih povezanosti.

Prikazani podatci obuhvaćaju tematske prijelaze u tri vremenska razdoblja: 2018–2020, 2021–2023 i 2024–2025, a analiza je izvedena na temelju naslova radova.[34]



Slika 19: Analiza tematske evolucije

Slikovni prikaz tematske evolucije ukazuje na razvoj istraživačkih tema povezanih s automatiziranim strojnim učenjem (*Automated Machine Learning – AutoML*) i njegovom integracijom s drugim metodologijama. Tijekom razdoblja 2018–2020, istraživanje je bilo usmjereno na osnovne koncepte, poput *machine learning approach*, *automated machine learning* i *scalable search space*. Ovi su koncepti činili temelj istraživanja u području automatizacije modela strojnog učenja.

U razdoblju 2021–2023 dolazi do diverzifikacije i specijalizacije istraživačkih tema. Primjetan je razvoj u području *automated deep learning*, *pipeline optimization tool*, *machine learning techniques* i *breast cancer prediction*, što sugerira primjenu *AutoML*-a u različitim domenama, uključujući medicinu i optimizaciju cjevovoda modela strojnog učenja. Također, ključne teme poput *machine learning pipelines* i *automatic machine learning* postaju sve prisutnije, što ukazuje na rastući interes za razvojem automatiziranih rješenja u znanstvenim istraživanjima.

U razdoblju 2024–2025, analizirani podatci pokazuju konvergenciju i daljnju specijalizaciju tema, pri čemu ključni koncepti poput *automated machine learning*, *machine learning pipelines* i *supervised machine learning* postaju dominantni. Ovo sugerira povećani interes za integraciju automatiziranih pristupa s nadziranim učenjem te

daljnji razvoj modela skalabilnog strojnog učenja. Također, pojava tema poput *machine learning framework* i *tree-based pipeline optimization* ukazuje na usmjerenje prema strukturnoj optimizaciji modela.

Tablični podatci, u nastavku, detaljno prikazuju prijelaz ključnih istraživačkih tema i koncepata iz jednog vremenskog razdoblja u drugo. Na primjer, pojava *automated machine learning* kao dominantne teme u sva tri razdoblja s visokim indeksom ponderirane uključenosti[3] (engl. *Weighted Inclusion Indexom)* - 0,95, pokazuje njegovu konzistentnu prisutnost i važnost u znanstvenim istraživanjima. S druge strane, teme poput *supervised machine learning* pojavljuju se tek u posljednjem razdoblju, što ukazuje na novu nišu istraživanja. Indeks stabilnosti[4] (engl. *Stability Index*) od 0,50 za određene teme sugerira postojanost u istraživačkom fokusu, dok niže vrijednosti ukazuju na novonastale ili prolazne teme.

Od	**Do**	**Riječi**	**Indeks ponderirane uključenosti**	**Indeks uključenosti[5]**	**Pojavljivanja**	**Indeks stabilnosti**
automated evolutionary approach	machine learning pipelines--2021-2023	automated evolutionary approach;composite machine	1,00	0,33	2	0,20

[3] Indeks mjeri koliko je neka tema iz prethodnog vremenskog razdoblja uključena u novo razdoblje, ali uz ponderiranje na temelju broja pojavljivanja ili drugih faktora koji naglašavaju značaj teme. Što je veći Indeks ponderirane uključenosti, to znači da je tema izuzetno važna i dominantna u oba razdoblja. Kad je vrijednost blizu 1, tema je visoko povezana s novim razdobljem.[34]

[4] Indeks pokazuje koliko je neka tema postojana i stabilna kroz različita vremenska razdoblja. Vrijednost blizu 1 označava stabilnu temu koja se kontinuirano istražuje, dok niska vrijednost sugerira kratkotrajni trend ili prolazni interes. [34]

[5] Indeks mjeri postotak podudarnosti između dvije teme u različitim vremenskim razdobljima, odnosno koliko se tema iz prošlog razdoblja pojavljuje u novom. Vrijednost blizu 1 znači da je tema potpuno prisutna u oba razdoblja, dok niže vrijednosti ukazuju na djelomičnu povezanost ili promjene u fokusu istraživanja.[34]

--2018-2020		learning;machine learning pipelines				
automated machine learning--2018-2020	automated machine learning--2021-2023	automated machine learning;tree-based automated machine;biology-based feature selection;coronary artery disease;machine learning automl;machine learning informed;machine learning pipeline	0,95	0,13	29	0,03
machine learning approach--2018-2020	automated machine learning--2021-2023	machine learning approach	1,00	1,00	2	0,04
machine learning based--	machine learning based--	machine learning based	1,00	1,00	3	1,00

2018-2020	2021-2023					
scalable search space--2018-2020	scalable search space--2021-2023	scalable search space;search space decomposition	1,00	0,50	3	0,33
automated deep learning--2021-2023	automated deep learning--2024-2025	automated deep learning	1,00	1,00	3	0,33
automated machine learning--2021-2023	automated machine learning--2024-2025	automated machine learning;machine learning pipeline;machine learning approach;machine learning automl	0,74	0,08	90	0,03
automated machine learning--2021-2023	machine learning framework--2024-2025	machine learning framework	0,36	0,25	5	0,03

automated machine learning--2021-2023	machine learning model--2024-2025	machine learning model	1,00	1,00	4	0,04
automated machine learning--2021-2023	supervised machine learning--2024-2025	supervised machine learning	0,33	0,50	2	0,04
automatic machine learning--2021-2023	machine learning framework--2024-2025	automatic machine learning	0,29	0,25	5	0,10
breast cancer prediction--2021-2023	breast cancer prediction--2024-2025	breast cancer prediction;grid search method	1,00	0,50	3	0,25
machine learning approaches--2021-2023	machine learning approaches--2024-2025	machine learning approaches	1,00	1,00	2	0,50

machine learning based--2021-2023	auto machine learning--2024-2025	machine learning based	0,50	1,00	4	0,50
machine learning models--2021-2023	automated machine learning--2024-2025	machine learning models	0,33	0,25	6	0,06
machine learning pipelines--2021-2023	machine learning pipelines--2024-2025	machine learning pipelines	1,00	1,00	4	0,33
machine learning techniques--2021-2023	machine learning techniques--2024-2025	machine learning techniques	1,00	1,00	4	0,25
pipeline optimization tool--2021-2023	tree-based pipeline optimization--2024-2025	pipeline optimization tool;tree-based pipeline optimization	1,00	0,50	4	0,33

scalable search space--2021-2023	scalable search space--2024-2025	scalable search space;search space decomposition	1,00	0,50	6	0,33

Tablica 15: Analiza tematske evolucije

Zaključno, tematska analiza pokazuje kontinuirani rast i razvoj *AutoML*-a, s početnih tema i koncepata koji su se bavili temeljnim metodološkim pristupima, prema sve većoj specijalizaciji i praktičnoj primjeni u različitim domenama. Povećani interes za *machine learning pipelines* i *tree-based pipeline optimization* sugerira usmjerenje prema optimizaciji i skalabilnosti algoritama, dok prisutnost *supervised machine learning* ukazuje na nastavak integracije nadziranog učenja u automatizirane procese. Prethodno navedeni rezultati pružili su uvide u smjerove budućeg istraživanja i potencijalne tematske prilike za daljnji znanstveni doprinos, predlaganjem metodološkog okvira razvoja analitičkog informacijskog sustava temeljenog na metodama strojnog učenja za vlastito razvijeni sustav automatiziranog strojnog učennja, naziva *MoziaisMl* gdje je razvijen i ugrađen algoritam latencije koji stavlja brzinu odnosno vrijeme izvršavanja kao prvi rang važnosti ispred prediktivne točnosti, pri čemu je validnost znanstvenog doprinosa potvrđena definiranom funkcijom cilja.

Nakon početnog istraživanja i analize o razvoju sustava za automatizaciju cijelog procesa razvoja modela strojnog učenja, u nastavku će se istražiti praznine u istraživanju pristupa automatizaciji algoritama, inženjerstva značajki, redukciji dimenzionalnosti, optimizaciji hiperparametara, odabira modela strojnog učenja, performansi sustava i metodologija. S tom namjerom su se i postavili upiti te iz tih upita izdvojili radovi najbliži temi rada za daljnju analizu praznina u istraživanju.

2.2. Praznine u istraživanju relevantne literature

Autori Li, et al. [39] u svom radu su predstavili *VolcanoML*, skalabilni okvir osmišljen za učinkovito istraživanje velikih pretraživačkih prostora u *AutoML* sustavima.

Sustav *VolcanoML* uvodi strukturne komponente koje omogućuju dekompoziciju složenih pretraživačkih prostora u manje, upravljive dijelove, čime se poboljšava učinkovitost i prilagodljivost modela strojnog učenja. Autori Li, et al. [39] su prepoznali izazov skalabilnosti *AutoML* sustava koji proizlazi iz inherentne složenosti velikih pretraživačkih prostora te ističu potrebu za uvođenjem strukturiranih apstrakcija koje omogućuju njihovu učinkovitu dekompoziciju. Također, naglašavaju da postojeći sustavi često nailaze na poteškoće pri optimizaciji pretraživačkog prostora u kontekstu zajedničkog prilagođavanja više komponenti modela, što ukazuje na nedostatke u postojećim metodologijama. U radu su analizirali više strategija dekompozicije pretraživačkog prostora, pri čemu ističu da samo određeni pristupi postižu zadovoljavajuće performanse, što otvara prostor za daljnja istraživanja i unaprjeđenja. Nadalje, identificirali su ograničenja postojećih *AutoML* sustava u proširivanju funkcionalnosti s domensko-specifičnim značajkama, što ukazuje na manjak fleksibilnosti i prilagodljivosti trenutnih rješenja.[39]

Nadalje, autori Gosiewska, et al. [40] su predložili *SAFE* algoritam kao pristup za poboljšanje performansi modela strojnog učenja uz istovremeno povećanje njegove interpretabilnosti. U radu se naglašava potreba za razvojem pouzdanih prediktivnih modela koji istovremeno osiguravaju visoku točnost, auditabilnost, interpretabilnost i automatiziranost u procesu analize podataka. Trenutne praznine u istraživanju *SAFE* algoritma su identifikacija i ekstrakcija interakcija između značajki, što bi dodatno unaprijedilo sposobnost modela da prepozna složene odnose unutar podataka. Nadalje, razvoj automatiziranih transformacija podataka predstavlja ključnu smjernicu za daljnje poboljšanje prediktivne sposobnosti modela. Poseban naglasak stavlja se na potrebu za povećanjem interpretabilnosti novo konstruiranih značajki, za što, autori Gosiewska, et al. [40], ističu da je ključno za izgradnju povjerenja u modele strojnog učenja i njihovu primjenu u stvarnim scenarijima.[40]

Zatim, autori Yakovlev, et al. [41] predstavljaju *Oracle AutoML*, napredni *AutoML* sustav koji omogućuje razvoj i implementaciju modela bez iterativnog procesa, čime se značajno ubrzava razvoj modela strojnog učenja. U radu se naglašava izazov ponovljivosti rezultata *AutoML* sustava, što je ključno za regulatornu usklađenost i objašnjivost modela. Posebna pažnja posvećena je optimizaciji među-modelske i unutar-modelske paralelizacije kako bi se postigle optimalne performanse, osobito u

okruženjima računalnog oblaka. Nadalje, rad prepoznaje problematiku meta-učenja (engl. *metalearning*), čiji složeni i dugotrajni proces može predstavljati prepreku u praktičnoj primjeni *AutoML* sustava. Također, istaknuli su otvoreno istraživačko pitanje koje se odnosi na sposobnost *AutoML* sustava da generalizira znanje i prilagodi se disruptivnim promjenama u temeljnim sustavima, što predstavlja ključni izazov u daljnjem razvoju automatiziranih tehnika strojnog učenja.[41]

Autori Fister, et al. [42] predstavljaju *NiaAML*, napredni *AutoML* okvir namijenjen automatiziranom kreiranju cjevovoda strojnog učenja i optimizaciji hiperparametara. U radu ističu potrebu za integracijom tehnika dubokog učenja u *NiaAML* okvir, čime bi se dodatno proširila njegova primjenjivost i istraživački potencijal. Autori Fister, et al. [42] ukazuju na nedostatak istraživanja pristupa u kontekstu poboljšanja robusnosti modela dubokog učenja protiv adverzijalnih (engl. *adversarial*) napada. Nadalje, naglašavaju važnost razvoja objašnjivih (engl. *explainable*) AI modela koristeći tehnike nadahnute prirodnim mehanizmima, čime bi se povećala transparentnost i razumljivost prediktivnih modela. Autori Fister, et al. [42] ističu kako bi se buduće istraživanje moglo usmjeriti na optimizaciju hiperparametara i arhitekture dubokih neuronskih mreža u stvarnom vremenu, čime bi se poboljšale njihove prilagodljive sposobnosti u dinamičnim uvjetima. Također, u radu sugeriraju daljnja istraživanja dinamičke optimizacije modela u okruženjima koja su podložna promjenama, čime bi se omogućila bolja generalizacija i prilagodljivost modela strojnog učenja u stvarnim aplikacijama.[42]

Nadalje, autor EldeebiElshawi [43] predstavlja BigFeat, skalabilan i objašnjiv okvir za automatizaciju ključnih faza cjevovoda strojnog učenja, uključujući inženjerstvo značajki i optimizaciju hiperparametara. U radu naglašavaju kako primjena mjere informacijskog dobitka za određivanje važnosti značajki nije optimalno rješenje za sve modele strojnog učenja, budući da može dovesti do redundancije značajki unutar skupa podataka koji sadrže velik broj međusobno sličnih atributa. Nadalje, istaknuta je ograničena primjenjivost dubokih neuronskih mreža zbog njihove zahtjevnosti u pogledu podataka i računalnih resursa, čime se sužava njihova operabilnost u uvjetima s ograničenim kapacitetima. Ističu, kako je jedan od ključnih izazova postojećih *AutoML* okvira nedostatak interpretabilnosti, što otežava donošenje pouzdanih odluka temeljenih na predikcijama modela. Također, u radu identificiraju problem ekspanzije značajki

(engl. *feature explosion*), koji nastaje kada se u skup podataka dodaju pretjerano veliki broj novih značajki, što može smanjiti efikasnost modela te povećati računalnu složenost procesa modeliranja.[43]

Autori Kedziora, et al. [44] analiziraju razvoj sustava koji omogućuje autonomnu prilagodbu rješenja strojnog učenja tijekom vremena. Poseban naglasak stavljen je na meta-učenje, pri čemu se ističe predviđanje vremena izvođenja algoritama kao potencijalna prilika za unaprjeđenje procesa raspoređivanja i upravljanja resursima. Autori Kedziora, et al. [44] ističu kako su dosadašnja istraživanja zanemarivala vremenske zahtjeve prilikom evaluacije modela, što je imalo negativne implikacije na njihovu praktičnu primjenu u stvarnim sustavima. Nadalje, u radu prepoznaju neizbježne kompromisne odluke u dizajnu sustava strojnog učenja te ističu kako će vjerojatno dizajn sustava ostati konstantna prepreka u budućem razvoju. Ističu i fragmentiranost postojećih pristupa, pri čemu se različiti mehanizmi i teorijski okviri još uvijek ne integriraju na sustavan način unutar *AutoML* sustava, što otežava njihovu skladnu optimizaciju i primjenu u dinamičnim okruženjima.[44]

Na kraju, autori Wu, et al. [45] predstavljaju *ChaCha* algoritam, inovativni pristup za online odabir hiperparametara unutar sustava strojnog učenja. Naglašavaju važnost učinkovitog otkrivanja optimalnih konfiguracija unutar ograničenih računalnih resursa, pri čemu je jedan od ključnih izazova pouzdana identifikacija boljih konfiguracija i predviđanje među suprotstavljenim alternativama. Ističu kako, poseban problem predstavlja evaluacija performansi algoritama u dinamičnim podatkovnim okruženjima, gdje kontinuirane promjene u podatcima mogu značajno utjecati na stabilnost i generalizacijsku sposobnost modela. Također, ističu da odabir veličine podskupa podataka ima kritičan utjecaj na konačnu učinkovitost modela, pri čemu je taj izbor specifičan za svaki skup podataka. Zaključno, u radu naglašavaju potrebu za strogo definiranim računalnim ograničenjima u kontekstu online učenja, gdje je ključno optimizirati računalnu složenost, a pritom zadržati visoku razinu prediktivne preciznosti.[45]

S obzirom na istraživačke praznine povezane s problemom prevelikog broja značajki, koje su istaknuli EldeebiElshawi [43], ovo istraživanje pruža znanstveni doprinos s razvojem vlastitih metoda redukcije dimenzionalnosti. Razvijene metode

omogućuju dinamički odabir veličine podskupa podataka, pri čemu se taj odabir prilagođava specifičnostima pojedinog skupa podataka, što je u skladu s preporukama autora Wu, et al. [45].

Dodatno, Kedziora, et al. [44] naglašavaju vrijeme izvršavanja algoritama kao ključni čimbenik optimizacije, što je u ovom istraživanju adresirano razvojem metodološkog okvira i automatiziranog cjevovoda. Predloženi okvir uključuje algoritam latencije, koji prioritizira brzinu izvršavanja ispred prediktivne točnosti odnosno greške, čime se optimizira računalna učinkovitost modela odnosno smanjuju računalni troškovi.

2.3. Procjena tehnologija kao znanstveni doprinos

Pregled relevantne literature je pokazao da većina znanstvenih radova u području automatiziranog strojnog učenja ima postavljen, ali ne nužno CRISP-DM, metodološki proces od pripreme podataka, treniranja modela strojnog učenja pa do evaluacije, no redukcija dimenzionalnosti i latencija modela, zajedno, nisu dovoljno razrađeni u postojećim metodološkim procesima relevantnih znanstvenih radova. Analiza tematske mape i mreže ko-učestalosti ukazuje da su pojmovi poput *automated machine learning*, *feature selection* i *optimization* ključni, ali nisu integrirani s vremenskom komponentom izvođenja modela strojnog učenja.

Prema analizi istraživačkih praznina u prethodnom poglavlju, niti jedan od navedenih radova ne integrira latenciju modela, niti ima inferencijsko vrijeme kao parametar funkcije cilja unutar optimizacijskog procesa automatiziranog cjevovoda. Nadalje, tematska evolucija pokazuje jasan prijelaz s pojmova poput *automated evolutionary approach* i *scalable search space* prema *machine learning pipelines* i *pipeline optimization tool*, ali bez direktnog povezivanja s latencijom odnosno metrikama inferencijskog vremena izvođenja modela strojnog učenja.

Kroz analizu *PageRank* i *betweenness centrality* u mreži ko-učestalosti vidi se da pojmovi *automl* i *performance* čine semantičku jezgru pojmova unutar prikazanih klastera. Međutim, latencija kao koncept, definiran u ovom doktorskom radu, nije ni među čvorištima ni u semantički centralnim pojmovima prikazanih klastera, što potvrđuje

da je u relevantnoj literaturi za ovaj doktorski rad zanemarena. Stoga se otvara istraživačka praznina za konceptualno pozicioniranje latencije i inferencijskog vremena izvršavanja kao primarne metrike funkcije cilja u višekriterijskoj optimizaciji.

Na temelju detaljne tehničke analize relevantne literature, u ovom doktorskom radu se predlaže razvoj automatiziranog sustava naziva *MoziaisMl* koji je modularan prema CRISP-DM fazama, ima integriran automatizirani algoritam latencije, koristi vlastite metode redukcije dimenzionalnosti, evaluira dobivenu predikciju s višestrukim metrikama (inferencijsko vrijeme i točnost/greška) te ima generalizacijsku sposobnost primjene na različite poslovne domene.

S obzirom na identificirane istraživačke praznine, može se zaključiti da trenutno stanje relevantne znanstvene literature pruža potrebne temelje za razvoj *MoziaisMl* sustava jer ne postoji strukturirani i empirijski potvrđeni pristup koji integrira metode redukcije dimenzionalnosti i optimizaciju latencije kao istraživački problem. Predloženi pristup razvoja *MoziaisMl* sustava predstavlja doprinos u vidu automatiziranog algoritma latencije s višekriterijskim odlučivanjem i optimizacijom latencije te normiranom evaluacijom putem funkcije cilja. Stoga, rezultati ovog pregleda relevantne literature potvrđuju cilj i svrhu istraživanja u vidu znanstvenog doprinosa metodologiji za izradu *MoziaisMl* sustava. Nadalje, predložene metode za pretprocesiranje podataka i redukciju dimenzionalnosti unutar ovog okvira dodatno unaprjeđuju učinkovitost automatiziranih modela strojnog učenja, osiguravajući balans između vremena izvršavanja koje proizlazi iz računalne složenosti i prediktivne preciznosti.

U sljedećem poglavlju predstavljene su teorijske osnove popularnih i korištenih tehnika odnosno metoda redukcije dimenzionalnosti, koje će se usporediti s vlastito razvijenim metodama za redukciju dimenzionalnosti.

3. TEHNIKE REDUKCIJE DIMENZIONALNOSTI

Svrha i cilj ovog poglavlja je predstaviti teorijske osnove popularnih i korištenih tehnika odnosno metoda redukcije dimenzionalnosti jer prilikom rada sa stvarnim podatcima, česta je pojava visokodimenzionalnih skupova podataka, koji mogu sadržavati stotine značajki. Iako skupovi podataka sa svim značajkama u svojoj izvornoj dimenzionalnoj strukturi sadrže maksimalnu količinu informacija, u slučajevima kada se treba postići niže vrijeme izvršavanja predikcije preciznosti ili smanjiti računalne troškove odnosno smanjiti upotrebu računalnih resursa javlja se potreba za redukcijom dimenzionalnosti. U nastavku prikazane su teorijske osnove najčešće korištenih tehnika koje su detaljno opisane u sljedećim potpoglavljima – PCA[14-16], t-SNE[14-17], UMAP[18] i autoenkoderske neuronske mreže[19].

3.1. Teorijske osnove redukcije dimenzionalnosti

Redukcija dimenzionalnosti često se provodi u eksplorativnoj analizi podataka radi vizualizacije podataka, pri čemu se podatci sa stotinama značajki projiciraju u 2D ili 3D prostor kako bi se, na jednostavniji način, omogućila njihova interpretacija. Međutim, primjena redukcije dimenzionalnosti nije ograničena samo na vizualizaciju. Postoje scenariji, kada brzina izvršavanja metoda strojnog učenja ima prednost nad postizanjem maksimalne predikcijske preciznosti, gdje se, primjer, skup podataka s 350 značajki može reducirati na 35 značajki kako bi se ubrzalo vrijeme izvršavanja predikcije i smanjili računalni troškovi.[14-19, 46]

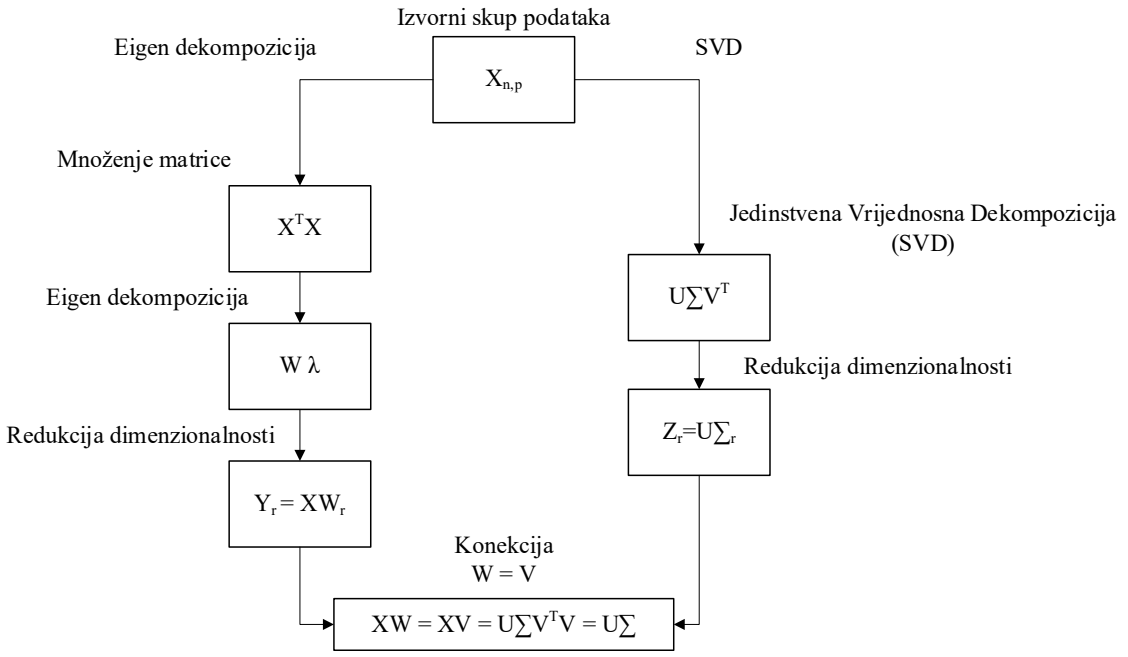
S obzirom na široku primjenu u raznim domenama, tehnike redukcije dimenzionalnosti predstavljaju neizostavan segment analize i procesiranja podataka te omogućuju bržu analizu, poboljšanu interpretaciju i optimizaciju modela strojnog učenja, osobito u kontekstu velikih i složenih skupova podataka.[46] Nastavno na prethodno navedeno, u ovom poglavlju predstavljaju se četiri korištene i popularne metode redukcije dimenzionalnosti – PCA[14-16], t-SNE[14-17], UMAP[18] i autoenkoderske neuronske mreže[19]. Četiri prethodno navedene i široko korištene metode redukcije dimenzionalnosti primjenjuju se na značajke, odnosno atribute unutar skupa podataka, gdje se svaka značajka može interpretirati kao komponenta u višedimenzionalnom

prostoru. U primjeru skupa podataka koji sadrži 350 značajki, postupkom redukcije dimenzionalnosti, izdvajanje 35 značajki rezultira projekcijom podataka u prostor smanjene dimenzionalnosti, pri čemu odabrane značajke predstavljaju glavne komponente novog dimenzionalnog, 35 komponentnog, prostora. Time se postiže eliminacija redundantnih informacija, istovremeno ojačavajući što veću količinu relevantnih podataka, gdje se poboljšava učinkovitost odnosno vrijeme izvršavanja predikcije uz smanjenje računalnih troškova.[14-19, 46, 47]

3.2. Analiza glavnih komponenti

Temeljna ideja i cilj PCA jest smanjenje dimenzionalnosti skupa podataka koji se sastoji od velikog broja međusobno povezanih varijabli, pri čemu se nastoji očuvati što veći dio informacija sadržanih u izvornim podatcima. Prethodno navedeni cilj, postiže se transformacijom podataka u novi skup značajki, poznatih kao glavne komponente (engl. *Principal Components*, PCs). Glavne komponente su međusobno nekorelirane te su poredane prema redoslijedu koji osigurava da prve komponente zadrže najveći mogući udio ukupnih informacija prisutnih u izvornom skupu podataka. Ovakav pristup omogućava učinkovitu reprezentaciju podataka u prostoru niže dimenzionalnosti, uz minimalni gubitak informacija, što je osobito korisno u kontekstu vizualizacije, ubrzanja izračuna i smanjenja složenosti modela.[47]

Najčešća dva korištena pristupa smanjenja dimenzionlanosti skupa podataka temelje na *eigen* dekompoziciji (engl. *Eigen decomposition*) i jedinstvenoj vrijednosnoj dekompoziciji (engl. *Singular Value Decomposition*, SVD). Autor Elior Cohen u *Medium TDS Archive* publikaciji[46] prikazao je oba pristupa s primjerom i grafičkim prikazom radnog tijeka PCA analize u nastavku.



Slika 20: Radni tijek PCA analize (Izvor: [46])

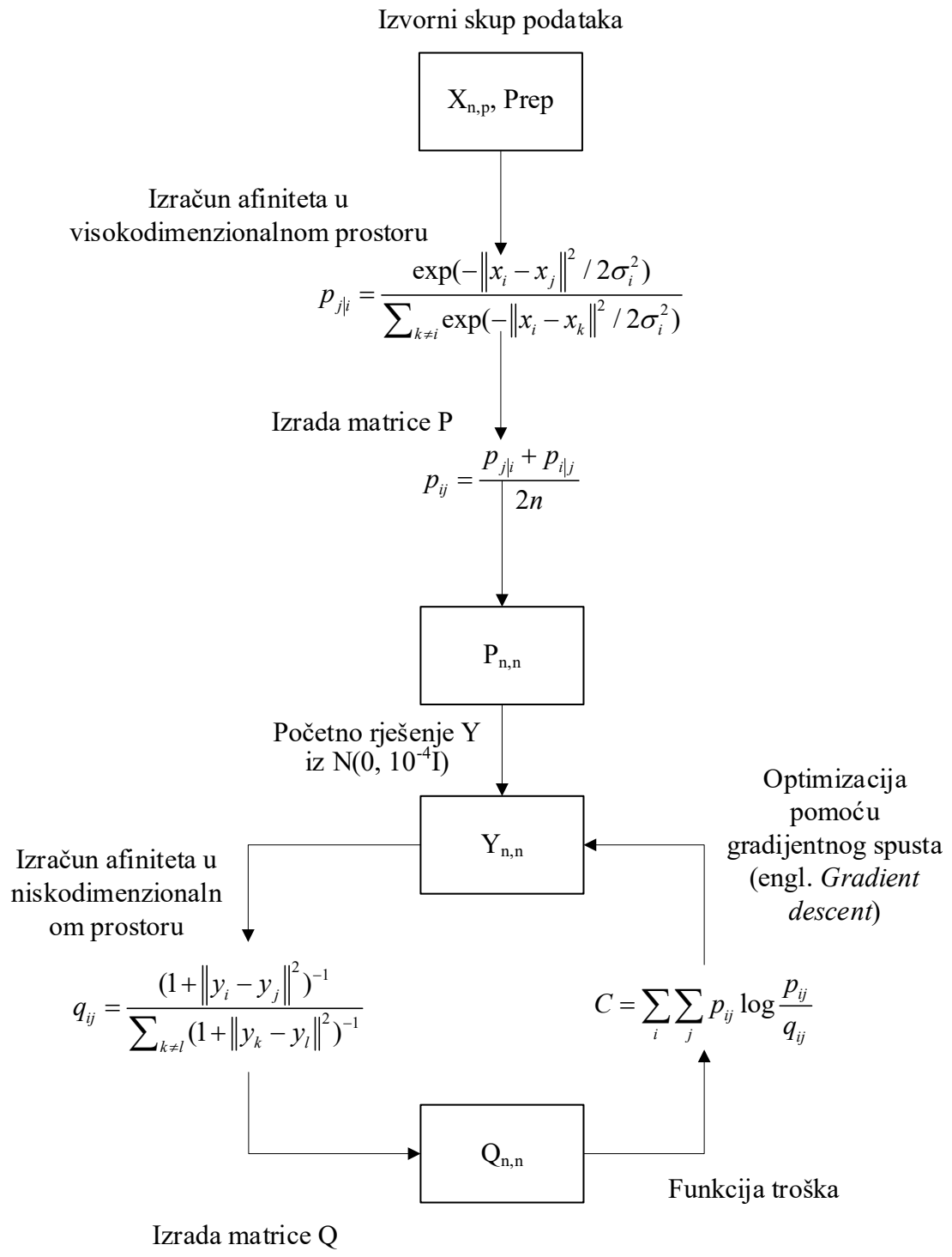
U danom grafičkom primjeru [46] *X* označava matricu skupa podataka dimenzija (*n,p*), gdje je *n* broj uzoraka, a *p* broj značajki (dimenzija). Cilj *eigen* dekompozicije i SVD-a je pronaći način na koji se dana matrica *X* može transformirati i dekomponirati na način da se kasnije može rekonstruirati uz očuvanje maksimalne količine informacija, ali u prostoru smanjene dimenzionalnosti. Metode *eigen* dekompozicije i SVD-a se temelje isključivo na linearnoj algebri te zbog toga PCA ne mijenja podatke već omogućuje promatranje istih iz drugačije perspektive. Prethodno navedeno, predstavlja ključno svojstvo PCA metode koje ga razlikuje od alternativnih pristupa redukciji dimenzionalnosti koji započinju slučajnom reprezentacijom podataka u nižedimenzionalnom prostoru i zatim pokušavaju očuvati strukturu podataka. Među značajnijom prednošću PCA metode ističe se vrijeme izvođenja odnosno brzina, budući da se sve operacije svode na brze linearne izračune. Nadalje, PCA daje konzistentne rezultate, što znači da će, za isti skup podataka, uvijek generirati istu transformaciju, što nije slučaj kod nekih drugih metoda redukcije dimenzionalnosti. U primjeru korištenja SVD-a, ključno svojstvo je dijagonalna matrica Σ, koja sadrži singularne vrijednosti. Navedene vrijednosti predstavljaju važnost svake dimenzije u očuvanju informacija. U postupku redukcije dimenzionalnosti, odabire se prvih *r* najvećih singularnih vrijednosti iz matrice Σ, pri čemu *r* određuje ciljani broj dimenzija. Ovaj pristup omogućava prilagodljivo očuvanje informacija, jer se odabire broj dimenzija koje osiguravaju

očuvanje određene razine varijance podataka (npr. odabirom *r* tako da se izgubi maksimalno 40% ukupnih informacija). Metoda PCA se pokazala osobito korisnom u pretprocesiranju podataka prije primjene modela strojnog učenja, gdje može smanjiti redundantnost i poboljšati računalnu učinkovitost. [14-16, 46, 47]

3.3. T-distribuirano stohastičko ugrađivanje susjeda

Metoda t-SNE temelji se na transformaciji podataka iz višedimenzionalnog prostora u prostor niže dimenzionalnosti, gdje se nastoje održati relativne udaljenosti i sličnosti među podatkovnim točkama. Primjer, ako su u izvornom višedimenzionalnom prostoru dvije podatkovne točke smještene blizu jedna drugoj zbog visoke sličnosti, nakon smanjenja dimenzionalnosti te bi točke i dalje trebale ostati blizu u novom prostoru. S druge strane, podatkovne točke koje su inicijalno bile međusobno udaljene trebale bi zadržati tu udaljenost i nakon transformacije. Prethodno navedeno očuvanje strukture podataka ključno je za osiguravanje da vizualizirani skup podataka vjerno odražava stvarne odnose među podatcima, čime se omogućuje bolja interpretacija i razumijevanje kompleksnih uzoraka. Navedeni pristup čini t-SNE iznimno korisnim alatom za vizualizaciju složenih skupova podataka i otkrivanje skrivenih uzoraka unutar njih. [17, 48]

Metoda t-SNE temelji se na iterativnom postupku smanjenja dimenzionalnosti, što ga razlikuje od metoda poput PCA, koja se oslanja na linearnu transformaciju podataka. [14-17, 46]



Slika 21: t-SNE radni tijek (Izvor: [46])

Na prethodno prikazanom grafičkom prikazu je radni tijek metode t-SNE, koji se može podijeliti u nekoliko ključnih koraka[46]:

1. Ulazni podatci – metoda t-SNE prihvaća dvije vrste ulaznih podataka:

- $X_{n,m}$ predstavlja skup podataka prikazanu kao matricu
- Perpleksnost (engl. *Perp*), koja određuje balans između lokalne i globalne strukture podataka tijekom optimizacije. Perpleksnost definira koliko će susjednih točaka biti uzeto u obzir pri izgradnji distribucije sličnosti. Preporučeni raspon je između 5 i 50, pri čemu viša perpleksnost uzima u obzir veći broj susjednih točaka, dok niža perpleksnost naglašava lokalne odnose.

2. Izračun afiniteta u visokodimenzionalnom prostoru – korištenjem Gaussove distribucije definiraju se uvjetne vjerojatnosti p_{ji}, koje opisuju koliko je podatkovna točka *xj* slična podatkovnoj točki x_i. Navedene vjerojatnosti se zatim normaliziraju kako bi se dobila matrica sličnosti $P_{n,n}$. Perpleksnost se koristi kako bi se odredila standardna devijacija σ_i Gaussove distribucije, gdje se optimalna vrijednost određuje binarnom pretragom.
3. Početna inicijalizacija niskodimenzionalnih reprezentacija – Generira se početna niskodimenzionalna matrica $Y_{n,n}$ uzorkovanjem iz normalne distribucije $N(0, 10^{-4}I)$.
4. Izračun afiniteta u niskodimenzionalnom prostoru odnosno vjerojatnosti q_{ij}, koje predstavljaju odnose među podatkovnim točkama u smanjenom prostoru. Time se osigurava očuvanje relativne udaljenosti i sličnosti među podatkovnim točkama iz izvornog prostora.
5. Optimizacija pomoću gradijentnog spusta (engl. *Gradient descent*) – Funkcija troška (Kullback-Leibler divergencija[49]) minimizira se iterativnim postupkom optimizacije, gdje se minimizira razlika između visokodimenzionalnih i niskodimenzionalnih sličnosti.

Zaključno, metoda t-SNE je iznimno učinkovita metoda za vizualizaciju visokodimenzionalnih podataka, posebno u kontekstu otkrivanja skrivenih struktura i obrazaca unutar kompleksnih skupova podataka.[17, 46, 48] Međutim, iako se može primijeniti i kao tehnika redukcije dimenzionalnosti u kontekstu prediktivnog modeliranja, njezina praktična upotreba u tom segmentu ograničena je računalnom zahtjevnošću i osjetljivošću na hiperparametre, poput perpleksnosti. Na složenijim podatkovnim skupovima, metoda t-SNE često dovodi do suboptimalnih rezultata u

prediktivnom modeliranju, a njeno vrijeme izvršavanja može biti znatno duže u usporedbi s alternativnim metodama, poput metode PCA ili autoenkoderskih neuronskih mreža.

U okviru ovog istraživanja, metoda t-SNE je korištena za redukciju dimenzionalnosti u svrhu poboljšanja prediktivne preciznosti modela, pri čemu su rezultati analizirani kroz fazu Implementacije unutar prilagođenog CRISP-DM metodološkog okvira. Dobiveni rezultati potvrđuju da, iako metoda t-SNE može pružiti korisne uvide u strukturu podataka, primjena metode t-SNE u kontekstu prediktivnog modeliranja zahtijeva pažljivo balansiranje između računalne učinkovitosti i točnosti modela, obzirom kako je za određene zadatke redukcije dimenzionalnosti njeno izvršavanje trajalo više od 3 sata.

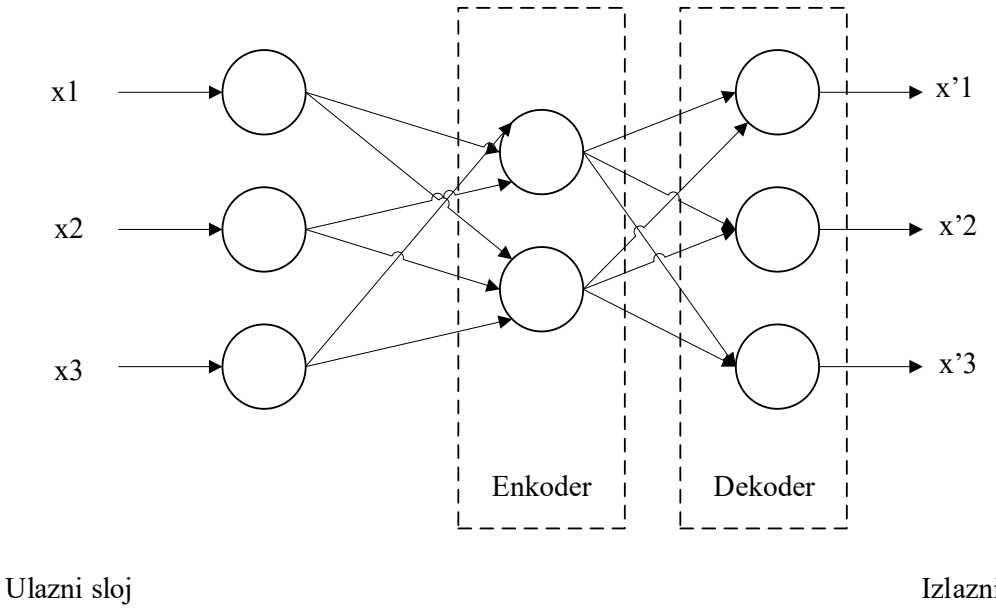
3.4. Uniformna aproksimacija mnogostrukosti i projekcija za redukciju dimenzionalnosti

Teorijske osnove metode UMAP prvenstveno se temelje na teoriji mnogostrukosti (engl. *manifold theory*) i topološkoj analizi podataka (engl. *topological data analysis*). Metoda UMAP koristi lokalne aproksimacije mnogostrukosti i spaja ih u jedinstvenu topološku reprezentaciju podataka visokih dimenzija. Za dobivenu niskodimenzionalnu reprezentaciju podataka provodi se sličan postupak kako bi se konstruirala ekvivalentna topološka struktura. Cilj metode UMAP je zatim optimizirati raspored podataka u niskodimenzionalnom prostoru minimizacijom unakrsne entropije između dviju topoloških reprezentacija. Proces konstrukcije topološke reprezentacije temelji se na dvije ključne faze: prvo se aproksimira mnogostrukost na kojoj podatci leže, a zatim se konstruira neizraziti (engl. *fuzzy*) simplicijalni skup koji opisuje aproksimiranu mnogostrukost. U daljnjim koracima metode UMAP razmatra se način konstrukcije *fuzzy* simplicijalnog skupa povezanog s niskodimenzionalnom reprezentacijom podataka (gdje se mnogostrukost svodi na standardni R^d) te se definira optimizacijska funkcija koja minimizira razliku između dviju topoloških reprezentacija. Prva faza UMAP algoritma uključuje aproksimaciju mnogostrukosti na kojoj se podatci nalaze. Ukoliko nije unaprijed poznata geometrijska struktura mnogostrukosti, potrebno ju je procijeniti iz podataka. Ključno je osigurati pretpostavku da su podatci ravnomjerno distribuirani po

mnogostrukosti. Kako bi se omogućilo ujednačeno predočavanje strukture podataka, UMAP normalizira udaljenosti među podatkovnim točkama skaliranjem prema udaljenosti do *k*-tog najbližeg susjeda. Sljedeća faza uključuje konstrukciju *fuzzy* topološke reprezentacije podataka. Ovaj proces zahtijeva transformaciju podataka iz metričkog prostora u neizraziti (engl. *fuzzy*) simplicijalni skup, čime se omogućuje objedinjavanje lokalnih metričkih prostora u globalnu reprezentaciju podataka. Metoda spajanja koristi *fuzzy* uniju lokalnih metričkih prostora kako bi se osigurala konzistentna globalna struktura. Na kraju, optimizacija niskodimenzionalne reprezentacije temelji se na minimizaciji unakrsne entropije između *fuzzy* simplicijalnih skupova u izvornom i reduciranim prostorima. Budući da *fuzzy* simplicijalni skupovi mogu biti predstavljeni kao ponderirani grafovi, optimizacija rasporeda podataka u nižoj dimenziji temelji se na tehnikama poput optimizacije metodom gradijentnog spusta. Metoda UMAP se pokazala osobito korisnim u analizi velikih i složenih skupova podataka, osobito u biomedicinskim istraživanjima, obradi prirodnog jezika te analizi vizualnih podataka.[18]

3.5. Autoenkoderske neuronske mreže

Autoenkoderi su vrsta neuronske mreže koja je specijalizirana za nenadzirano učenje, prvenstveno korištena za smanjenje dimenzionalnosti podataka i rekonstrukciju ulaznih uzoraka. Ključna značajka autoenkodera je njihova sposobnost mapiranja ulaznih podataka u komprimiranu reprezentaciju i naknadna rekonstrukcija tih podataka u izlazni sloj.[19] Arhitektura autoenkodera prikazana je u nastavku.



Slika 22: Arhitektura autoenkodera (Izvor: [50])

Arhitektura autoenkodera sastoji se od tri osnovna dijela: enkodera, latentnog prostora i dekodera. Enkoder transformira visoko-dimenzionalne ulazne podatke u kompaktnu latentnu reprezentaciju, dok dekoder pokušava rekonstruirati izvornu strukturu podataka na temelju te reprezentacije. Latentni prostor predstavlja srednju fazu u kojoj se nalazi reducirana dimenzionalnost podataka, koja može služiti za različite svrhe poput smanjenja redundantnih informacija, ekstrakcije značajki te poboljšanja performansi modela u zadatcima klasifikacije i klasteriranja. Optimizacija autoenkodera temelji se na minimizaciji funkcije troška koja uspoređuje originalne ulazne podatke s rekonstruiranim izlazima, gdje se obično koristi mjera poput srednje kvadratne pogreške (engl. *mean squared error* , MSE). Učenje se odvija primjenom tehnika optimizacije poput gradijentnog spusta (engl. *Gradient descent*) i propagacije unazad (engl. *back-propagation*). Autoenkoderi se široko primjenjuju u raznim domenama, uključujući otkrivanje anomalija, smanjenje dimenzionalnosti podataka, generiranje sintetičkih uzoraka te poboljšanje kvalitete podataka. Fleksibilnost i sposobnost autoenkodera u nenadziranom učenju čine ih često korištenom i popularnom metodom u području analize podataka. [19]

Zaključno, na temelju proučene literature i napisanih teorijskih osnova o metodama PCA, t-SNE, UMAP i Autoenkoderskim mrežama, autor ovo rada izvlači sljedeće zaključke: Metoda PCA je optimalna za reduciranje dimenzionalnosti pri

očuvanju globalnih struktura, metoda t-SNE, primarno omogućuje detaljnu vizualizaciju klasterskih struktura, metoda UMAP najčešće se koristi za nelinearnu redukciju dimenzionalnosti i vizualizaciju podataka, dok autoenkoderi kroz proces enkodiranja i dekodiranja podataka, tijekom treniranja, uče komprimiranu reprezentaciju ulaznih podataka u skrivenome sloju, gdje eliminiraju šum i ekstrahiraju ključne značajke. [14-19, 46-48]

Nastavno na zaključak, unutar ovog istraživanja, primjenom prilagođenog CRISP-DM metodološkog okvira, u fazi Implementacije, podfazi Analize generalizacijskih sposobnosti, provesti će se usporedba korištenih, standardnih i popularnih metoda redukcije dimenzionalnosti, uključujući metodu PCA, t-SNE, UMAP te autoenkoderske neuronske mreže, s vlastito razvijenim metodama za redukciju dimenzionalnosti. Cilj navedene usporedbe jest evaluacija učinkovitosti različitih pristupa u smanjenju dimenzionalnosti podataka, pri čemu će se posebna pažnja posvetiti njihovom utjecaju na, prvom po važnosti parametru, vrijeme izvršavanja odnosno brzini, zatim prediktivnoj preciznosti te sposobnost metoda za generalizaciju na nepoznatim podatcima.

U sljedećem poglavlju opisan je vlastito razvijeni algoritam latencije te razvijene metode pretprocesiranja podataka i redukcije dimenzionalnosti. Nadalje, predstavljeni su principi na kojima se temelji razvijeni algoritam latencije i navedene metode, s naglaskom na njihovu primjenu u kontekstu smanjivanja latencije (vremena izvršavanja), učinkovitosti u pretprocesiranju velikih količina podataka te smanjenja korištenja računalnih resursa uz očuvanje prediktivne preciznosti modela strojnog učenja.

4. RAZVIJENI AUTOMATIZIRANI ALGORITAM LATENCIJE

Kako bi se unaprijedila učinkovitost te automatizirao razvoj modela strojnog učenja uslijed eksponencijalnog rasta količine podataka, razvijen je automatizirani algoritam latencije kako bi se postigao cilj ovog rada – minimizacija vremena inferencije odnosno postizanje više brzine uz neznatne gubitke preciznosti u procesima predikcije u produkcijskom okružju.

Tradicionalni modeli strojnog učenja zahtijevaju veću računalnu snagu i duže inferencijsko vrijeme izvršavanja kada se primjenjuju nad visokodimenzionalnim podatcima. Kako bi se smanjila latencija nad visokodimenzionalnim podatcima, razvijen je automatizirani algoritam kao cjevovod, koji optimizira ključne faze modeliranja – od pripreme podataka i redukcije dimenzionalnosti do treniranja modela i predikcije, sve s ciljem postizanja najkraćeg mogućeg vremena analize bez značajnog gubitka u prediktivnoj preciznosti. U nastavku se uvodi u pregled teorijske i metodološke osnove automatiziranog algoritma latencije

4.1. Teorijska i metodološka osnova automatiziranog algoritma latencije

Pojam automatiziran, u nazivu, dodatno naglašava da algoritam ne zahtijeva ručnu intervenciju domenskih stručnjaka i može dinamički prilagoditi parametre modela, osiguravajući optimalan balans između brzine izvršavanja i točnosti predikcije. Nadalje, razvijeni automatizirani algoritam omogućava kontinuirani priljev podataka, gdje je ključno automatizirano procesiranje podataka i ažuriranje modela u stvarnom vremenu, što je slučaj u produkcijskim okružjima.

Razvijene metode pretprocesiranja podataka obuhvaćaju automatizirani tijek redoslijednih koraka odnosno cjevovod pripreme podataka od otkrivanja i obrade nedostajućih vrijednosti, uklanjanja dupliciranih vrijednosti, manipulacije tekstualnim podatcima, kodiranje kategorija, transformacije, skaliranja do ujednačavanja treniranih, testnih i validacijskih skupova podataka prije treniranja i testiranja modela.

Automatizirani cjevovod pretprocesiranja podataka znatno smanjuje vrijeme izvršavanja pri predikciji jer većinu potrebnih transformacija obavlja prije treniranja i testiranja modela, tako da pri samom predviđanju nema usporavanja zbog procesiranja podatka. Automatizirani cjevovod pretprocesiranja podataka na taj način ubrzava predikciju u *MoziaisMl* sustavu i smanjuje ukupnu latenciju (vrijeme inferencije).

U produkcijskim okružjima pretprocesiranje podataka može postati glavni ograničavajući faktor brzine izvršavanja, čak dovodeći do neiskorištenosti modela strojnog učenja ako podatci nisu pročišćeni, transformirani, kodirani i skalirani.[51]

Razvijena metoda redukcije dimenzionalnosti *Glasanja* je metoda odabira značajki koja koristi metode za klasifikacijske ili regresijske zadatke *RandomForest (RF)*, *XGBoost (XGB)*, *Mutual Information (MI)* i *Recursive feature elimination (RFE)* istovremeno te bira one značajke koje su konzistentno rangirane, po važnosti, visoko. Svaka od navedenih metoda izdvoji svoj skup najvažnijih značajki, nakon čega se za svaku značajku broji koliko se puta pojavila na tim listama. Konačnim glasovanjem odabrani skup čine one značajke koje imaju najviše glasova (pojavljivanja) – značajke koje većina metoda smatra važnima bivaju zadržane. Ovakav pristup ansambla (engl. *ensemble*) rezultira robusnim skupom značajki: spaja različite perspektive odabira, smanjujući pristranosti pojedinačnih metoda. Također, vlastito razvijena metoda redukcije dimenzionalnosti *Glasanja* daje stabilniji i pouzdaniji skup značajki – više metoda „odobrava“ svaku odabranu značajku te time je veća vjerojatnost da su odabrane značajke uistinu značajne za predikcijsku preciznost.

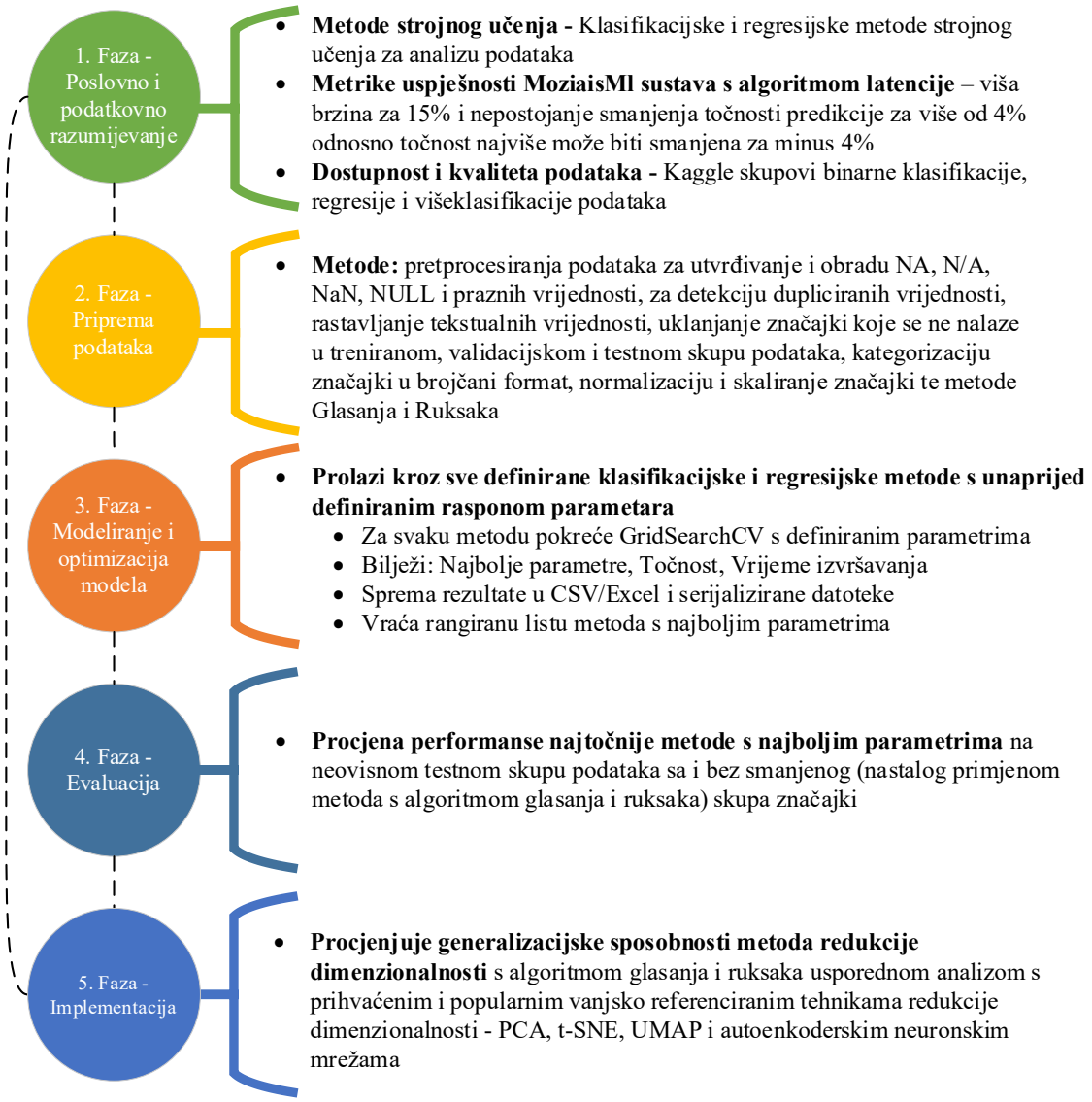
Razvijena metoda redukcije dimenzionalnosti *Glasanja* zadržava samo one značajke koje provjereno nose informaciju (podržane iz više različitih metoda), gdje se postiže, u najgorem slučaju, minimalan gubitak u prediktivnoj preciznosti unatoč reduciranju dimenzionalnosti. Manje ulaznih značajki znači i manje izračuna prilikom predikcije, što skraćuje inferencijsko vrijeme izvođenja.

Zatim, razvijena metoda redukcije dimenzionalnosti *Ruksak* rješava problem odabira značajki kao optimizacijski zadatak s ograničenjima na prilagođeni način, prema izvornom algoritmu problema ruksaka[52, 53]. Svakoj značajki dodjeljuje se vrijednost značajnosti i varijanca kao ograničenje za ukupnu težinu značajki koja ne smije prelaziti zadani kapacitet, težine, ruksaka. Zatim se odabire podskup značajki koji maksimizira

ukupnu vrijednost unutar dopuštene težine kapaciteta ruksaka. Cilj razvijene metode redukcije dimenzionalnosti *Ruksaka* je zadržati što informativnije značajke, ali istovremeno držati ukupnu „težinu“ skupa značajki ispod zadane granice – maksimalne težine ruksaka.

Razvijena metoda redukcije dimenzionalnosti *Ruksaka* daje optimiziran skup značajki u odnosu na zadani parametar maksimalne težine koji smiju stati u ruksak. Za razliku od rangiranja po značajnosti i odabira prvih top N značajki, pristup razvijene metode redukcije dimenzionalnosti *Ruksaka* uzima u obzir i težinu (varijancu) značajki – dvije značajke srednje važnosti zajedno daju više informacija nego jedna izrazito važna značajka gdje razvijena metoda *Ruksaka* identificira takvu kombinaciju, dvije značajke srednje važnosti, da ukupno donosi veću korist unutar maksimalne dopuštene težine ruksaka. Time se redundantni prostor pretraživanja smanjuje fokusiranjem na najbolje omjere korisnosti (značajnosti) i težine (varijance), što je bitno za inferencijsko vrijeme izvođenja gdje velika dimenzionalnost povećava vrijeme pretrage i smanjuje brzinu.

Rezultat istraživanja ovog rada potvrđuje kako model s odabranim značajkama metode *Ruksaka* postiže višu točnost i smanjenju grešku u odnosu na model s punim skupom značajki, uz daleko manju potrošnju vremena inferencije. S prethodno navedenim, razvijena metoda redukcije dimenzionalnosti *Ruksaka* ispunjava cilj minimalnih gubitaka (čak ih ni nema) u preciznosti uz maksimalno moguće ubrzanje: model radi brže, a i preciznost je viša. Na slici, u nastavku, prikazana je prilagođena verzija CRISP-DM metodološkog okvira analitičkog informacijskog sustava temeljenog na metodama strojnog učenja u produkcijskom okružju s algoritmom latencije.

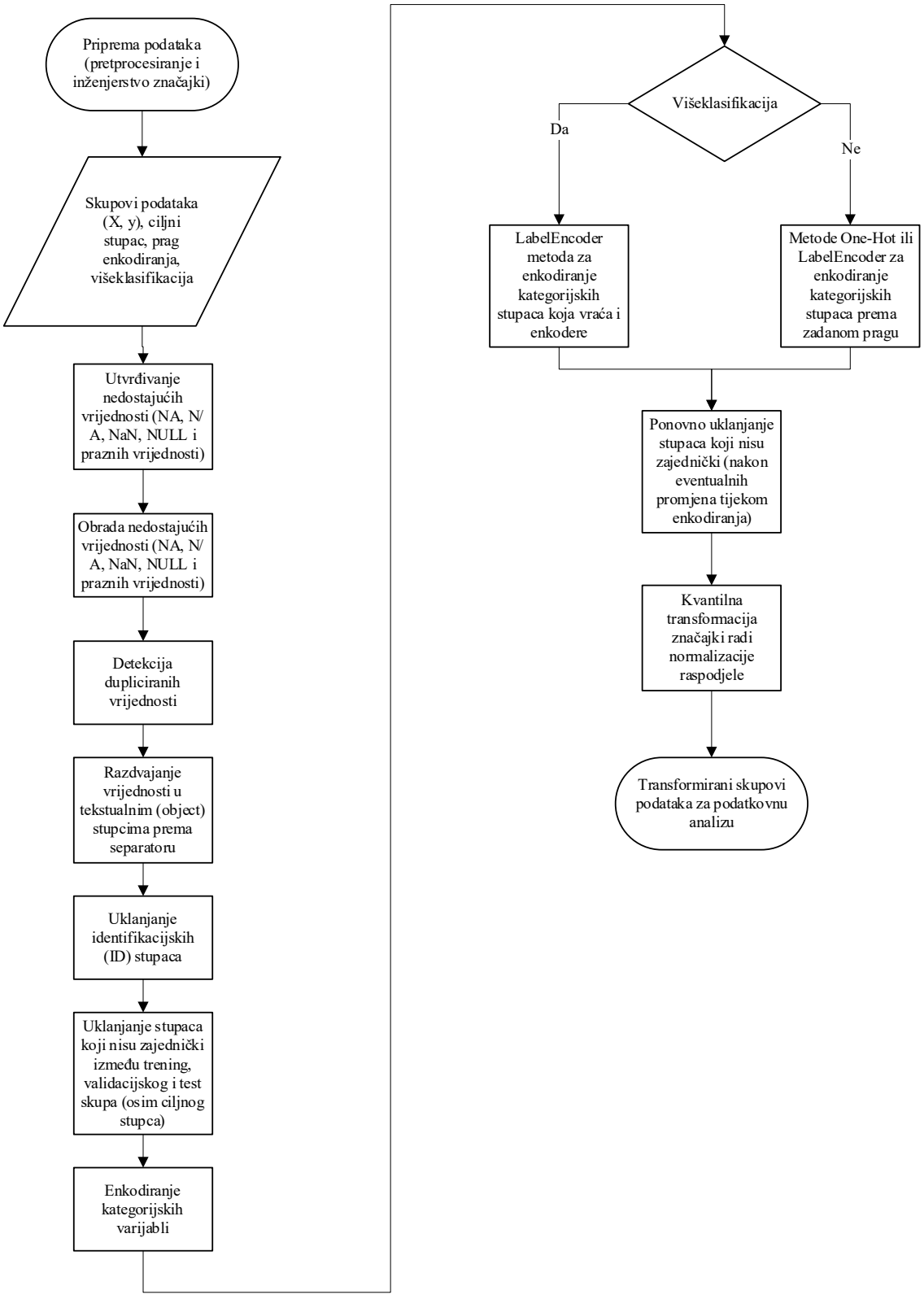


Slika 23: Prilagođena verzija CRISP-DM metodološkog okvira analitičkog informacijskog sustava temeljenog na metodama strojnog učenja (MoziaisMl sustava) u produkcijskom okružju (prilagođeno prema: [11, 21, 22, 54])

MoziaisMl sustav u prethodno prikazanoj prilagođenoj verziji CRISP-DM metodološkog okvira temelji se na modularnom, višefaznom konceptualnom modelu koji omogućuje potpunu automatizaciju procesa razvoja modela strojnog učenja za predikciju na reduciranom dimenzionalnom skupu podataka – od početnog procesiranja sirovih podataka, modeliranja/optimizacije modela do reducirane selekcije značajki i predikcije na reduciranom skupu značajki. Sustav je strukturiran u tri osnovne funkcionalne komponente: **Priprema podataka** (pretprocesiranje i inženjerstvo značajki), **Modeliranje/optimizacija modela** i **Redukcija dimenzionalnosti značajki**, koje zajedno čine koherentnu arhitekturu dizajniranu za poboljšanje predikcijske točnosti i

minimizaciju latencije (inferencijskog vremena izvršavanja) odnosno postizanja brzine unutar klasifikacijskih i regresijskih skupova podataka.

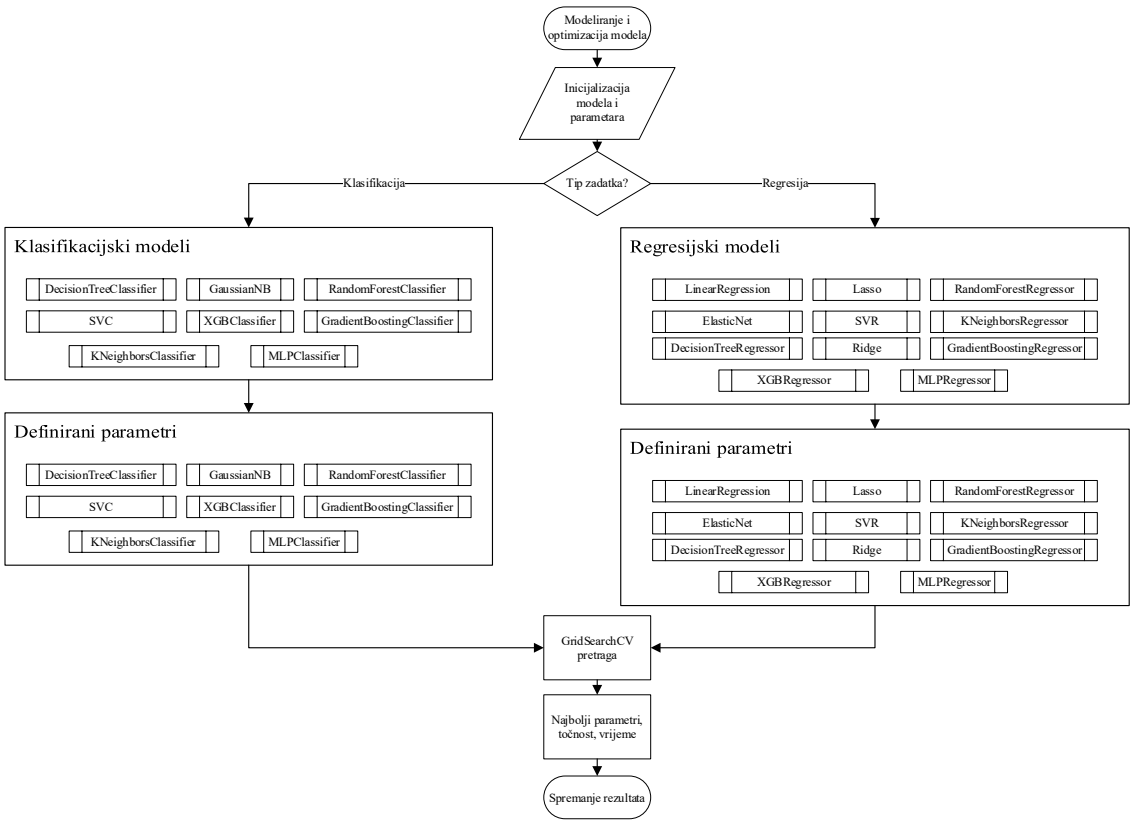
Faza pripreme podataka (pretprocesiranje i inženjerstvo značajki), na slici u nastavku ilustrira automatiziran pristup pretprocesiranju i inženjerstvu značajki unutar *MoziaisMl* sustava, čime se značajno nadilaze koraci definirani standardnim CRISP-DM metodološkim okvirom. Prikazana faza ima ključnu ulogu u osiguravanju kvalitete ulaznih podataka, eliminaciji izvora potencijalne pristranosti te optimizaciji podataka za kasnije faze modeliranja, optimizacije i evaluacije modela.



Slika 24: Faza pripreme podataka (pretprocesiranje i inženjerstvo značajki)

Prethodna slika ilustrira kako faza pripreme podataka u *MoziaisMl* sustavu uvodi visoki stupanj automatizacije, metodološku konzistenciju, repetitivnost i analitičku robusnost, nadilazeći korake standardne CRISP-DM metodologije. Uvođenjem vlastitih metoda za identifikaciju, transformaciju i enkodiranje podataka te kvantitativne stabilizacije distribucije, značajno se poboljšava kvaliteta ulaznih značajki, čime se stvara temelj za pouzdaniju, robusniju i generalizabilniju izgradnju modela u kasnijim fazama prilagođene CRISP-DM metodologije.

Faza modeliranja i optimizacije modela, kako je prikazana na slici u nastavku, metodološki se oslanja na princip automatizacije odabira i evaluacije modela strojnog učenja, čime se ostvaruje temeljna funkcionalnost *AutoML* pristupa u MoziaisMl sustavu. Ova faza započinje inicijalizacijom modela i pripadajućih hiperparametara, uz eksplicitno razgraničenje tipa zadatka na klasifikacijski ili regresijski, što omogućuje dinamičku prilagodbu skupa dostupnih algoritama.

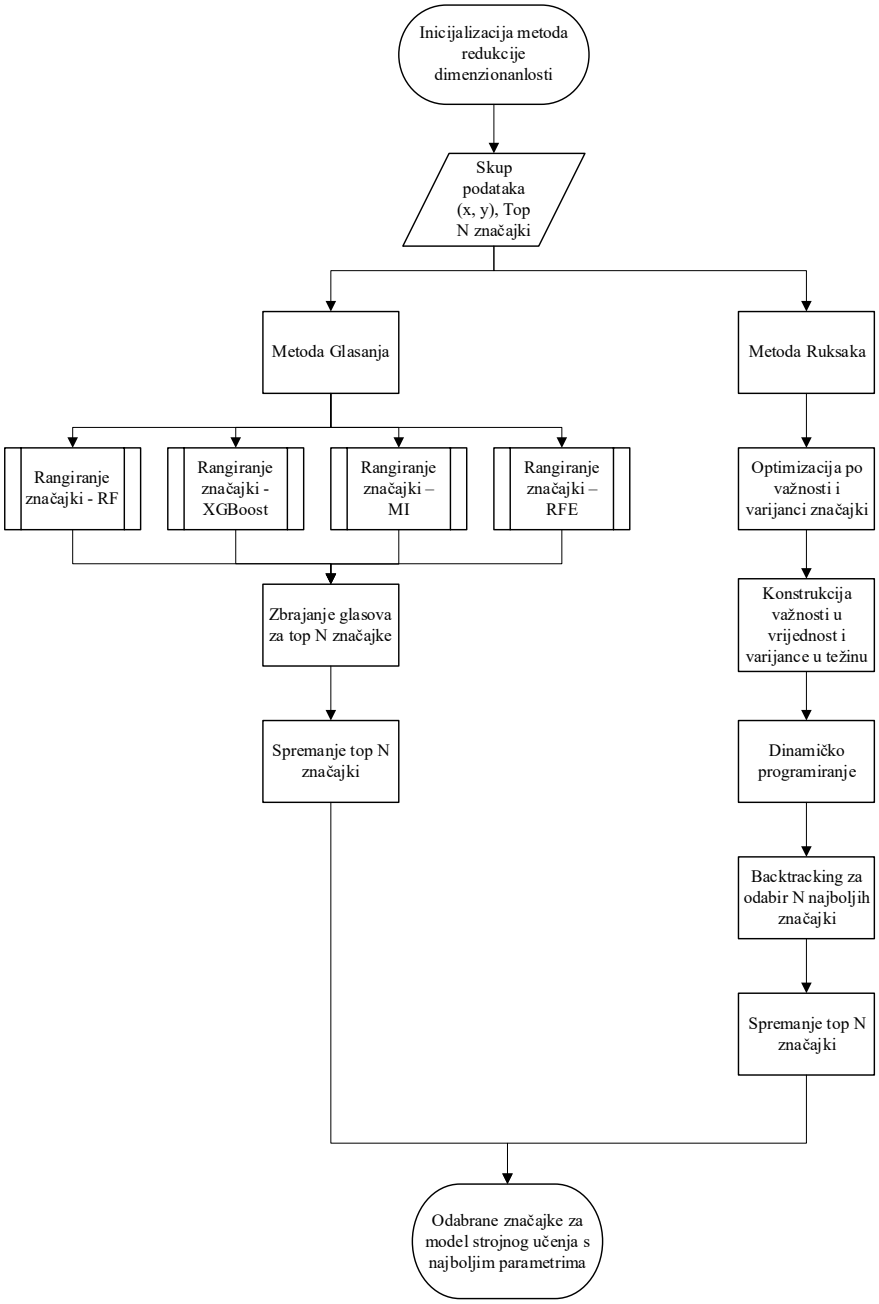


Slika 25: Faza modeliranja i optimizacije modela

Za svaki od navedenih modela/metoda strojnog učenja definirani su skupovi hiperparametara koji se testiraju primjenom *GridSearchCV* metode. Prethodno navedena

metoda provodi iscrpnu pretragu kroz prostor parametara s unakrsnom validacijom čime se osigurava robusnost odabira modela te minimizira rizik prenaučenosti.[31] Tijekom svake iteracije pretrage, sustav bilježi relevantne metrike performansi, među kojima se posebno ističu: točnost klasifikacije, srednje kvadratno odstupanje (RMSE) za regresijske zadatke, kao i vrijeme treniranja. U kontekstu standardnog CRISP-DM metodološkog okvira, predložena faza nadilazi tradicionalni pristup modeliranja i optimizacije modela, koji se oslanja na jednu metriku (primjer, točnost), integrirajući umjesto toga višekriterijsku optimizaciju gdje se uzima u obzir i metrički pokazatelj vremena treniranja. Po završetku faze modeliranja i optimizacije modela, *MoziaisMl* sustav automatski identificira konfiguraciju modela s optimalnim odnosom između točnosti i vremenske učinkovitosti te istu serijalizira i sprema u format prikladan za produkcijsku implementaciju i kasniju upotrebu.

Metode redukcije dimenzionalnosti, prikazane slikom u nastavku, integrirane su u prilagođeni CRISP-DM metodološki okvir *MoziaisMl* sustava i oslikavaju dvofazni pristup selekciji značajki: metodu *Glasanja* i metodu *Ruksaka*. Razvijene metode osmišljene su s ciljem smanjenja dimenzionalnosti ulaznih podataka bez smanjivanja prediktivne učinkovitosti modela strojnog učenja, uz istovremeno smanjenje latencije izvršavanja u produkcijskim okružjima.



Slika 26: Metode redukcije dimenzionalnosti

Obje metode, metoda *Glasanja* i metoda *Ruksaka*, završavaju istom točkom – generiranjem skupa odabranih značajki koji se koristi za predikciju modela strojnog učenja s prethodno odabranim optimiziranim parametrima. Time se omogućuje vremenski učinkovitija prediktivna analiza podataka. Prikazani višekriterijski pristup selekciji značajki ujedno predstavlja znanstveni doprinos u odnosu na klasične metode redukcije dimenzionalnosti, jer objedinjuje optimizacijski učinak s funkcijom cilja, gdje

dobivena optimizacijska vrijednost povećava interpretabilnost modela u različitim poslovnim domenama primjene.

4.2. Konceptualni model i arhitektura MoziaisMl sustava

MoziaisMl sustav implementira višeslojni konceptualni model temeljen na principima automatizacije, reproducibilnosti i optimizacije vremenske složenosti izvođenja modela strojnog učenja nad klasifikacijskim i regresijskim skupovima podataka. Svaka komponenta sustava obavlja specifičan skup zadataka:

1. **Priprema podataka (pretprocesiranje i inženjerstvo značajki)** – komponenta (klasa) za pripremu podataka
    1. Ova komponenta predstavlja prvi korak procesa u *MoziaisMl* sustavu te automatizira klasične zadatke pretprocesiranja podataka koristeći metode koje obuhvaćaju:
        a. **Utvrđivanje (detekciju) i obradu (imputaciju) nedostajućih vrijednosti (*NA, N/A, NaN, NULL* i praznih vrijednosti)**, uz dodatno generiranje indikatorskih značajki (*_was_missing*), čime se omogućuje kvantifikacija informacija koje bi inače bile izgubljene.
        b. **Detekciju dupliciranih vrijednosti**, čime se osigurava integritet i jedinstvenost zapisa.
        c. **Inženjerstvo značajki putem metode za razdvajanje tekstualnih vrijednosti,** koja vrši dekonstrukciju složenih tekstualnih atributa u više stupaca, čime se postiže interpretabilnost i mogućnost enkodiranja stupaca te uključivanje takvih stupaca u značajke za proces razvoja modela strojnog učenja i predikcije.
        d. **Uklanjanje identifikacijskih i nesinkroniziranih stupaca** kako bi se osigurala dosljednost između treniranog, validacijskog i testnog skupa podataka.
        e. **Automatsko enkodiranje kategorijskih varijabli**, koristeći hibridni pristup praga koji ovisno o broju jedinstvenih vrijednosti primjenjuje

Label Encoding ili *One-Hot Encoding*, čime se kontrolira razina dimenzionalnosti skupa podataka.

f. **Kvantilnu transformaciju**, koja podatke mapira na uniformnu distribuciju radi ublažavanja utjecaja odstupanja (engl. *outliers*).

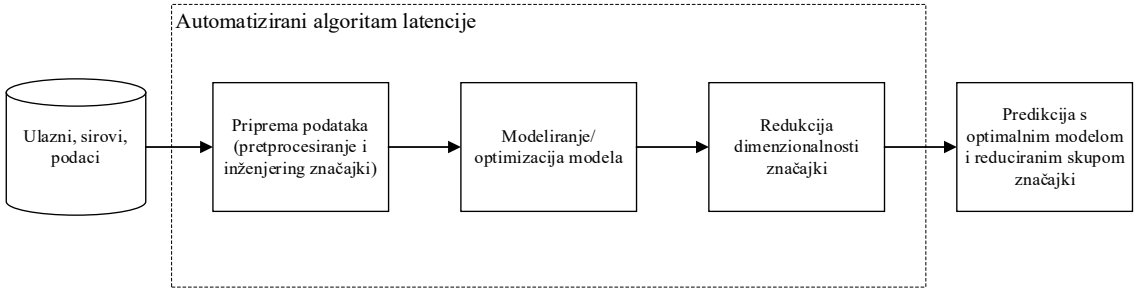
2. Ova komponenta dovodi do čistog, transformiranog i standardiziranog skupa podataka, spremnog za postupke selekcije i modeliranja, uz značajno smanjenu latenciju u odnosu na ručne pristupe.

2. **Modeliranje/optimizacija modela –** komponenta (klasa) za odabir optimalnog modela
    1. Ova komponenta omogućuje evaluaciju i optimizaciju različitih klasifikacijskih i regresijskih modela:
        a. Metoda u komponenti koristi postupak *GridSearchCV*[31] u kombinaciji s unakrsnom validacijom kako bi se pronašla optimalna konfiguracija hiperparametara za zadani model.
        b. Klase, unutar komponente, *clf_models_and_params()* i *reg_models_and_params()* definiraju modele (metode) i vrijednosti parametara za klasifikacijske i regresijske zadatke.
    2. Posebnost ove komponente je mjerenje i spremanje odnosno arhiviranje vremena treniranja, što omogućuje da vrijeme treniranja bude sastavni dio evaluacije modela u modeliranju i optimizaciji za izbor najboljeg modela, čime ovaj rad odstupa od tradicionalnog vrednovanja modela strojnog učenja isključivo s mjerom postotka greške ili točnosti.
3. **Redukcija dimenzionalnosti značajki –** komponenta (klasa) za selekciju značajki
    1. U ovoj komponenti primjenjuju se vlastito razvijene metode za ocjenjivanje značajnosti ulaznih varijabli skupa podataka:
        a. Metoda *Glasanja* implementira metode *RandomForest (RF)[25, 26]*, *XGBoost (XGB)[27]*, *Mutual Information (MI)[28, 29]* i *Recursive feature elimination (RFE)[30]* za procjenu značajki temeljem heuristika i modelskih karakteristika značajnosti (atribut metode: *feature_importances_*) ili koeficijenta (atribut metode: *coef_*).

b. Metoda *Glasanja* agregira rezultate iz više metoda u konsenzus značajki, čime se postiže robusniji odabir.
c. Metoda *Ruksaka* predstavlja napredniji pristup u odnosu na metodu *Glasanja* u kojem se koristi optimizacijski model ruksaka koji uzima u obzir značajnost značajke (vrijednost) i njezinu varijancu (trošak), uz definiran kapacitet ruksaka prema složenosti modela odnosno maksimalno dopuštenom trošku.

2. Rezultat ove komponente je smanjeni skup značajki optimiziran za brže vrijeme izvođenja modela strojnog učenja (niskolatentna evaluacija) bez značajnog kompromisa u točnosti predikcije.

Cjelokupni proces cjevovodne arhitekture *MoziaisMl* sustava slijedi sekvencijalni i komponenti podatkovni tok:



Slika 27: Cjelokupni proces cjevovodne arhitekture MoziaisMl sustava

Arhitektura *MoziaisMl* sustava omogućuje jednostavno praćenje, replikaciju i proširivost sustava s dodatnim komponentama u budućnosti. Korištenjem internih datoteka (CSV, Excel) i serijalizacije modela putem modula *pickle* omogućuje se transparentnost i interpretabilnost procesa, što omogućuje *MoziaisMl* sustavu ispunjavanje visokih zahtjeva za nadzorom (engl. *audit*) i objašnjivošću (engl. *explainability*).

Razvijeni *MoziaisMl* sustav predstavlja primjer integriranog, automatiziranog okvira koji kombinira metodološku preciznost, inženjerstvo značajki temeljen na heuristikama, višekriterijsku optimizaciju i sustavnu kontrolu nad latentnim troškovima (inferencijskim vremenom izvršavanja) odnosno nad postizanjem brzine bez značajnog kompromisa u predikcijskoj preciznosti. Njegova algoritamska prilagodljivost različitim

domenama predikcije – klasifikacijskim i regresijskim zadatcima, uz arhitekturu temeljenu na modularnim komponentama, čini *MoziaisMl* sustav pogodnim za široku primjenu u domenama s velikim količinama podataka i potrebom za brzim i pouzdanim modelima strojnog učenja.

Zaključno, razvijeni *MoziaisMl* sustav s automatiziranim algoritam latencije odražava osnovnu svrhu ovog rada – smanjenje inferencijskog vremena izvršavanja u predikcijskom procesu strojnog učenja kroz automatizirane razvijene metode pretprocesiranja podataka i razvijene metode redukcije dimenzionalnosti *Glasanja* i *Ruksaka*, uz minimalne gubitke u prediktivnoj preciznosti.

U sljedećem poglavlju predstavljen je teorijski okvir prilagođene CRISP-DM metodologije te, unutar automatiziranog algoritma latencije, razvijene metode pretprocesiranja podataka i razvijene metode redukcije dimenzionalnosti.

5. METODOLOŠKI OKVIR RAZVITKA AUTOMATIZIRANOG SUSTAVA MOZIAISML

Poglavlje metodologija predstavlja razvijene metode redukcije dimenzionalnosti za produkcijsku primjenu unutar prilagođene verzije CRISP-DM metodološkog okvir za izrađeni analitički informacijski sustav, naziva *MoziaisMl*, temeljenog na metodama strojnog učenja u produkcijskom okružju. Poglavlje metodologija podijeljeno je na dva dijela – prvi dio predstavlja prilagođeni teorijski okvir metodologije istraživanja temeljen na CRISP-DM fazama i *Design Science Research* (DSR) metodologiji, dok drugi dio predstavlja prilagodbu CRISP-DM faza za DSR.

Svrha istraživanja je razvoj automatiziranog algoritma latencije odnosno razvoj metoda redukcije dimenzionalnosti *Glasanja* i *Ruksaka* za produkcijsku primjenu unutar prilagođene verzije CRISP-DM metodološkog okvira razvoja *MoziaisMl*-a. Razvijene metode *Glasanja* i *Ruksaka* prilagođavaju ulazni skup podataka kako bi se dobio izlazni skup podataka temeljem kojeg će metode strojnog učenja raditi brže uz neznatne gubitke u točnosti predviđanja.

5.1. Teorijski i metodološki okvir MoziaisMl sustava

MoziaisMl sustav se temelji na prilagođenoj verziji CRISP-DM[11, 21] i *Cross-Industry Standard Process for Machine Learning* (CRISP-ML)[22, 54] metodološkog okvira. Metodologije CRISP-DM i CRISP-ML predstavljaju okvir za organizaciju i provedbu projekata procesiranja podataka i strojnog učenja. Svaka faza nadograđuje prethodnu, osiguravajući sustavan i iterativan proces.[11, 21, 22]

U nastavku su prikazane i sažeto opisane faze koje su međusobno povezane, čime tvore koherentan i učinkovit pristup procesiranja podataka i razvoju modela strojnog učenja[11, 21, 22]:

1. **Poslovno razumijevanje,**
2. **Razumijevanje podataka,**
3. **Priprema podataka,**
4. **Modeliranje i optimizacija modela,**

5. **Evaluacija,**
6. **Implementacija.**

Faza **Poslovno razumijevanje** uključuje precizno definiranje poslovnog problema, koji se nastoji riješiti te detaljnu analizu zahtjeva, ciljeva i ograničenja projekta. Ključni cilj je osigurati usklađenost tehničkog rješenja s poslovnim potrebama. [11, 21, 22]

U fazi **Razumijevanja podataka** provodi se prikupljanje, istraživanje i analiza podataka koji će se koristiti u, trećoj i četvrtoj fazi, pripremi podataka i modeliranju. Identificiraju se potencijalni problemi, poput nedostajućih vrijednosti, odstupanja ili nekonzistentnosti u skupu podataka. [11, 21, 22]

U fazi **Pripreme podataka** provodi se inženjerstvo značajki, pretprocesiranje podataka s ciljem poboljšanja performansi modela i prilagodbe ulaznih podataka zahtjevima algoritama. [11, 21, 22]

U fazi **Modeliranja i optimizacije modela** odabiru se i optimiziraju algoritmi strojnog učenja koji će se koristiti za rješavanje problema. [11, 21, 22]

Fazom **Evaluacije**, nakon izgradnje modela, provodi se procjena modela pomoću relevantnih metričkih pokazatelja i tehnika validacije, poput križne validacije i testiranja na izdvojenom skupu podataka. Cilj je osigurati da model ispunjava postavljene ciljeve i pokazuje zadovoljavajuću robusnost i generalizacijsku sposobnost. [11, 21, 22]

U fazi **Implementacije** model se integrira u produkcijsko okružje gdje postaje operativan i omogućuje donošenje predikcija ili odluka temeljenih na analizi podataka. Nakon implementacije, radi se usporedna analiza učinkovitosti, vlastito razvijenih metoda redukcije dimenzionalnosti *Glasanja* i *Ruksaka*, u odnosu na prihvaćene i popularne tehnike redukcije dimenzionalnosti - PCA, t-SNE, UMAP i autoenkoderske neuronske mreže. Cilj ove usporedbe učinkovitosti jest procjena generalizacijskih sposobnosti vlastito razvijenih metoda *Glasanja* i *Ruksaka*. [11, 21, 22]

Prethodno navedene faze čini sveobuhvatan pristup istraživačkom procesu analize podataka omogućujući procesno i sustavno praćenje, u slučaju ovog rada, istraživanja i razvoja prilagođenog okvira *MoziaisMl* sustav. U sljedećem potpoglavlju predstavljena je prilagođena verzija CRISP-DM faza.

5.2. Prilagodba faza u MoziaisMl sustavu

Prilagodba CRISP-DM faza kreće od prve dvije faze, spajajući ih u 1 fazu. U metodološkom okviru na temelju dostupne literature (više u [11, 21, 22, 54]), faza Poslovno razumijevanje i Razumijevanje podataka postaju jedna i prva faza metodološkog okvira ovog rada.

5.2.1. Prilagodba faze poslovnog i podatkovnog razumijevanja unutar MoziaisMl sustava

Novostvorena i međusobno spojena, prva faza obuhvaća poslovno i podatkovno razumijevanje[11, 21, 22, 54], ključna je za izvedivost *MoziaisMl* sustava. Navedena faza predstavlja temeljni trenutak u cijelom procesu, jer omogućuje analitički pristup odlučivanju o primjeni algoritama strojnog učenja ovisno o kakvom se skupu podataka radi i što se predviđa. Sustavno se analizira i odlučuje o sljedećim bitnim čimbenicima metodološkog okvira (prilagođeno prema [11, 21, 22, 54]):

1. **Potrebne metode strojnog učenja** - Evaluacija potrebnih klasifikacijskih i regresijskih metoda strojnog učenja za analizu podataka.
2. **Definiranje metrika uspješnosti** – viša brzina za 15% i nepostojanje smanjenja točnosti predikcije za više od 4% odnosno točnost najviše može biti smanjena za minus 4%.
3. **Dostupnost i kvaliteta podataka** – Analiza raspoloživosti skupova podataka na Kaggle[55] platformi za klasifikacijske i regresijske skupove podataka.

5.2.2. Automatizirana faza inženjerstva značajki i metoda Glasanja i Ruksaka u MoziaisMl sustavu

Spajanjem prve i druge faze u jednu koja postaje prva faza, druga faza prilagođenog metodološkog okvira postaje Priprema podataka. U fazi Pripreme podataka provode se ključni zadatci pretprocesiranja podatka, koji uključujući odabir relevantnih podataka, čišćenje podataka, inženjerstvo značajki, transformacija podataka te redukcija dimenzionalnosti. Kako bi se osigurala učinkovitost i ponovljivost postupka te integracija okvira u produkcijsko okružje, razvio se automatizirani cjevovod pretprocesiranja podataka, naziva – *algoritam latencije*.

U drugoj fazi Priprema podataka razvijene su metode pretprocesiranja koje utvrđuju nepostojeće i prazne vrijednosti, razdvajaju složene tekstualne atribute, uklanjaju duplikate i nejednake atribute između treniranog, validacijskog i testnog skupa podataka, identificiraju atribute koji predstavljaju jedinstvena polja unutar skupa podataka te klasificiraju vrijednosti u kategorije. Nadalje, faza Priprema podataka sadrži i metodu za provođenje normalizacije podataka osiguravajući time konzistentnost za daljnju analizu i predikciju.

Također, u drugoj fazi Priprema podataka gdje se provode zadatci pretprocesiranja podataka razvijene su metode *Glasanja* i *Ruksaka.* Metoda *Ruksaka* je prilagođena verzija za redukciju dimenzionalnosti algoritma *Ruksaka* [52, 53].

5.2.2.1. Metoda pretprocesiranja podataka za utvrđivanje *NA, N/A, NaN, NULL* i praznih vrijednosti

Prva metoda pretprocesiranja podataka utvrđuje *NA, N/A, NaN, NULL* i prazne vrijednosti. Metoda je za razvijena za identifikaciju nestalih vrijednosti unutar ulaznog skupa podataka te za generiranje izvještaja i vizualizaciju distribucije tih vrijednosti. Metoda analizira podatke identificiranjem stupaca u kojima postoje podatci s nepostojećim vrijednostima, što je jedna od ključnih analiza u fazi Priprema podataka. Metoda prihvaća više parametara koji omogućuju prilagodbu analize. Parametar df_{temp} predstavlja skup podataka u kojem se provodi analiza. Parametar *Opis skupa podataka* je proizvoljan i omogućuje korisniku *MoziaisMl* sustava da specificira opis skupa podataka, čime se poboljšava interpretabilnost ispisa rezultata. Parametar *Tihi ispis* je logička varijabla (*Boolean*) koja kontrolira hoće li se generirati konzolni ispis rezultata detekcije, gdje *Tihi ispis=True* onemogućuje ispis, a *Tihi ispis=False* omogućuje detaljan prikaz rezultata. Parametar *Graf* je također logički parametar koji određuje hoće li se vizualizirati distribucija nestalih vrijednosti unutar stupaca skupa podataka. Na kraju povratna vrijednost metode je broj i lista imena stupaca koji sadrže *NA, N/A, NaN, NULL* i prazne vrijednosti odnosno nestale vrijednosti.

Metoda se sastoji se od 6 procesnih radnji, prikazanih u nastavku:

1. Ulazni podatci
    - Skup tabličnih podataka df_{temp}

- Opis skupa podataka: proizvoljan ulazni podatak
- Opcije:
    - Tihi ispis: *Boolean* vrijednost (Točno/Netočno) koja kontrolira ispis podataka. Točno predstavlja neprikazivanje ispisa.
    - Graf: *Boolean* vrijednost (Točno/Netočno) koja kontrolira ispis grafičkog prikaza rezultata.

2. Provjera ukupnog broja *NA, N/A, NaN, NULL* i praznih vrijednosti
    - Proces:
        - Za svaki element u skupu podataka df_{temp} provjerava je li *NA, N/A, NaN, NULL* i prazna vrijednost.
        - Zbrajaju se svi *NA, N/A, NaN, NULL* i prazna vrijednost elementi koristeći:

$$count\_missing = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(x_{ij}\ \ je\ NA, N/A, NaN, NULL, ""\right) \tag{4}$$

        - Ako je count_missing = 0, metoda prelazi na kraj jer nema *NA, N/A, NaN, NULL* i praznih vrijednosti
    - Izlaz: Ukupan broj *NA, N/A, NaN, NULL* i praznih vrijednosti (count_missing).
3. Identifikacija stupaca s *NA, N/A, NaN, NULL* i praznim vrijednostima
    - Proces
        - Iterira se kroz svaki stupac *k* u u skupu podataka $df_{temp.}$
        - Računa se broj *NA, N/A, NaN, NULL* i praznih vrijednosti u stupcu *k* koristeći:

$$col\_missing(k) = \sum_{i=1}^{n}\left(x_{ik}\ \ je\ NA, N/A, NaN, NULL, ""\right) \tag{5}$$

        - Ako *col_missing(k)* > 0, stupac *k* dodaje se u listu *columns_with_missing*
    - Izlaz: Lista stupaca s *NA, N/A, NaN, NULL* i praznim vrijednostima (*columns_with_missing*).

4. Ispis rezultata
    - Proces:
        - Ako je *Tihi ispis* = *Netočno*, ispisuje se:
            - Ukupan broj *NA, N/A, NaN, NULL* i praznih vrijednosti.
            - Broj *NA, N/A, NaN, NULL* i praznih vrijednosti u *columns_with_missing.*
        - Ako je dan opis skupa podataka, uključuje se u ispis.
    - Izlaz: Tekstualni izvještaj o *NA, N/A, NaN, NULL* i praznim vrijednostima (ako nije Tihi ispis).
5. Izrada grafičkog prikaza
    - Proces:
        - Ako je plot = Točno i postoji najmanje jedna *NA, N/A, NaN, NULL* ili prazna vrijednost:
            - Generira se stupičasti graf s osi:
                - x: Imena stupaca (columns_with_missing).
                - **y:** Broj *NA, N/A, NaN, NULL* i praznih vrijednosti u svakom stupcu (*col_missing(k)*).
    - Izlaz: Grafički prikaz *NA, N/A, NaN, NULL* i praznih vrijednosti.
6. Povratna vrijednost metode
    - Lista stupaca *NA, N/A, NaN, NULL* i praznih vrijednosti (*columns_with_missing*).
    - Lista se koristi za daljnju obradu podataka.

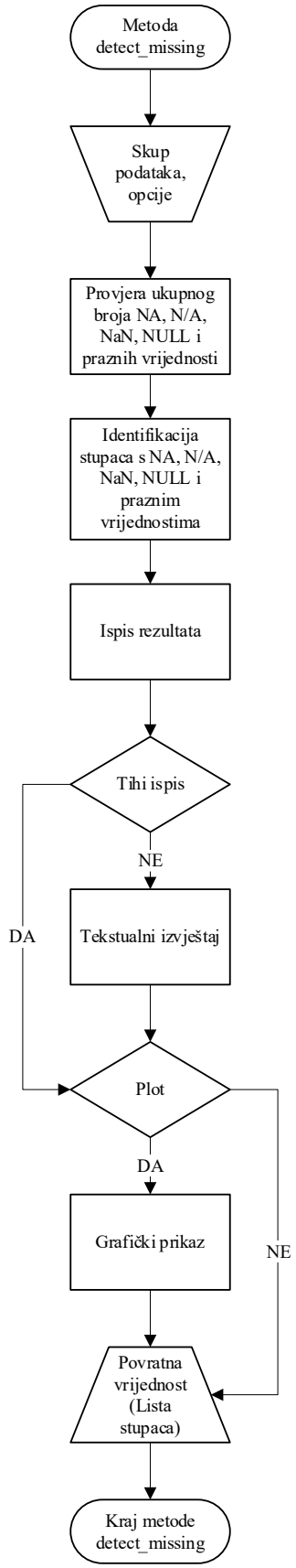
Primjer rada metode za utvrđivanje NA, N/A, NaN, NULL i praznih vrijednosti u skupu podataka:

1. Ulazni skup podataka (df_{temp}) i lista stupaca iz prethodne metode za utvrđivanje NA, N/A, NaN, NULL i praznih vrijednosti (*columns_with_missing*):

Id	Stambena površina u m2	Tip stambenog objekta	Površina podruma u m2	Širina kolnog ulaza
1	55	Stan	20	NA
2	245	Kuća u nizu	NULL	NA
3	180	Samostojeća kuća	50	6
4	155	Dvojna	NULL	10

2. Brojanje *NA, N/A, NaN, NULL* i praznih vrijednosti
   - *count_missing* = 4 (2 u Površina podruma u m2, 2 u Širina kolnog ulaza)
3. Identifikacija stupaca
   - *columns_with_missing* = ["Površina podruma u m2", " Širina kolnog ulaza "].
4. Ispis
   - Ukupno: 4, Površina podruma u m2: 2, Širina kolnog ulaza: 2
5. Vizualizacija
   - Stupčasti grafikon s 2 stupca - Površina podruma u m2: 2, Širina kolnog ulaza: 2
6. Rezultat
   - Vraćena lista: ["Površina podruma u m2", " Širina kolnog ulaza "]

Dijagram toka podataka metode za traženje *NA, N/A, NaN, NULL* i praznih vrijednosti prikazan je slikom u nastavku.



Slika 28: Traženje NA, N/A, NaN, NULL i praznih vrijednosti

U sljedećem poglavlju opisana je metoda pretprocesiranja podataka za obradu *NA, N/A, NaN, NULL* i praznih vrijednosti, metoda koja je povezana s prethodno navedenom metodom za traženje *NA, N/A, NaN, NULL* i praznih vrijednosti.

5.2.2.2. Metoda pretprocesiranja podataka za obradu *NA, N/A, NaN, NULL* i praznih vrijednosti

Druga metoda namijenjena je obradi *NA, N/A, NaN, NULL* i praznih vrijednosti odnosno nestalih vrijednosti u skupu podataka te omogućuje njihovu zamjenu unaprijed definiranim vrijednostima. Osim imputacije podataka, metoda dodatno kreira indikatorske stupce koji bilježe informaciju o prisutnosti izvornih nestalih vrijednosti s binarnim vrijednostima (0 i 1), gdje 1 označava da je određena vrijednost u izvornom stupcu bila *NA, N/A, NaN, NULL* ili prazna vrijednost, čime se omogućuje kasnija analiza i potencijalna upotreba tih značajki u modeliranju i predikciji s metodama strojnog učenja. Metoda se pokazala korisnom za očuvanje informacija o nestalim vrijednostima, umjesto da se one popune nekim određenim izračunom ili uklone i zanemare.

Metoda se sastoji od 3 procesne radnje, prikazane u nastavku:

1. Ulazni podatci
    - Skup tabličnih podataka df_{temp}
    - Stupci: Lista stupaca za obradu *columns* iz prethodne metode za utvrđivanje *NA, N/A, NaN, NULL* i praznih vrijednosti (*columns_with_missing*)
    - Vrijednost Točno/Netočno za zamjenu *NA, N/A, NaN, NULL* i praznih vrijednosti *value*
2. Iteracija po svakom stupcu
    - Za svaki stupac c ∈ *columns*:
        - Kreiranje stupca koji označava ima li *NA, N/A, NaN, NULL* i praznih vrijednosti:

$$c^{(i)}_{\_was\_missing} = \left(c^{(i)} = NA, N/A, NaN, NULL, ""\right), \forall i \in \{1, 2, ..., n\} \quad (6)$$

    - Logička provjera koja provjerava postoji li *NA, N/A, NaN, NULL* ili prazna vrijednost:

$$col\_was\_missing(i) = \begin{cases} value\ 1,\ ako\ postoji\ x_{ij} = NA, N/A, NaN, NULL, ""\ u\ stupcu\ col \\ value\ 0,\ u\ protivnom \end{cases}$$

(7)

- x_{ij} je element stupca *col* u retku *i*, 1 sadrži *NA, N/A, NaN, NULL* ili praznu vrijednost, 0 ne sadrži *NA, N/A, NaN, NULL* ili praznu vrijednost

3. Povratna vrijednost metode
    - Uređeni skup tabličnih podataka s novim stupcima df_{temp}

Primjer rada metode za obradu *NA, N/A, NaN, NULL* i praznih vrijednosti u skupu podataka:

7. Ulazni skup podataka (df_{temp}) i lista stupaca iz prethodne metode za utvrđivanje *NA, N/A, NaN, NULL* i praznih vrijednosti (*columns_with_missing*):

Id	Stambena površina u m2	Tip stambenog objekta	Površina podruma u m2	Širina kolnog ulaza
1	55	Stan	20	NA
2	245	Kuća u nizu	NULL	NA
3	180	Samostojeća kuća	50	6
4	155	Dvojna	NULL	10

8. Procesiranje stupca Površina podruma u m2, stvaranje stupca - Površina podruma u m2_was_missing
    - df_temp["Površina podruma u m2"].isnull() → [0, 1, 0, 1]
9. Trenutni skup podataka nakon procesiranja stupca Površina podruma u m2:

Id	Stambena površina u m2	Tip stambenog objekta	Površina podruma u m2	Širina kolnog ulaza	Površina podruma u m2_was_missing
1	55	Stan	20	NA	0
2	245	Kuća u nizu	NULL	NA	1
3	180	Samostojeća kuća	50	6	0
4	155	Dvojna	NULL	10	1

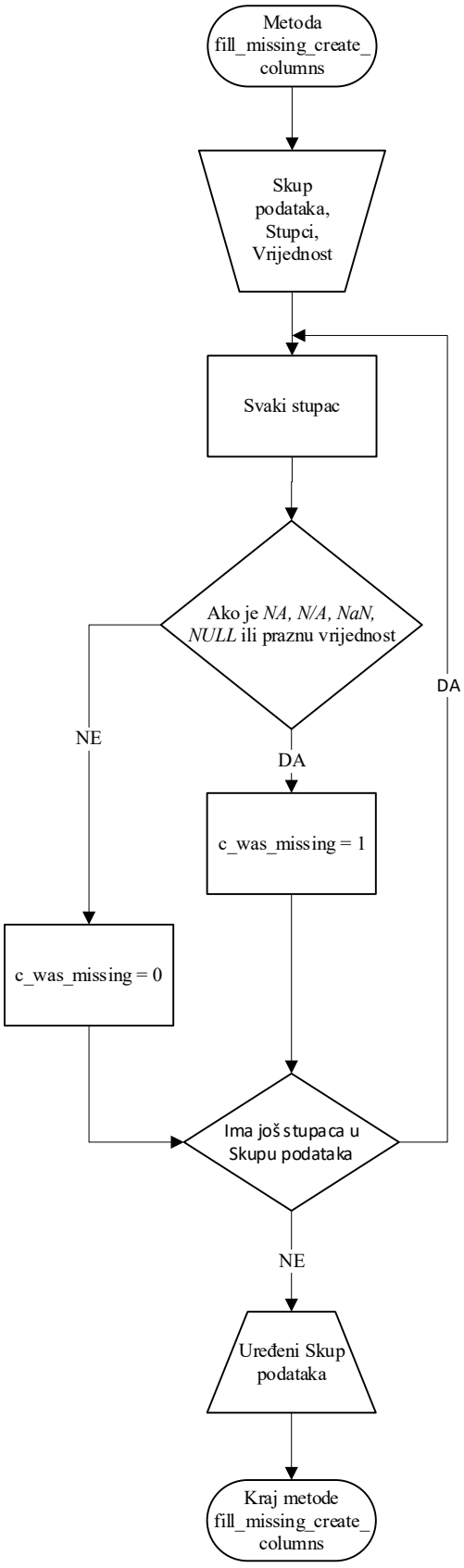
10. Procesiranje stupca Širina kolnog ulaza, stvaranje stupca Širina kolnog ulaza _was_missing
    - missing_markers = "NA", "N/A", "NAN", "NULL", ""
    - df_temp["Širina kolnog ulaza"].astype(str).str.strip().str.upper().isin(missing_markers)→ [1 , 1, 0, 0]
11. Završni skup podataka nakon procesiranja svih stupaca:

Id	Stambena površina u m2	Tip stambenog objekta	Površina podruma u m2	Širina kolnog ulaza	Površina podruma u m2_was_missing	Širina kolnog ulaza_was_missing
1	55	Stan	20	NA	0	1
2	245	Kuća u nizu	NULL	NA	1	1
3	180	Samostojeća kuća	50	6	0	0
4	155	Dvojna	NULL	10	1	0

Bitno je istaknuti kako izvorni stupci u skupu podataka se ne mijenjaju, metoda s novokreiranim indikatorskim stupcima/značajkama (*_was_missing*) omogućuje dodatnu analizu prisutnosti i utjecaja *NA*, *N/A*, *NaN*, *NULL* i praznih vrijednosti. Ovakav automatizirani pristup pridonosi boljem razumijevanju podataka u procesu prediktivnog modeliranja i optimizacije modela.

U prethodnom primjeru, u slučaju predikcije cijene stambenog objekta, odsutnost atributa, poput kolnog ulaza ili podruma, značajno utječe na konačnu procjenu vrijednosti nekretnine. Primjenom razvijene metode za obradu *NA*, *N/A*, *NaN*, *NULL* i praznih vrijednosti omogućilo se kvantificiranje utjecaja indikatorske značajke (*_was_missing*) na predikcijsku točnost modela. Ovakav automatizirani pristup je osigurao da se informacija o prisutnosti ili odsutnosti vrijednosti ne gubi, već integrira u automatizirani analitički proces, pri čemu se povećala interpretabilnost modela i omogućilo preciznije donošenje zaključaka. Bez procesiranja podataka razvijene metode, nedostajuće vrijednosti bi rezultirale gubitkom informacija i smanjenjem prediktivne točnosti modela. U rezultatima ovog istraživanja, ovakve značajke su bile ključne u procesu redukcije dimenzionalnosti, gdje su se zadržale u najrelevantnijim značajkama u reduciranom skupu podataka za predikciju preciznosti modela.

Dijagram toka podatka metode za utvrđivanje i označavanje (Točno (1)/Netočno (0)) redaka koji su imali *NA, N/A, NaN, NULL* i praznih vrijednosti u stupcima skupa podataka prikazan je slikom u nastavku.



Slika 29: Utvrđivanje i označavanje redaka koji su imali NA, N/A, NaN, NULL i praznih vrijednosti u stupcima skupa podataka

U sljedećem poglavlju opisana je metoda pretprocesiranja podataka za detekciju dupliciranih vrijednosti, metoda koja analizira ulazni skup podataka nakon metoda za traženje i obradu *NA, N/A, NaN, NULL* i praznih vrijednosti.

5.2.2.3. Metoda pretprocesiranja podataka za detekciju dupliciranih vrijednosti

Treća metoda namijenjena je za detekciju dupliciranih vrijednosti unutar skupa podataka. Primarni cilj metode je identificirati redundantne podatke koji mogu negativno utjecati na analizu ili modeliranje modela strojnog učenja. Dupli cirani zapisi mogu uzrokovati pristranost modela, povećati složenost analize te smanjiti učinkovitost predikcije metoda strojnog učenja. Metoda prihvaća dva ulazna parametra. Skup podataka df_{temp} koji se analizira te *Tihi ispis* - logički parametar (*bool*) koji određuje hoće li se rezultati detekcije dupliciranih vrijednosti ispisivati u konzoli. Kada je *Tihi ispis=False*, funkcija ispisuje ukupan broj dupliciranih zapisa te popis stupaca koji su isključeni iz analize. Prvi korak u metodi je identifikacija identifikatorskih stupaca koji sadrže isključivo jedinstvene vrijednosti, poput primarnih ključeva i *Id*-ova gdje broj jedinstvenih vrijednosti u stupcu odgovara broju redaka u skupu podataka. Takvi stupci se dodaju u zasebnu listu *id_cols* i isključuju iz daljnje analize jer ne mogu sadržavati duplicirane vrijednosti. Sljedeći korak uključuje kreiranje filtriranog podatkovnog skupa koji sadrži samo one stupce koji nisu identificirani kao identifikatori, što omogućuje precizniju detekciju dupliciranih redaka. Nakon toga, metoda računa broj dupliciranih redaka unutar preostalog skupa podataka. U slučaju da parametar *Tihi ispis* nije postavljen na *True*, funkcija ispisuje rezultate analize, uključujući ukupan broj dupliciranih redaka i popis identifikacijskih stupaca koji su izostavljeni. Zadaća metode je kontrola kvalitete podataka pružajući informacije o dupliciranim vrijednostima unutar skupa podataka.

Metoda se sastoji od 5 procesnih radnji, prikazanih u nastavku:

1. Ulazni podatci
    - Skup tabličnih podataka df_{temp}
    - Opcije:
        - Tihi ispis: Boolean vrijednost (Točno/Netočno) koja kontrolira ispis podataka. Točno predstavlja neprikazivanje ispisa.

2. Razdvajanje u dvije liste identifikatorskih stupaca i stupaca s dupliciranim vrijednostima
    - Inicijalizacija praznih lista:
        - *cols_to_use = []* : Lista stupaca koji s dupliciranim vrijednostima
        - *id_cols = []* : Lista stupaca koji imaju jedinstvene vrijednosti (identifikatori, Id, primarni ključevi)
    - Iteracija po svakom stupcu
        - Za svaki stupac c ∈ columns(df_{temp}):

$$IsId(c) = \begin{cases} 1, \ ako \ \left| jedinstvena(c) \right| = n \\ 0, \ u \ protivnom \end{cases} \tag{8}$$

        - Id stupci (*id_cols*):

$$\{c \mid IsId(c) = 1\} \tag{9}$$

        - Ostali stupci (*cols_to_use*):

$$\{c \mid IsId(c) = 0\} \tag{10}$$

3. Izrada novog skupa podataka
    - Stvaranje skupa podataka koji sadrži samo stupce iz liste *cols_to_use:*

$$df_{temp} = df_{temp}[cols\_to\_use] \tag{11}$$

    - S novim , stvorenim skupom podataka dupli cirane vrijednosti se provjeravaju samo u *cols_to_use* stupcima, dok se *id_cols* ignoriraju
4. Utvrđivanje dupliciranih vrijednosti
    - Utvrđuje dupli cirane vrijednosti u retcima *i* u skupu podataka $df_{filtered:}$

$$\forall c \in cols\_to\_use, df_{filtered}[c][i] = df_{filtered}[c][j] \tag{12}$$

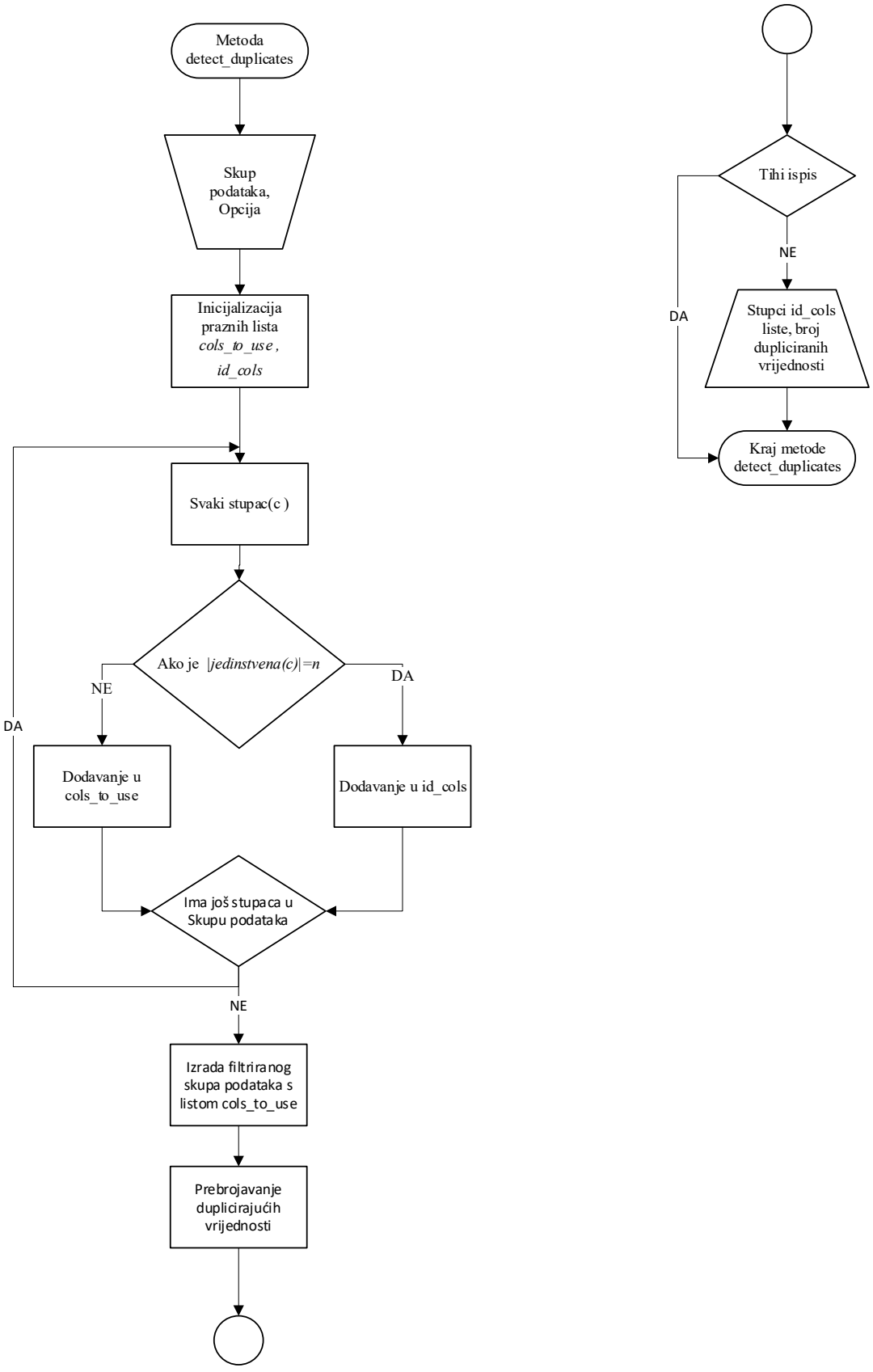
    - Prebrojava dupli cirajuće vrijednosti

$$count\_dupes = \sum_{i=1}^{n} \left(\exists j < i : \underset{c}{\wedge} c^{(i)} = c^{(j)} \right) \tag{13}$$

- Vraća 1 ako je vrijednost duplicirana, u protivnom 0

5. Ispis rezultata
    - Proces:
    - Ako je *Tihi ispis* = *Netočno*, ispisuje:
        - Nazive identifikatorskih stupaca iz liste *id_cols*
        - Ukupan broj dupliciranih vrijednosti

Dijagram toka podatka metode za pronalaženje i utvrđivanje dupliciranih vrijednosti u skupu podataka prikazan je slikom u nastavku.



Slika 30: Pronalaženje i utvrđivanje duplih vrijednosti

U sljedećem poglavlju opisana je metoda pretprocesiranja podataka za razdvajanje tekstualnih vrijednosti, metoda koja analizira ulazni skup podataka te traži tekstualne vrijednosti i razdvaja ih na više novih stupaca.

5.2.2.4. Metoda pretprocesiranja podataka za razdvajanje tekstualnih vrijednosti

Metoda za razdvajanje tekstualnih vrijednosti (engl. *string*) je razvijena za segmentaciju podataka unutar stupaca koji sadrže tekstualne vrijednosti u ulaznom skupu podataka (varijabla df_{temp}). Primarna svrha metode za razdvajanje tekstualnih vrijednosti je razdvajanje tekstualnih podataka na više novih stupaca na temelju unaprijed definiranog separatora (varijabla *separator_teksta*). Metoda za razdvajanje tekstualnih vrijednosti unutar skupa podataka koristi se za pretprocesiranje podataka odnosno za izvlačenje ključnih informacija te normalizaciju tekstualnih vrijednosti značajki u skupu podataka.

Metoda prima dva ulazna parametra. Prvi parametar je ulazni skup podataka (df_{temp}), unutar skupa podataka se pretražuju stupci koji sadrže tekstualne podatke. Parametar *separator_teksta* je separator koji definira način segmentacije podataka, pri čemu može biti bilo koji znak ili niz znakova, poput znaka razmaka, zareza, crtice, *CL* za razdvajanje svake riječi koja započinje velikim početnim slovom i slično.

Prvi korak metode uključuje selektiranje svih stupaca koji sadrže samo tekstualne podatke, čime iz analize, metoda, isključuje numeričke i druge tipove podataka – prvi korak je ključan zbog postizanja veće brzine u iterativnom postupku. Potom metoda iterativno prolazi kroz sve tekstualne stupce, provjeravajući sadrže li traženi separator. U slučaju da stupac sadrži separator, stupac se dijeli na više novih stupaca, pri čemu se svaki odvojeni segment tekstualne vrijednosti smješta u zaseban stupac.

Na završetku izvršavanja metode za razdvajanje tekstualnih vrijednosti novi stupci dodaju se u početni ulazni skup podataka (df_{temp}) te metoda vraća modificirani ulazni skup podataka s dodatnim stupcima koji sadrže podijeljene vrijednosti. Metoda se pokazala korisnom za bolju strukturalnu obradu i daljnju analizu tekstualnih podataka.

Metoda se sastoji se od 6 procesnih radnji, prikazanih u nastavku:

1. Ulazni podatci

- Skup tabličnih podataka df_{temp}
- Znak ili niz znakova koji određuje granicu između vrijednosti unutar stupca *separator_teksta*
- Skup tabličnih podataka df_{temp} je matrica koja se sastoji se od m redaka i n stupaca, pri čemu su neki stupci tekstualni:

$$df_{temp} = \{X_{ij} \mid i \in [1,m], j \in [1,n]\} \tag{14}$$

- Zadana razdjelnica *separator_teksta* definirana je kao simbol *S*, koji određuje gdje se dijeli vrijednost.

2. Identifikacija tekstualnih stupaca
   - Filtriraju se svi stupci koji imaju tip podataka Python *object*, budući da su to tekstualni stupci u skup stupaca C_T:

$$C_T = \{c \in C \mid X_{ic} \in string, \forall i\} \tag{15}$$

   - Pohranjuju se samo oni stupci koji imaju barem jednu pojavu znaka za razdvajanje
3. Razdvajanje stupaca prema razdjelnici
   - Za svaki stupac $c \in C_T$ koji sadrži *separator_teksta*, metoda razdvaja svaku vrijednost u stupcu na više novih stupaca, pri čemu se kreira nova matrica:

$$df_{temp}' = \{X_{ij}' \mid X_{ij}'\} \tag{16}$$

   - Ako je separator_teksta = ",", tada se *"tekstualna, vrijednost"* razdvaja u *"tekstualna"* i *"vrijednost"*.
   - Preimenuju se novonastali stupci, kako bi se očuvao kontekst izvornih podataka. Novi nazivi se generiraju na temelju originalnog naziva stupca *col* i rednog broja (*k*).
   - Ako je *separator_teksta* ≠ *" "*, tada se imenuju kao:

$$c_k' = c + S + k \tag{17}$$

   - Ako *separator_teksta* = *" "* , tada se imenuju kao:

$$c_k^{'} = c + k \tag{18}$$

- Ako *je col = "ImePrezime"* i *separator_teksta* = "_", tada *"Pero_Perić"* tada vrijednost *Pero* ide u novi stupac *" ImePrezime _0"*, a vrijednost *Perić* u novi stupac *" ImePrezime _1"*.

4. Popunjavanje praznih vrijednosti
   - Prazne vrijednosti koje nastanu uslijed razdvajanja zamjenjuju se nulama (brojčanom znamenkom 0) kako bi se osigurala konzistentnost podataka:

$$X_{ij}^{'} = 0, \text{ ako je } X_{ij}^{'} \text{ nedostaje} \tag{19}$$

5. Spajanje novog skupa podataka s originalnim
   - Novonastali segmentirani stupci dodaju se natrag u originalni skup podataka:

$$df_{temp}^{*} = df_{temp} \cup df_{temp}^{'} \tag{20}$$

6. Povratna vrijednost metode
   - Ažurirani skup tabličnih podataka df_{temp} s originalnim stupcima i novo segmentiranim stupcima koji su nastali podjelom tekstualnih vrijednosti.

Primjer rada metode za razdvajanje tekstualnih vrijednosti u skupu podataka:

1. Ulazni skup podataka (df_{temp}):

Id	Ime	Karta
1	Icard, Miss. Amelie	PC 17604
2	Slocovski, Mr. Selman	C.A. 392086

2. Poziv razvijene metode gdje je *separator_teksta* jedno prazno mjesto *(*" ")

$$break\_up\_by\_string(df\_temp, separator\_teksta = " ")$$

3. Tijek rada:
   - Odabrani stupci: Ime i Karta (tip *object*, tekstualni podatci)

- Za stupac Ime:
    - Razdvajanje na ["Icard,", " Miss.", " Amelie"] i ["Slocovski,", " Mr.", " Selman"]
    - Novi stupci: Ime0, Ime1 i Ime2
- Za stupac Karta:
    - Razdvajanje na ["PC", "17604"] i ["C.A.", "392086"]
    - Novi stupci: Karta0 i Karta1

4. Ažurirani skup podataka koji metoda vraća

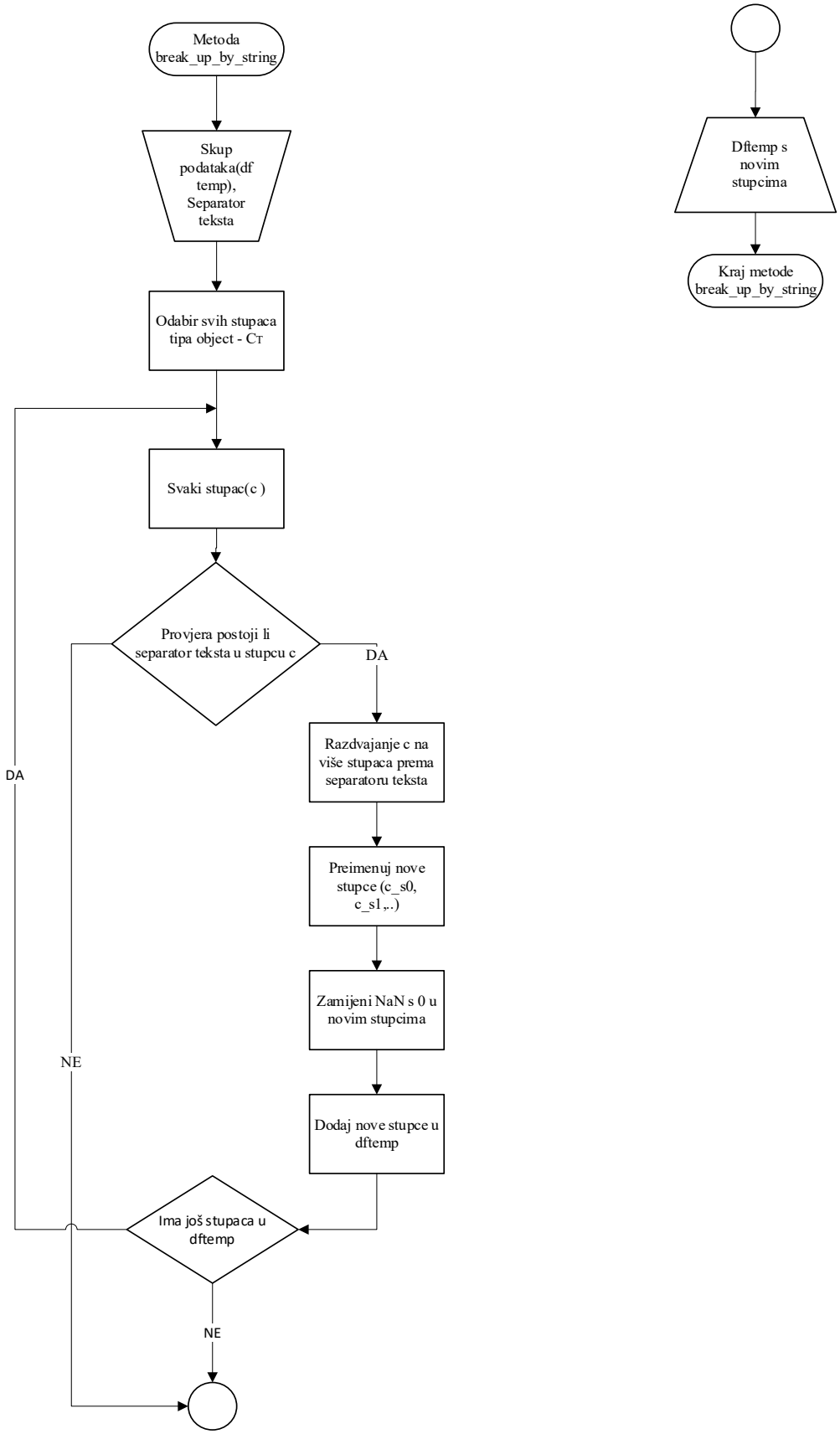
Id	Ime	Karta	Ime0	Ime1	Ime2	Karta0	Karta1
1	Icard, Miss. Amelie	PC 17604	Icard,	Miss.	Amelie	PC	17604
2	Slocovski, Mr. Selman	C.A. 392086	Slocovski,	Mr.	Selman	C.A.	392086

Bitno je istaknuti kako izvorni stupci ostaju nepromijenjeni te metoda radi samo sa stupcima tipa *object* (tekstualni podatci).

U konkretnom primjeru Titanic[56] skupa podataka, za predviđanje ishoda preživljavanja putnika, razdvajanjem atributa imena izdvojila se komponenta titule (*Miss.* i *Mr*.) koja nosi informaciju o spolu i društvenom statusu pojedinca (*Lady* ili *Sir*, ako je plemićki status) što predstavlja važnu značajku u kontekstu evakuacije Titanic-a - žene označene s *Miss* ili *Mrs* preživljavale su češće nego muškarci s titulom *Mr*.[57]. Zatim, podjela karata na prefiks i broj također donosi uvid u kabinsku klasu, prefiks *PC* označava prvu klasu, dok *C.A.* označava treću klasu[57]. Izdvajanjem titule i prefiksa stvoreni su uvjeti za izvršavanjem kategorizacije prefiksa karata s razvijenom metodom za kategorizaciju značajki u brojčani format. Iako, broj karte nema utjecaj na ishod preživljavanja te zbog toga i ne treba biti u reduciranom skupu značajki, s druge strane prefiks karte nosi informaciju o socijalnom, društvenom i materijalnom statusu putnika,

što, također, predstavlja važnu značajku u kontekstu evakuacije Titanic-a – putnici s višim socijalnim, društvenim i materijalnim statusom preživljavali su češće nego putnici s nižim statusom[57].

Dijagram toka podatka metode za razdvajanje tekstualnih vrijednosti u skupu podataka prikazan je slikom u nastavku.



Slika 31: Razvijena metoda za razdvajanje tekstualnih vrijednosti

U sljedećem poglavlju opisana je metoda pretprocesiranja podataka za uklanjanje značajki koje se ne nalaze u treniranom, validacijskom i testnom skupu podataka, metoda koja osigurava istu strukturu značajki u procesu treniranja i testiranja.

5.2.2.5. Metoda pretprocesiranja podataka za uklanjanje značajki koje se ne nalaze u treniranom, validacijskom i testnom skupu podataka

Metoda uklanja značajke odnosno stupce koji nisu zajednički prisutni u oba skupa podataka (treniranom, validacijskom ili testnom), uz iznimku određenih stupaca koji se eksplicitno izuzimaju od brisanja. Metoda se pokazala ključnom u fazi Pripreme podataka *MoziaisMl* sustava jer treniranje i testiranje modela se provodi na različitim podatkovnim skupovima, također u produkcijskom okružju priljev novih podataka dolazi konstantno, pa je bilo potrebno osigurati istu strukturu značajki kako bi predikcija metoda strojnog učenja bila učinkovita.

Metoda prima tri ulazna parametra - df_{temp} je prvi podatkovni skup koji se uspoređuje s drugim skupom podatka (df_{temp_2}) kako bi se identificirali neusklađeni stupci te osiguralo da oba skupa podataka imaju istu strukturu. Parametar *izuzmi_stupce* omogućuje definiranje stupaca koji se neće uklanjati, čak i ako nisu zajednički prisutni u oba skupa. Parametar se u *MoziaisMl* sustavu koristi kada postoji unaprijed definirana značajka kod problema klasifikacije ili regresije koja se ne smije izbrisati.

Metoda je uklanjanjem neusklađenih stupaca iz oba podatkovna skupa omogućila dosljednost ulaznih značajki te izravno poboljšala predikcijske performanse metoda strojnog učenja na različitim podatcima u fazi Evaluacija.

Metoda se sastoji se od 4 procesne radnje, prikazane u nastavku:

1. Ulazni podatci
    - Prvi skup tabličnih podataka df_{temp}
    - Drugi skup tabličnih podataka koji se uspoređuje s prvim df_{temp_2}
    - Lista stupaca koji se neće uklanjati, čak i ako nisu zajednički *izuzmi_stupce*
    - Skup svih stupaca u prvom skupu podataka:

$$C_1 = \left\{ c \in df_{temp} \right\} \tag{21}$$

- Skup svih stupaca u drugom skupu podataka:

$$C_2 = \{c \in df_{temp\_2}\} \tag{22}$$

- Skup stupaca koji su zajednički u oba skupa podataka:

$$C_{shared} = C_1 \cap C_2 \tag{23}$$

- Skup stupaca koji se neće uklanjati:

$$C_{exclude} = \{c_1, c_2, ..., c_n\} \tag{24}$$

2. Identifikacija stupaca koji nisu zajednički

- Iterira se kroz sve stupce df_{temp_2} i provjerava postoji li stupac u df_{temp}:

$$\forall c \in C_2, \text{ako } c \notin C_1 \text{ i } c \notin C_{exclude,} \text{ briši } c \tag{25}$$

- Ako stupac ne postoji u df_{temp} i nije u *izuzmi_stupce*, on se briše iz df_{temp_2}.
- Isto se ponavlja za df_{temp}:

$$\forall c \in C_1, \text{ako } c \notin C_2 \text{ i } c \notin C_{exclude,} \text{ briši } c \tag{26}$$

3. Brisanje stupaca koji nisu zajednički
   - Ako stupac ne postoji u drugom skupu podataka i nije u *izuzmi_stupce*, stupac se briše.
   - Operacija brisanja provodi pritom osiguravajući da se podatci mijenjaju unutar postojećih objekata, bez potrebe za dodatnim kopiranjem.
   - Nakon ovog koraka, preostali stupci u df_{temp} i df_{temp_2} su identični, osim onih koji su izuzeti od brisanja.
4. Rezultat
   - Metoda ne vraća eksplicitnu vrijednost, već modificira oba skupa podataka df_{temp} i df_{temp_2} u mjestu.
   - Nakon izvršenja metode, oba skupa podataka imaju istu strukturu značajki, osim stupaca koji su izričito izuzeti od brisanja.

Primjer rada metode za uklanjanje značajki koje se ne nalaze u treniranom, validacijskom i testnom skupu podataka:

1. Prvi ulazni skup podataka (df_{temp}):

Id	Ime	Dob	Županija
1	Ivana	26	Osječko-baranjska
2	Pero	29	Varaždinska

2. Drugi ulazni skup podataka (df_{temp_2}):

Id	Grad	Plaća	Županija
1	Osijek	1.200	Osječko-baranjska
2	Varaždin	1.800	Varaždinska

3. Poziv razvijene metode gdje je vrijednost parametara *izuzmi_stupce*: Dob

```
drop_unshared_columns(df_temp, df_temp_2, izuzmi_stupce="Dob")
```

4. Tijek rada:
   - Procesiranje iteracijom petlje kroz stupce u drugom ulaznom skupu podataka (df_{temp_2}):
     - Id: Prisutan u df_{temp} → metoda zadržava
     - Grad: Nije prisutan u df_{temp} i nije u parametru *izuzmi_stupce* → metoda uklanja
     - Plaća: Nije prisutan u df_{temp} i nije u parametru *izuzmi_stupce* → metoda uklanja
     - Županija: Prisutan u df_{temp} → metoda zadržava
   - Rezultat drugog ulaznog skupa podataka (df_{temp_2}) nakon iteracije petljom:

Id	Županija
1	Osječko-baranjska
2	Varaždinska

- Procesiranje iteracijom petlje kroz stupce u prvom ulaznom skupu podataka (df_{temp}):
    - Id: Prisutan u df_{temp_2} → metoda zadržava
    - Ime: Nije prisutan u df_{temp_2} i nije u parametru *izuzmi_stupce* → metoda uklanja
    - Dob: Nije prisutan u $df_{temp\_2,}$ ali je u parametru *izuzmi_stupce* → metoda zadržava
    - Županija: Prisutan u df_{temp_2} → metoda zadržava
- Rezultat prvog ulaznog skupa podataka (df_{temp}) nakon iteracije petljom:

Id	Dob	Županija
1	26	Osječko-baranjska
2	29	Varaždinska

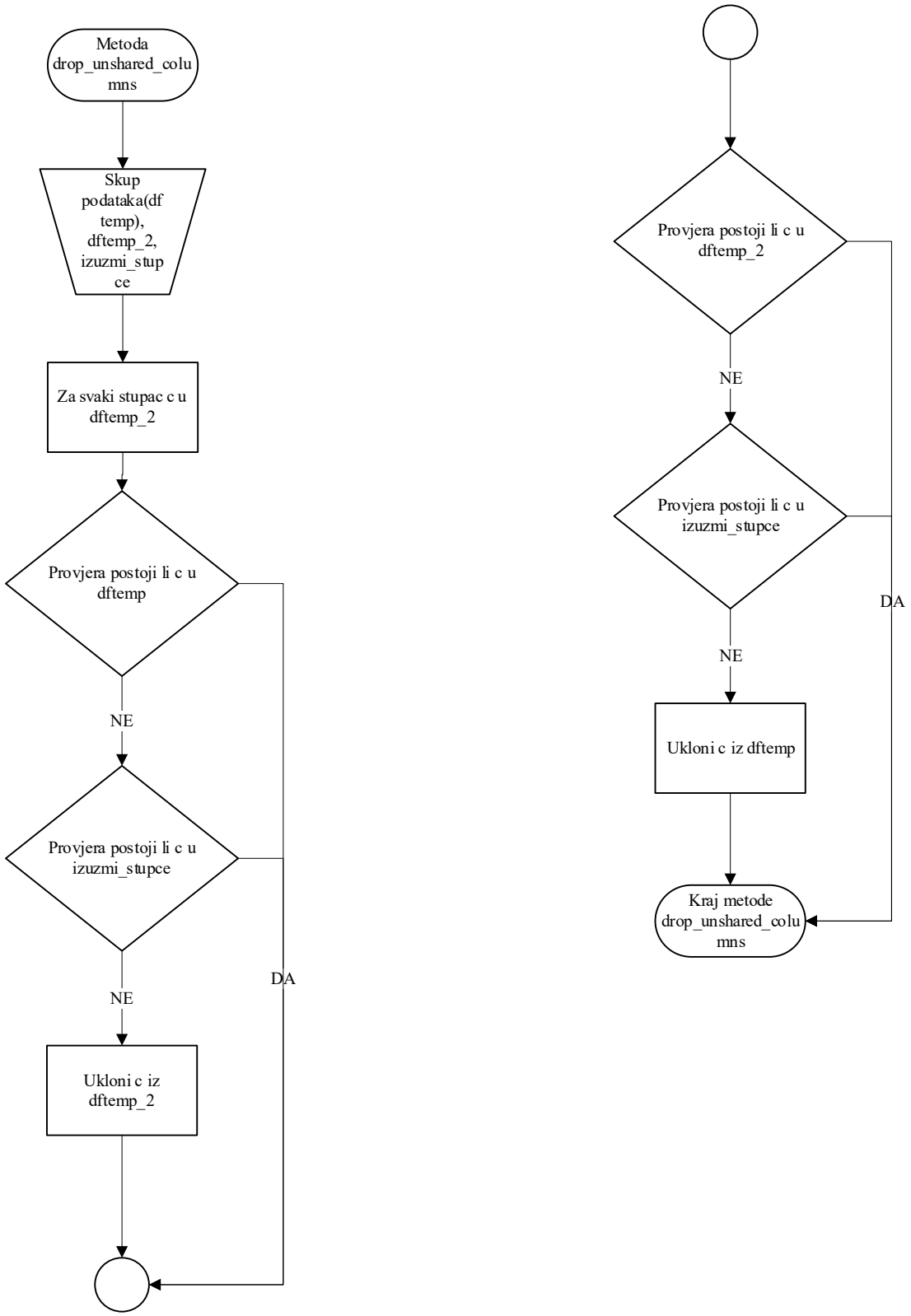
5. Završni rezultat:
    - Prvi ulazni skup podataka (df_{temp}):

Id	Dob	Županija
1	26	Osječko-baranjska
2	29	Varaždinska

- Drugi ulazni skup podataka (df_{temp_2}):

Id	Županija
1	Osječko-baranjska
2	Varaždinska

Dijagram toka podatka metode za uklanjanje značajki koje se ne nalaze u treniranom, validacijskom ili testnom skupu podataka prikazan je slikom u nastavku.



Slika 32: Metoda za uklanjanje značajki koje se ne nalaze u treniranom, validacijskom i testnom skupu podataka

U sljedećem poglavlju opisana je metoda pretprocesiranja podataka za kategorizaciju značajki u brojčani format, metoda koja omogućuje upotrebu kategorijskih/tekstualnih značajki u procesu predikcije raznih metoda i modela strojnog učenja.

5.2.2.6. Metoda pretprocesiranja podataka za kategorizaciju značajki u brojčani format

Metoda pretprocesiranja podataka za kategorizaciju značajki razvijena je za pretvaranje kategorijskih varijabli u brojčani format, čime se omogućuje, s brojčanim formatom, njihova upotreba u modelima strojnog učenja. Kategorijske varijable često metode strojnog učenja ne mogu izravno analizirati, zbog toga razvijena metoda kategorizacije značajki automatizmom primjenjuje jedan od dva pristupa kategorizacijskih metoda – *LabelEncoder*[58] ili *OneHotEncoder*[59], ovisno o broju jedinstvenih vrijednosti u stupcu, a s ciljem omogućavanja analize. Većina metoda strojnog učenja zahtijeva numeričke ulazne podatke za predikciju nad podatcima. Stoga, kako bi se omogućila adekvatna reprezentacija kategorijskih značajki unutar skupa podataka, potrebno je provesti postupak pretvorbe, s razvijenom metodom, u brojčani format, gdje se kategorijske značajke kategoriziraju u brojčani format. Na taj način osigurava se optimalna redukcija dimenzionalnosti, bez gubitka informacija – što predstavlja svrhu ovog istraživanja. Metoda prihvaća četiri ulazna parametra. Parametar *df* predstavlja skup podataka koji sadrži stupce koje treba kodirati u brojčani format. Parametar *stupci* sadrži popis stupaca u skupu podataka koji će biti transformirani u brojčani format. Parametar *test_df* je skup podataka koji se koristi za kodiranje podataka iz testnog skupa podataka na temelju istih pravila kao u treniranju – predstavlja važan element u metodološkom okviru i *MoziaisMl* sustavu zbog potrebe za radom *MoziaisMl*-a u produkcijskom okružju. Parametar *prag*, sa zadanom vrijednošću 12, određuje prag broja jedinstvenih vrijednosti u stupcu, prema kojem se odlučuje hoće li se koristiti metoda *OneHotEncoder*[59] (za varijable s manjim brojem kategorija) ili metoda *LabelEncoder*[58] (za varijable s većim brojem kategorija). Uloga praga s vrijednošću 12 ima značaj kod kategorijskih značajki koje imaju 12 ili manje jedinstvenih vrijednosti, tada se koristiti metoda *OneHotEncoder*[59] jer broj novih generiranih varijabli neće

značajno povećati dimenzionalnost skupa podataka. U protivnom, ako kategorijska značajka ima više od 12 jedinstvenih vrijednosti korištenje metode *OneHotEncoder*[59] rezultira s velikim brojem novokreiranih značajki te s tim zahtjeva i veće računalne resurse kod procesiranja, a svrha ovog rada jest smanjena upotreba računalnih resursa, smanjenje financijskih troškova i postizanje više brzine. Stoga, za kategorijske značajke s većim brojem jedinstvenih vrijednosti kategorija koristi se metoda *LabelEncoder*[58] jer metoda *LabelEncoder*[58] ne rezultira s velikim brojem novokreiranih značajki kad kategorijska značajka ima više od 12 jedinstvenih vrijednosti. Ovaj pristup omogućuje kontrolu dimenzionalnosti i onemogućavanje stvaranje visokodimenzionalnih podataka, što je posebno važno za ovo istraživanje u kojemu je i razvijen sustav *MoziaisMl* za smanjivanje dimenzionalnosti podataka. Programska logika metode analizira prelazi li zadani prag broj jedinstvenih vrijednosti unutar stupca te ako ne prelazi zadani prag i nije binarna varijabla, koristi se metoda *OneHotEncoder*[59], pri čemu se za svaku kategoriju kreira zaseban binarni stupac. U slučaju da je broj jedinstvenih vrijednosti veći od praga, koristi se metoda *LabelEncoder*[58], gdje se svaka kategorija pretvara u numeričku vrijednost. Metoda pretprocesiranja podataka za kategorizaciju značajki automatizira proces kodiranja kategorijskih varijabli, osiguravajući konzistentnost između skupa podataka za treniranje i testiranje te omogućujući kontrolu dimenzionalnosti kodiranih značajki, čime poboljšava brzinu i predikciju metoda strojnog učenja.

Metoda pretprocesiranja podataka za kategorizaciju značajki integrira metodu *LabelEncoder*[58] i *OneHotEncoder*[59], ali i uvodi dodatnu logiku za donošenje odluka o odabiru između dviju strategija kodiranja i prilagodbu testnog skupa podataka. Doprinos metode je odluka o vrsti kodiranja na temelju broja jedinstvenih vrijednosti u stupcu i unaprijed definiranog praga (parametar *prag*). Prethodno navedena logika omogućuje da stupci s manjim brojem jedinstvenih kategorija koriste metodu *OneHotEncoder*[59], dok se za stupce s velikim brojem kategorija koristi metoda *LabelEncoder*[58] te na taj način razvijena metoda sprječava prekomjernu dimenzionalnost podataka. Drugi doprinos metode je povezan s testnim skup podataka (parametar *test_df*), gdje metoda osigurava dosljedno kodiranje neviđenih vrijednosti. To postiže iterativnim pretraživanjem svakog retka testnog skupa podataka te zamjenom nepoznatih kategorija oznakom *None*, koja se prethodno dodaje prilikom treniranja *LabelEncoder*[58] objekta. Ova funkcionalnost nije izvorno podržana u metodama

LabelEncoder[58] i *OneHotEncoder*[59], već je dodatno razvijena za *MoziaisMl* sustav ovog istraživanja, kako bi se omogućila generalizacija modela i spriječile greške povezane s novim kategorijama u testnom skupu podataka.

Metoda se sastoji se od 4 procesne radnje, prikazane u nastavku:

1. Ulazni podatci
    - Skup tabličnih podataka koji sadrži kategoričke značajke *df*
    - Lista stupaca koje je potrebno kodirati *stupci*
    - Testni skup podataka koji će biti transformiran na temelju skupa tabličnih podataka koji sadrži kategoričke značajke *test_df*
    - Prag za odabir metode kodiranja *prag*
        - Ako broj jedinstvenih vrijednosti stupca < *prag*, koristi se metoda *OneHotEncoder*[59].
        - Ako broj jedinstvenih vrijednosti stupca ≥ *prag*, koristi se metoda *LabelEncoder*[58].
    - Skup svih stupaca koje je potrebno kodirati:

$$C = \{c_1, c_2, ..., c_n\} \tag{27}$$

    - Skup jedinstvenih vrijednosti za svaki stupac:

$$U_c = \{X_{ic} \mid i \in [1, m]\} \tag{28}$$

2. Iteracija kroz stupce koje je potrebno kodirati
    - Za svaki stupac *c* iz stupci:
        - Kreira se instanca *LabelEncoder():*

$$LE_c = LabelEncoder() \tag{29}$$

        - Prikupljaju se sve unikatne vrijednosti unutar stupca i sortiraju:

$$U_c = \{x_1, x_2, ..., x_k\}, \text{ pri čemu je } k = |U_c| \tag{30}$$

        - Dodaje se dodatna kategorija *"None"* za nepoznate vrijednosti:

$$U_c^{'} = U_c \cup \{"None"\} \tag{31}$$

3. Primjena kodiranja
    - Ako broj unikatnih vrijednosti u stupcu *c (k)* je manji od *prag* i nije binarna varijabla:
        - Primjenjuje se metoda O*neHotEncoder*[59]
        - Generira se nova matrica značajki df_{temp}' u kojoj svaki jedinstveni unos postaje zaseban binarni stupac:

$$X_{ij}^{'} = \begin{cases} 1, \textit{ako je} & X_{ij} = U_k \\ 0, \textit{inače} \end{cases} \tag{32}$$

        - Ako *test_df* nije *None*, istu transformaciju metoda primjenjuje na testni skup.
    - Ako broj jedinstvenih vrijednosti (*k*) je veći ili jednak prag:
        - Primjenjuje se metoda *LabelEncoder*[58]
        - Svakoj kategoriji dodjeljuje se jedinstven cijeli broj:

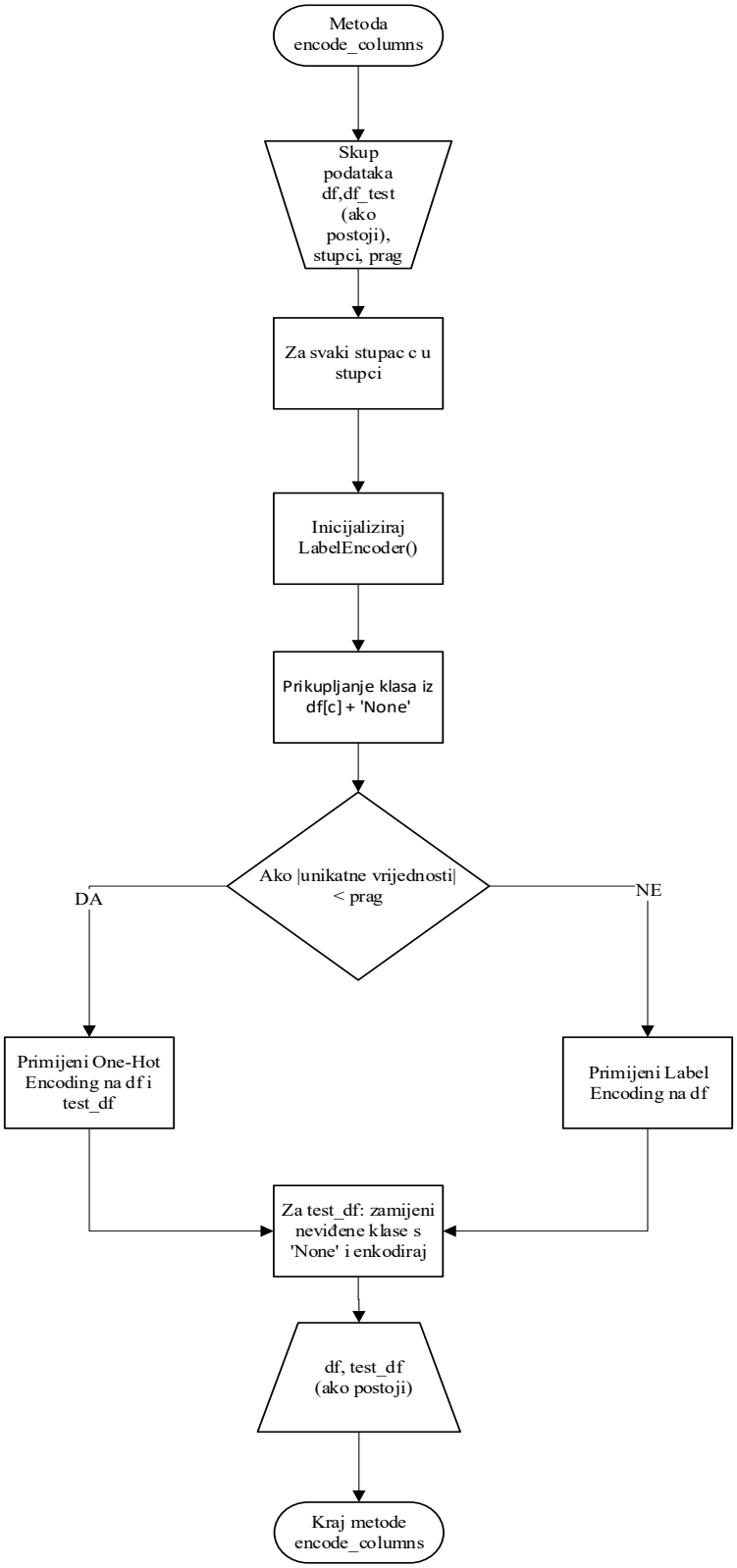
$$LE_c\left(U_k\right) = \left\{x_1 : 0, x_2 : 1, ..., x_k : \left(k-1\right)\right\} \tag{33}$$

        - Kodirana vrijednost se zamjenjuje u originalnom skupu podataka:

$$X_{ic} = LE_c\left(X_{ic}\right) \tag{34}$$

    - Ako postoji *test_df*, potrebno je ukloniti nepoznate oznake u testnom skupu:
        - Iteriramo kroz *test_df* i metoda provjerava je li vrijednost prisutna u *df[c]*.
        - Ako vrijednost nije u skupu *df[c],* zamjenjuje se *"None"* i transformira pomoću metode *LabelEncoder*[58].
4. Povratna vrijednost metode
    - Metoda vraća dva skupa podataka:
        - transformirani skup podataka za treniranje *df*
        - ako postoji, transformirani skup podataka za testiranje *test_df*

Dijagram toka podatka metode podataka za kategorizaciju značajki u brojčani formatu u skupu podataka za treniranje i, ako postoji, testnom skupu podataka prikazan je slikom u nastavku.



Slika 33: Metoda za kategorizaciju značajki u brojčani format

U sljedećem poglavlju opisana je metoda pretprocesiranja podataka za normalizaciju i skaliranje značajki, metoda koja omogućuje normalizaciju i stabilizaciju varijance podataka.

5.2.2.7. Metoda pretprocesiranja podataka za normalizaciju i skaliranje značajki

Metoda pretprocesiranja podataka za normalizaciju i skaliranje značajki implementira kvantilnu transformaciju stupaca unutar skupa podataka čime postiže redistribuciju vrijednosti kako bi se podatci uskladili s određenom distribucijom. Navedena metoda transformaciju koristi za normalizaciju i stabilizaciju varijance podataka zbog mogućih odstupanja u distribuciji ulaznih značajki. Cilj metode je smanjivanje utjecaja iznimnih vrijednosti i poboljšanje metoda strojnog učenja u zadatcima predikcije jer modeli metoda strojnog učenja zahtijevaju podatke s približno normaliziranom distribucijom za optimalne performanse.

Metoda pretprocesiranja podataka za normalizaciju i skaliranje značajki glavnu funkcionalnost transformacije kvantila preuzima iz metode *QuantileTransformer*[60], koja prilagođava distribuciju podataka tako da slijedi odabranu ciljnu distribuciju - *uniform* ili *normal*. Također, primjenjuje monotono povećavajuću funkciju transformacije kako bi zamijenila vrijednosti značajki njihovim empirijskim kvantilima, čime se postiže stabilizacija varijance i uklanjanje utjecaja iznimnih vrijednosti. Međutim, razvijena metoda za normalizaciju i skaliranje značajki ima i dodatno razvijenu logiku, koja nije dio metode *QuantileTransformer*[60]. Prvi dio dodatne logike odnosi se na dinamičko određivanje broja kvantila (parametar *n_samples*). Umjesto korištenja fiksnog broja kvantila, metoda za normalizaciju i skaliranje značajki provjerava veličinu skupa podataka i postavlja parametar *n_samples* na manju vrijednost ako je veličina skupa podataka manja od 1000, s čim sprječava brojčane pogreške i optimizira izvedbu transformacije. Drugi dio dodatne logike je iterativna obrada stupaca, gdje metoda za normalizaciju i skaliranje značajki prolazi kroz svaki pojedini stupac skupa podataka i provjerava postoji li ciljna varijabla (parametar *y*) te izuzima njezinu transformaciju kako bi se osiguralo zadržavanje izvorne distribucije izlaznih podataka. Dodatna logika metode za normalizaciju i skaliranje značajki poboljšava kompatibilnost s različitim veličinama ulaznih skupova podataka te specifičnim zahtjevima različitih metoda strojnog učenja –

bitan element metodološkog okvira *MoziaisMl* sustava koji analizira različite poslovne domene i veličine skupova podataka te za predikciju koristi različite metode strojnog učenja.

Metoda se sastoji se od 4 procesne radnje, prikazane u nastavku:

1. Ulazni podatci
    - Skup tabličnih podataka koji se transformira df_{temo}
    - Stupac koji se ne transformira *tc*
    - Ciljana raspodjela transformiranih podataka *output_distribution*
        - *"uniform"*: podatci se transformiraju tako da imaju jednoličnu (uniformnu) distribuciju.
        - *"normal"*: podatci se transformiraju tako da imaju normalnu distribuciju.
    - Skup svih numeričkih stupaca u skupu podataka:

$$C_N = \{c \in C \mid X_{ic} \in \mathbb{R}, \forall i\} \tag{35}$$

    - Skup numeričkih stupaca koji će biti transformirani:

$$C_T = C_N \setminus \{tc\} \tag{36}$$

2. Priprema broja uzoraka (*n_samples*)
    - Postavlja se maksimalni broj uzoraka (*n_samples*) za kvantilnu transformaciju
    - Ako broj redaka u df_{temp} prelazi 1000, koristi se 1000 kvantila.
    - Ako je broj redaka manji od 1000, *n_samples* se postavlja na stvarni broj redaka.

$$n_{samples} = \begin{cases} 1000, ako\ je & m \geq 1000 \\ m, \quad ako\ je & m < 1000 \end{cases} \tag{37}$$

3. Iteracija kroz stupce i primjena kvantilne transformacije
    - Za svaki numerički stupac *c* iz df_{temp} koji nije *tc*:

- Kreira se instanca metode *QuantileTransformer*[60] s definiranim brojem kvantila (*n_samples*) i ciljanom distribucijom (*output_distribution*).
- Primjenjuje se transformacija na svaki pojedinačni stupac:

$$X_{ic}^{'} = Q_T\left(X_{ic}\right) \tag{38}$$

gdje je Q_T funkcija kvantilne transformacije definirana kao:

$$Q_T\left(X\right) = \Phi^{-1}\left(F\left(X\right)\right) \tag{39}$$

pri čemu je

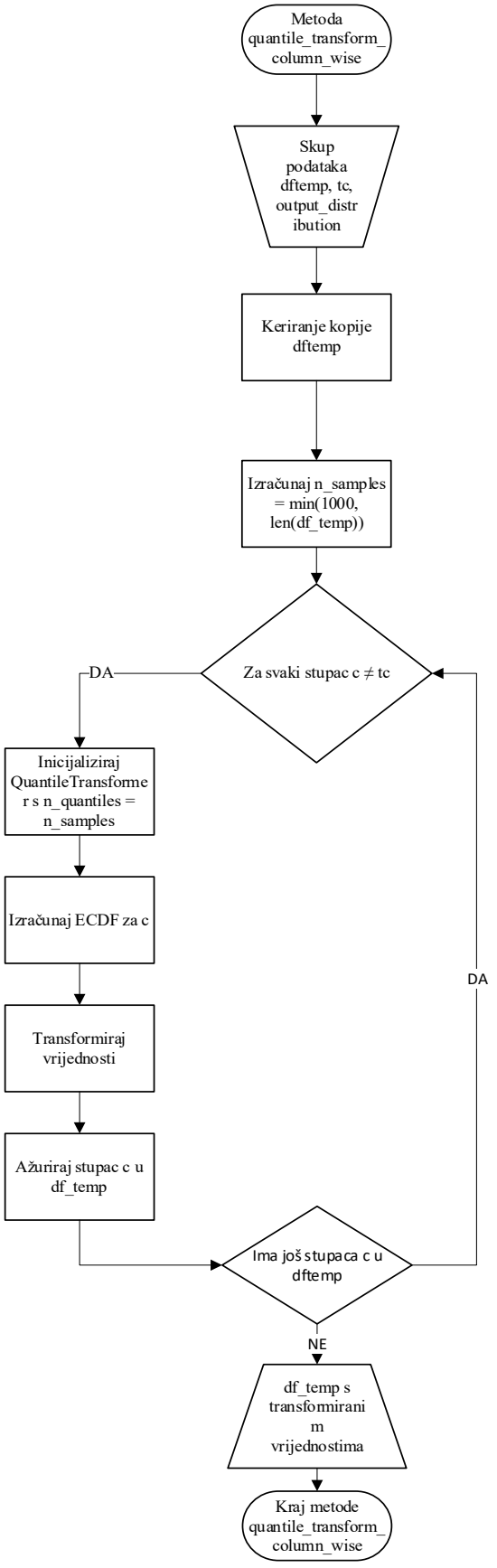
- *F(X)* empirijska distribucijska funkcija (ECDF) originalnih podataka.
- Φ^{-1} inverzna funkcija ciljne distribucije (npr. uniformna ili normalna).

- Transformirani podatci zamjenjuju originalne vrijednosti u podatkovnom okviru:

$$df_{temp}\left[c\right] = X_{ic}^{'} \tag{40}$$

4. Povratna vrijednost metode
   - Metoda vraća transformirani skup podataka $df_{temp,}$ gdje su svi numerički stupci, osim *tc*, transformirani u definiranu distribuciju - *uniform* ili *normal*.

Dijagram toka podatka metode za normalizaciju i skaliranje značajki u skupu podataka prikazan je slikom u nastavku.



Slika 34: Metoda za normalizaciju i skaliranje značajki

U sljedećem poglavlju opisana je razvijena metoda *Glasanja*, koja se temelji na agregaciji značajki koje su među najvažnijima prema razvijenom ansamblu četiri različite metode.

5.2.2.8. Metoda redukcije dimenzionalnosti podataka – metoda Glasanja

Metoda Glasanja koristi metode za klasifikacijske ili regresijske zadatke *RandomForest*, *XGBoost*, *Mutual Information* i *Recursive feature elimination* za procjenu važnosti značajki, a metodu *StratifiedKFold Cross-Validation* za klasifikacijske zadatake, a za regresijske zadatke *KFold* za izračun važnosti značajki.[38] Provedbom predistraživanja pokazalo se koje su metode te postavke metoda strojnog učenja najučinkovitije za korištenje kod problema redukcije dimenzionalnosti u *MoziaisMl* sustavu. Nastavno, metode strojnog učenja koje su se pokazale učinkovite za redukciju dimenzionalnosti, korištene su u istraživanju ovoga rada.

Korištenjem metode *Stratified K-Fold* (klasifikacijski zadatak) ili *KFold* (regresijski zadatak) *Cross Validation* [32, 33] rangiraju se značajke (engl. *feature ranking*) skupa podataka procjenjujući važnost značajki na temelju iterativnog treniranja modela na različitim podskupovima podataka (engl. *fold*) unutar ulaznog skupa podataka, čime se smanjuje mogućnost pristranosti u rezultatima te povećava pouzdanost evaluacije značajki. Metoda *Stratified K-Fold* (klasifikacijski zadatak) ili *KFold* (regresijski zadatak) *Cross Validation* u *MoziaisMl* sustavu prima tri ulazna parametra – *X*, *y* i *model*. Parametar *X* predstavlja skup značajki (iz skupa podataka) koji se analizira, dok *y* sadrži ciljnu varijablu koja se koristi za učenje modela. Parametar *model* odnosi se na instance 4 metode strojnog učenja koji se koristi za procjenu važnosti značajki - *RandomForest[25, 26]*, *XGBoost[27]*, *Mutual Information[28, 29]* i *Recursive feature elimination[30]*. U *MoziaisMl* sustavu je napravljen cjevovod koji automatiziranim putem prihvaća sve 4 metode strojnog učenja za procjenu važnosti značajki. Vlastito razvijena metoda *Glasanja* podržava dohvaćanje značajnosti značajki putem atributa *feature_importances_* ili *coef_*. Prvi korak uključuje inicijalizaciju metode *Stratified K-Fold* (klasifikacijski zadatak) ili *KFold* (regresijski zadatak) *Cross Validation* s pet podjela/podskupova podataka (engl. *K-Fold, 5-Fold* jer ima 5 podjela), pri čemu se osigurava da distribucija ciljne varijable u svakom preklopu ostane proporcionalna

ukupnoj distribuciji u skupu podataka. Nadalje, postavka metode *Stratified K-Fold* (klasifikacijski zadatak) ili *KFold* (regresijski zadatak) *Cross Validation shuffle=True* omogućuje permutaciju redoslijeda podataka prije podjele, dok *random_state=42* osigurava reproducibilnost eksperimenta. Zatim, se inicijalizira vektor *feature_importances*, koji pohranjuje kumulativnu važnost svake značajke tijekom iteracija *5-Folda*. Metoda *Glasanja* zatim prolazi kroz pet iteracija, pri čemu u svakoj iteraciji dijeli podatke na trening skup (*X_train*, *y_train*) i testni skup (*X_test, y_test*). Parametar *model* se trenira na trening skupu podataka, a nakon treniranja, provjerava se podržava li parametar *model* dohvaćanje važnosti značajki putem atributa *feature_importances_* ili *coef_*. Ako parametar *model* posjeduje *feature_importances_,* vrijednosti se zbrajaju s postojećim vektorom značajnosti, u protivnom ako parametar *model* koristi *coef_*, računa se apsolutna vrijednost koeficijenata kako bi se osigurala konzistentna interpretacija važnosti značajki. Nakon što je iteracija kroz svih pet podskupova podataka (*5-Fold*-ova) završena, normaliziraju se vrijednosti značajnosti značajki. Zatim, sortira se skup podataka svih značajki, pri čemu su sve značajke poredane prema njihovoj važnosti u silaznom redoslijedu.

Razvijena metoda *Glasanja* implementira vlastiti ansambal metoda *RandomForest [25, 26]*, *XGBoost[27]*, *Mutual Information[28, 29]* i *Recursive feature elimination [30]* za odabir značajki koristeći glasački konsenzus, pri čemu se kombiniraju rezultati 4 navedene metode za rangiranje značajki. Cilj metode *Glasanja* je poboljšanje selekcije značajki korištenjem 4 različite metode strojnog učenja odnosno procjene značajnosti značajki te 4 različite metode i odabir najrelevantnijih značajki na temelju njihovog učestalog pojavljivanja među najboljim značajkama unutar 4 prethodno navedene metode. Ovakav razvijeni algoritam omogućuje smanjenje varijance u procesu selekcije značajki, čime povećava stabilnost modela i poboljšava interpretabilnost rezultata. Metoda *Glasanja* prima dva ulazna parametra. Parametar *X* predstavlja skup podataka značajki, pri čemu redci predstavljaju uzorke, a stupci pojedinačne značajke. Parametar *y* je ciljna varijabla, koja sadrži pripadajuće klase ili kontinuirane vrijednosti ovisno o tipu problema - klasifikacija ili regresija. U prvom dijelu razvijene metode *Glasanja* provodi se rangiranje značajki korištenjem metode *Stratified K-Fold* (klasifikacijski zadatak) ili *KFold* (regresijski zadatak) *Cross Validation* [32, 33] koja koristi 4, prethodno navedene, različite metode:

1. Random Forest[25, 26] za procjenu značajnosti značajki na temelju raspodjele njihovih važnosti u stablima odlučivanja.
2. XGBoost[27] koji primjenjuje gradijentno pojačanje za rangiranje značajki.
3. Temelji se na međusobnoj informaciji[28, 29] kako bi se izmjerila ovisnost između značajki i ciljne varijable.
4. Koristi rekurzivnu eliminaciju značajki[30] koja iterativno uklanja najmanje važne značajke.

Nakon što su 4 različite metode (*RF, XGB, MI* i *RFE*) generirale vlastite rang liste značajki, određuje se broj značajki koje će biti uzete u obzir na temelju postotnog praga definiranog parametrom *top_postotak*. Broj top značajki (parametar *T*), koje se odabiru, izračunava se kao proporcija ukupnog broja značajki, primjer - ako je postotak postavljen na 10%, odabrat će se 10% najvažnijih značajki iz svake od 4 različite metode(*RF, XGB, MI* i *RFE*).

Sljedeći korak metode *Glasanja* uključuje inicijalizaciju glasova za svaku značajku. Kreira se skup podataka s nultim vrijednostima, gdje svaki stupac (značajka) dobiva početni broj glasova 0. Zatim, prolazi se kroz rezultate sve 4 različite metode (*RF, XGB, MI* i *RFE*) rangiranja, te se najvažnijim značajkama dodjeljuju glasovi. Svaka značajka koja se pojavljuje među vrijednosti parametra broj top značajki *T* u nekoj metodi dobiva jedan dodatni glas.

Nakon prikupljenih glasova, metoda *Glasanja* značajke sortira prema broju osvojenih glasova u silaznom redoslijedu, a konačni odabir vrši se uzimanjem vrijednošću broja top značajki *T* s najviše glasova. Ove značajke spremaju se u varijablu *odabrane_znacajke*, nakon čega funkcija vraća modificirani skup podataka *X*[*odabrane_znacajke*], koji sadrži samo odabrane značajke.

Metoda *Glasanja* kroz kombinirano glasovanje osigurava veću pouzdanost selekcije značajki jer smanjuje utjecaj specifičnih pristranosti pojedinačnih metoda kroz korištene sve 4 različite metode (*RF, XGB, MI* i *RFE*). Na taj način ulazni skup podataka ima najinformativnije značajke koje su konzistentno identificirane kao važne kroz različite pristupe, čime se poboljšava interpretabilnost i predikcija konačnog modela strojnog učenja.

Metoda Glasanja se sastoji od 7 procesnih radnji, prikazanih u nastavku:

1. Ulazni podatci
    - X: Ulazni skup podataka (matrica sa značajkama, dimenzije $n \times m$).
    - y: Izlazna, ciljna varijabla (vektor duljine n)
    - *top_postotak*: *n%* najvažnijih značajki za odabir, određuje korisnik sustava *MoziaisMl*
    - *tip_zadatka:* klasifikacija ili regresija
2. Odabir značajki s RF, XGB, MI i RFE metodama
    - Svaka metoda zasebno radi svoju rang listu značajki prema njihovoj važnosti koristeći *StratifiedKFold* ili *KFold Cross-Validation* za izračun značajnosti
    - Za svaku metodu $M \in \{RF, XGB, MI, RFE\}$, izračunava prosječnu važnost značajke f_i:

$$Važnost_M(f_i) = \frac{1}{K}\sum_{k=1}^{K} Izračun_M^{(k)}(f_i) \tag{41}$$

    - $K = 5$ (*KFold*), broj određuje korisnik sustava *MoziaisMl*
3. Određivanje broja značajki T za uzeti iz ulaznog skupa podataka na temelju postotka:

$$T = \left\lceil \frac{top\_postotak}{100} \times m \right\rceil \tag{42}$$

    - m predstavlja ukupan broj značajki u ulaznom skupu podataka
4. Inicijalizacija Glasanja
        - Za svaku metodu M, dodjeljuje glasove značajkama u Top T određenom broju značajki:

$$V(f_i) = V(f_i) + (f_i \in Top_t(M)), \forall M \tag{43}$$

5. Glasanje za značajke
        - Za svaku od metode $M \in \{RF, XGB, MI, RFE\}$, uzima se prvih Top T značajki
        - Svaka od Top T značajki dobiva jedan glas na temelju njezinog doprinosa predikcijskim performansama:

$$V(f_i) = V(f_i) + 1, \forall f_i \in Top_t(M) \tag{44}$$

- Nakon iterativnog postupka kroz $M \in \{RF, XGB, MI, RFE\}$, broj glasova za svaku značajku pokazuje koliko je ta značajka bila među najboljima.

6. Biranje značajki s najviše glasova
    - Bira *T* značajke s najvećim brojem glasova *V(fi)*:

$$odabrane\_znacajke = \underset{f_i}{\arg top}\left(V(f_i), T\right) \tag{45}$$

    - Sortira *V(fi)* silaznim redoslijedom od najvećeg do najmanjeg broja *i* odabire *T* značajke
7. Povratna vrijednost metode
    - Reducirani ulazni skup podataka $X_{reduced}$ s odabranim *T* značajkama:

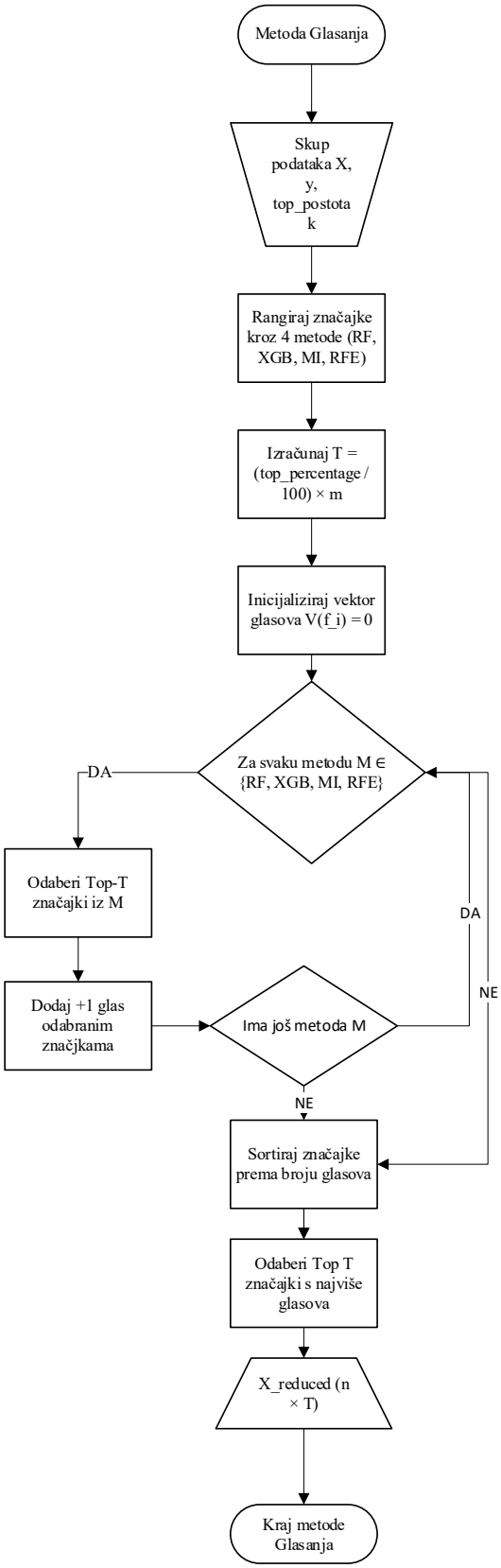
$$X_{reduced} = X[:, odabarne\_znacajke] \tag{46}$$

Primjer rada razvijene metode Glasanja je prikazan u nastavku:

- **Ulaz:**
    - *X*.stupci = ["A", "B", "C", "D"]
    - *top_postotak* = 50% → T = 2
- **Glasanje**
    - Rang liste:
        - *RF*: ["A", "B", "C", "D"]
        - *XGB*: ["B", "A", "D", "C"]
        - *MI*: ["C", "A", "B", "D"]
        - *RFE*: ["D", "B", "A", "C"]
- **Dobiveni glasovi:**
    - A: 3 (Top 2 (T=2) u *RF*, *XGB*, *MI*)
    - B: 3 (Top 2 (T=2) u *RF*, *XGB*, *RFE*)
    - C: 1 (Top 2 (T=2) u *MI*)
    - D: 1 (Top 2 (T=2) u *RFE*)
- **Izlaz:**

- **Odabrane značajke:** ["A", "B"] (jer su prve dvije po broju glasova).

Dijagram toka podatka metode Glasanja prikazan je slikom u nastavku.



Slika 35: Metoda glasanja

U sljedećem poglavlju opisana je razvijena metoda *Ruksaka*, koja se temelji na optimizacijskom pristupu maksimizacije relativne važnost značajki i njihove varijance kao oblika „troška“ unutar unaprijed definiranog kapaciteta ruksaka.

5.2.2.9. Metoda redukcije dimenzionalnosti podataka – metoda Ruksaka

Metoda *Ruksaka* implementira algoritam problem ruksaka za odabir optimalnog skupa značajki na temelju njihove važnosti (značajnosti) i varijance. Metoda *Ruksaka* koristi algoritamski pristup optimizacije kako bi odabrala podskup značajki koji istovremeno maksimizira njihovu važnost za model predikcije strojnog učenja i uzima u obzir njihovu informativnu varijabilnost. Algoritam problem ruksaka, optimizacijski je problem u kojem je cilj odabrati podskup elemenata iz skupa ograničenog kapacitetom, tako da ukupna vrijednost odabranih elemenata bude do maksimalnog ograničenja kapaciteta ruksaka.[52, 53] U ovom slučaju, značajke su elementi ruksaka, njihova važnost predstavlja vrijednost (značajnost), a varijanca se koristi kao težina, gdje se uvodi ograničenje ukupnog kapaciteta značajki koje mogu biti uključene u konačni skup odabranih najznačajnijih značajki.

Metoda *Ruksaka* prima tri ulazna parametra – *X*, *y* i *kapacitet*. Parametar *X* predstavlja skup svih značajki ulaznog skupa podataka, dok je *y* ciljana varijabla koja služi za treniranje modela strojnog učenja. Parametar *kapacitet* postavlja maksimalan dozvoljeni kapacitet ruksaka, odnosno određuje koliko značajki može biti uključeno u konačan odabir. Ova vrijednost kontrolira složenost modela, čime se balansira između performansi i interpretabilnosti. Metoda *Ruksaka* započinje procjenom važnosti značajki korištenjem metode *Random Forest*[25, 26] ili *XGBoost*[27] unutar *Stratified K-Fold* (klasifikacijski zadatak) ili *KFold* (regresijski zadatak) *Cross Validation[32]* metode. Metoda *Stratified K-Fold* (klasifikacijski zadatak) ili *KFold* (regresijski zadatak) *Cross Validation[32, 33]* dijeli podatke na pet podskupova (*K-Fold*=5, *5-fold*), osiguravajući da distribucija ciljne varijable *y* ostane proporcionalna u svakom podskupu, čime smanjuje mogućnost pristranosti u učenju modela strojnog učenja. Unutar svake iteracije RF ili XGB se trenira na trening skupu i evaluira na testnom skupu, pri čemu se značajnosti značajki iterativno akumuliraju kroz svih pet podskupova podataka. Konačne vrijednosti značajnosti značajki se normaliziraju, s čime se osigurava konzistentna usporedba među značajkama.

Nakon procjene važnosti značajki, metoda Ruksaka kreira rang listu, koja uključuje sljedeće informacije:

- naziv značajke,
- važnost značajke prema RF ili XGB modelu strojnog učenja,
- varijanca značajke u skupu podataka.

Kako bi se osigurala usporedivost metrika važnosti i varijance, vrši se normalizacija važnosti i varijance. Normalizirana važnost izračunava se kao udio pojedinačne važnosti u ukupnoj važnosti svih značajki, dok se varijanca normalizira u odnosu na ukupnu varijancu svih značajki. Zatim, normalizirana varijanca skalira se prema unaprijed definiranom kapacitetu (parametar *kapacitet*) kako bi se omogućila konzistentna interpretacija težine unutar izvornog algoritma problem ruksaka[52, 53].

Glavna optimizacija odvija se kroz izvorni algoritam problem ruksaka[52, 53], gdje se značajke promatraju kao predmeti, a njihova važnost i varijanca kao vrijednost i težina. U izvornom dijelu koda algoritma problem ruksaka implementira se dinamičko programiranje putem DP tablice (*dp*), dimenzija *(n+1) × (kapacitet+1)*, gdje *n* označava broj značajki. Svaka ćelija *dp[i][w]* predstavlja maksimalnu moguću važnost koju možemo postići odabirom *i* značajki uz ukupnu dopuštenu težinu *w*.[52, 53] Tijekom iteracije kroz značajke provodi se standardna rekurzija Ruksaka[52, 53]:

$$dp[i][w] = \max\left(dp[i-1][w], dp[i-1][w - težina_i] + vrijednost_i\right) \tag{47}$$

U slučaju da značajka može stati unutar preostalog kapaciteta, izvorni algoritam problem ruksaka odlučuje hoće li ju uključiti na temelju maksimizacije ukupne važnosti. Nakon razvoja DP tablice, slijedi rekonstrukcija optimalnog rješenja putem vraćanja unatrag (engl. *backtracking*), gdje se unatrag prolazi kroz tablicu kako bi se identificirale značajke koje su uključene u optimalni skup.[52, 53] Na kraju, odabrane značajke pohranjuju se u varijablu *odabarane_znacajke*, a metoda *Ruksaka* vraća podatkovni okvir koji sadrži samo odabrane značajke.

Doprinos u odnosu na izvorni algoritam problem ruksaka, se očituje u napisanom programskom kodu odnosno dodanim elementima za redukciju dimenzionalnosti:

- Procjena značajnosti značajki pomoću metode *Random Forest*[25, 26] ili *XGBoost*[27] unutar *Stratified K-Fold* ili *KFold Cross Validation[32, 33]* metode, što u izvornom algoritmu problem ruksaka ne postoji.
- Uvođenje varijance kao težinske funkcije predstavlja doprinos odnosno doradu izvornog algoritma problema ruksaka, gdje se varijanca koristi kao ograničenje kapaciteta količine predmeta (u razvijenoj metodi ovog rada količine značajki) koji se mogu smjestiti u ruksak, što u izvornoj metodi algoritma problema ruksaka nije predviđeno. S takvom razvijenom programskom logikom, proširuje se izvorni algoritam problema ruksaka jer ima dodatno razvijeni kriterij pri donošenju odluka o odabiru predmeta (u ovom radu značajki). Navedena nadogradnja programske logike izvornog algoritma problema ruksaka osigurava da konačni skup odabranih predmeta odnosno značajki ne ovisi isključivo o značajnosti odabranih predmeta/značajki, već i o njihovoj varijabilnosti uključivanjem statističke apsolutne mjere disperzije – Varijance, koja predstavlja prosječni zbroj kvadrata odstupanja vrijednosti obilježja (veličine) od aritmetičke sredine[61]. Normalizacija značajnosti i varijance, kako bi obje vrijednosti bile u istom skaliranom brojčanom rasponu i usporedive pri optimizaciji, što u izvornom algoritmu problema ruksaka ne postoji.
- Ograničenje broja značajki kroz kapacitet, s čim se osigurava da se ne koristi prevelik broj značajki koje mogu uzrokovati prekomjerno prilagođavanje (engl. *overfitting*), što u izvornom algoritmu problema ruksaka ne postoji.

Metoda *Ruksaka*, prilagođenog izvornog algoritma problem ruksaka, omogućuje optimalnu selekciju značajki kombinirajući dodane metode strojnog učenja i optimizacijske tehnike izvornog algoritma problem ruksaka, čime se osigurava ravnoteža između prediktivnih sposobnosti modela i interpretabilnosti podataka. Takav pristup omogućava bolju skalabilnost modela te poboljšava njegovu generalizacijsku sposobnost na neviđenim podatcima – što predstavlja bitnu komponentu u produkcijskom okružju gdje je cilj i koristiti razvijeni *MoziaisMl* sustav.

Metoda Ruksaka se sastoji od 8 procesnih radnji, prikazanih u nastavku:

1. Ulazni podatci
    - *X*: Ulazni skup podataka (matrica sa značajkama, dimenzije $n \times m$).
    - *y*: Izlazna, ciljna varijabla (vektor duljine n)
    - *kapacitet*: Kapacitet ruksaka, određuje korisnik sustava *MoziaisMl*
    - *tip_zadatka:* klasifikacija ili regresija
2. Izračun značajnosti kroz *KFold Cross-Validation:*
    - K, određuje korisnik sustava *MoziaisMl*
    - Svaki *fold k,* trenira $M \in \{RF \vee XGB\}$ i akumulira važnost:

$$F_{total} = \sum_{k=1}^{K} F \begin{matrix} (k) \\ M \end{matrix} \tag{48}$$

    - Prosječna važnost:

$$F_{avg} = \frac{F_{total}}{K} \tag{49}$$

3. Izračun varijance značajki:
    - Kako bi metoda radila, svaka značajka mora imati varijancu:

$$\sigma^2 (X_i) = \frac{1}{m} \sum_{j=1}^{m} \left(X_{ij} - \mu X_i \right)^2 \tag{50}$$

    - $\mu X_i j$ je srednja vrijednost X_i
    - Izrađuje se novi Ulazni skup podataka X : matrica sa značajkama te sad i njihovom važnošću i varijancom, dimenzije $n \times m$).
4. Normalizacija važnosti i varijance značajki te skaliranje varijance značajki
    - Normalizacija važnosti (*F*):

$$F_i = \frac{F_{avg,i}}{\sum_{j=1}^{m} F_{avg,j}} \tag{51}$$

    - Normalizacija varijance (*V*):

$$V_i = \frac{V_i}{\sum_{j=1}^{m} V_j} \tag{52}$$

- Skaliranje varijance:

$$w_i = V_i \times kapacitet \tag{53}$$

5. Prilagodba problema ruksaka za redukciju dimenzionalnosti
   - Predmeti: Svaka značajka f_i je predmet koji može u ruksak i ima svoju:
     - Vrijednost: $v_i = F_i$
     - Težinu: $w_i = V_i \times kapacitet$
   - Cilj razvijene metode: Maksimizirati $\sum v_i$ uz ograničeni kapacitet ruksaka $\sum v_i \leq kapacitet$
6. Dinamičko programiranje algoritma
   - DP tablica izvornog algoritma problem ruksaka[52, 53]:

$$dp[i][w] = \max\left(dp[i-1][w], dp[i-1][w-w_i] + v_i\right) \tag{54}$$

   - Inicijalizacija DP tablice:

$$dp[0][w] = 0 \forall w \tag{55}$$

7. Odabir značajki algoritmom ruksaka
   - Uzimajući u obzir $w = kapacitet$, metoda provjerava za svaki predmet odnosno značajku (f_i) u ulaznom skupu podataka je li je odabrana kao optimalni izbor:

$$Ako\ dp[i][w] \neq dp[i-1][w] => f_i\ dodana \tag{56}$$

   - Ažurira slobodni kapacitet ruksaka:

$$w = w - w_i \tag{57}$$

8. Povratna vrijednost metode

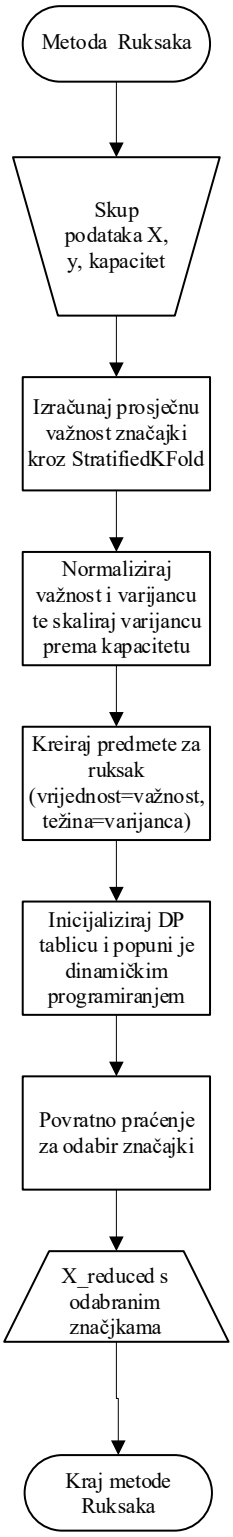
- Reducirani ulazni skup podataka $X_{reduced}$ s značajkama (f_i) koje su odabrane kao optimalni izbor:

$$X_{reduced} = X\left[:, opt\_izabarne\_znacajke\right] \tag{58}$$

Primjer rada razvijene metode Ruksaka je prikazan u nastavku:

- **Ulaz**:
  - X.stupci = ["A", "B", "C", "D"]
  - Kapacitet ruksaka: $kapacitet = 5$
  - Normalizirana važnost: $F = \left[0.4, 0.3, 0.2, 0.1\right]$
  - Normalizirana varijanca: $w = \left[2, 3, 4, 1\right]$
- **Prilagodba**
  - Svaka značajka postaje predmet koji može ići u ruksak
    - Vrijednost: $v_i = F_i$
    - Težina: $w_i = V_i \times kapacitet$
- **Odabir**:
  - Optimalno rješenje: Značajke "A" (v=0.4, w=2) i "B" (v=0.3, w=3) → ukupno w=5≤5, v=0.7.

Dijagram toka podatka metode ruksaka u ulaznom skupu podataka prikazan je slikom u nastavku.



Slika 36: Metoda Ruksaka

Sa sljedećim poglavljem započinje treća faza prilagođene CRISP-DM metodologije, koja se temelji na izgradnji klasifikacijskih i regresijskih modela strojnog učenja s najboljim skupom odabranih hiperparametara.

5.2.3. Faza modeliranja i višekriterijske optimizacije modela i hiperparametara

Faza modeliranja i optimizacije[11, 21, 22, 54] modela strojnog učenja predstavlja dio metodološkog okvira *MoziaisMl* sustava, specifičan za razvoj modela strojnog učenja za predviđanje koji vrši predikciju na skupu podatka i implementira se u produkcijskom okružju. Primarni cilj prilagodbe ove faze je izgradnja klasifikacijskih i regresijskih modela strojnog učenja koji zadovoljavaju postavljene zahtjeve poslovne domene ulaznog skupa podataka reducirane dimenzionalnosti, tehničke specifikacije *MoziaisMl* sustava i eksperiment ovog istraživanja - više brzine za 15% i nepostojanjem smanjenja za i više od 4% točnosti metrika predviđanja. Odabir klasifikacijskih ili regresijskih metoda strojnog učenja i hiperparametara ovisi o karakteristikama problema i specifičnim zahtjevima svakog ulaznog skupa podataka. Zbog toga je napravljen automatizirani cjevovod, naziva algoritam latencije, unutar *MoziaisMl* sustava koji automatizira proces odabira najboljeg klasifikacijskog ili regresijskog modela i optimizacije hiperparametara za svaki ulazni skup podataka. Za podešavanje hiperparametara, koristi se metoda *GridSearchCV*[31] s 3 podskupa (*3-Fold*) metode *Cross-validation*[62] omogućujući efikasno istraživanje prostora hiperparametara i poboljšanje performansi klasifikacijskih ili regresijskih metoda strojnog učenja. Provedbom predistraživanja pokazalo se kako je metoda *GridSearchCV*[31] s 3 podskupa (*3-Fold*) metode *Cross-validation*[62] najučinkovitija za korištenje u *MoziaisMl* sustavu. Nastavno, metoda koja je najučinkovitija je i korištena u istraživanju ovoga rada. Kod odabira metoda *MoziaisMl* sustav ima na raspolaganju više klasifikacijskih ili regresijskih metoda strojnog učenja jer bez obzira na automatizaciju, u metodološkom okviru, a i samom *MoziaisMl*-u vodi se računa o sekundarnoj validaciji modela metode strojnog učenja, što podrazumijeva provjeru generalizacijske sposobnosti na neovisnim podatcima kako bi se osiguralo da model nije prekomjerno prilagođen. Osim sekundarne validacije modela metode strojnog učenja, verzioniraju se i skupovi podataka jer na modelu metoda strojnog učenja treba izvršavati predikciju na skupu podataka reducirane dimenzionalnosti i skupu podataka gdje redukcija dimenzionalnosti nije izvršena. S verzioniranjem podataka optimizira se provođenje eksperimenta ovog istraživanja.[11, 21, 22, 54]

Provedbom predistraživanja pokazalo se koje su klasifikacijske i regresijske metode te postavke klasifikacijskih/regresijskih metoda najučinkovitije za korištenje u

MoziaisMl-u. Nastavno, metode koje su se pokazale učinkovite korištene su u istraživanju ovoga rada i prilagođenoj fazi metodologije naziva Modeliranje i optimizacija modela.

U nastavku su navedene klasifikacijske metode strojnog učenja sa svojim specifičnim parametrima koji kontroliraju način na koji model uči iz skupa podataka i prilagođava svoju strukturu u svrhu postizanja optimalne točnosti i generalizacijske sposobnosti:

1. *DecisionTreeClassifier*[63-66] - koristi sljedeće parametre za definiranje strukture stabla odlučivanja:
    - *criterion* određuje način mjerenja kvalitete podjele čvorova i može poprimiti vrijednosti Gini indeks (*'gini'*) ili entropija (*'entropy'*).
    - *max_depth* ograničava maksimalnu dubinu stabla, s čim sprječava prekomjerno prilagođavanje, a može imati vrijednosti neograničene dubine (*None*) ili brojčane vrijednosti 10, 20 i 30.
    - *max_features* definira maksimalni broj značajki koje se koriste za određivanje podjele u čvorovima, gdje vrijednost *None* označava sve značajke, dok vrijednosti '*sqrt*' i '*log2*' definiraju broj značajki kao kvadratni korijen ili logaritam baze 2 od ukupnog broja značajki.
    - *class_weight* omogućuje balansiranje klasa, gdje vrijednost *None* znači da sve klase imaju jednaku težinu, a vrijednost '*balanced*' ponderira težine obzirom na učestalost klasa.
2. *GaussianNB*[67] - koristi *Naive Bayes*[68] klasifikator s normalnom distribucijom s parametrima:
    - *priors* omogućuje definiranje apriornih vjerojatnosti za svaku klasu, vrijednost *None.*
    - *var_smoothing* kontrolira dodavanje male konstante radi stabilizacije izračuna varijance, s vrijednostima 1e-8, 1e-9, 1e-10, čime se sprječava dijeljenje s nulom i povećava brojčana stabilnost.
3. *RandomForestClassifier[25]* - proširuje funkcionalnost stabala odlučivanja kroz ansembliranje više stabala sa sljedećim parametrima:
    - *n_estimators* definira broj stabala u šumi, gdje veći broj može poboljšati stabilnost predikcija, a vrijednosti su 10, 50, 100.

- *max_depth* ograničava maksimalnu dubinu stabla, s čim sprječava prekomjerno prilagođavanje, a može imati vrijednosti neograničene dubine (*None*) ili brojčane vrijednosti 10, 20 i 30.
- *max_features* definira maksimalni broj značajki koje se koriste za određivanje podjele u čvorovima, gdje vrijednost *None* označava sve značajke, dok vrijednosti '*sqrt*' i '*log2*' definiraju broj značajki kao kvadratni korijen ili logaritam baze 2 od ukupnog broja značajki.
- *class_weight* omogućuje balansiranje klasa, gdje vrijednost *None* znači da sve klase imaju jednaku težinu, a vrijednost '*balanced*' ponderira težine obzirom na učestalost klasa.

4. *SVC (Support Vector Classifier)[69]* - koristi parametre:
    - *kernel* za definiranje jezgrine funkcije kojom se podatci transformiraju u višu dimenziju, pri čemu su opcije linearni razdjelnici ('*linear*') i radijalna bazna funkcija ('*rbf*').
    - *C* predstavlja regularizacijski hiperparametar koji kontrolira kompromis između točne klasifikacije i složenosti modela, s mogućim vrijednostima 0.1, 1, 10.
    - *gamma* određuje koliko daleko pojedinačni primjer utječe na granicu odlučivanja i može imati vrijednosti *'scale'*, *'auto'* ili sljedeće brojčane vrijednosti 0.1, 1, 10.
5. *GradientBoostingClassifier[70]* - koristi parametre:
    - *loss* za definiranje funkcije gubitka - '*deviance*' za *log-loss* funkciju i '*exponential*' za *AdaBoost* ekvivalent.
    - *learning_rate* određuje koliko brzo model uči, sa sljedećim brojčanim vrijednostima 0.01, 0.1, 0.2, 0.3.
    - *n_estimators* kontrolira broj stabala u skupu,
    - *criterion* definira način mjerenja podjele čvorova - '*friedman_mse*', '*mse*' i '*mae*'.
    - *max_depth* ograničava maksimalnu dubinu stabla, s čim sprječava prekomjerno prilagođavanje, a može imati vrijednosti neograničene dubine (*None*) ili brojčane vrijednosti 10, 20 i 30.

- *max_features* definira maksimalni broj značajki koje se koriste za određivanje podjele u čvorovima, gdje vrijednost *None* označava sve značajke, dok vrijednosti '*sqrt*' i '*log2*' definiraju broj značajki kao kvadratni korijen ili logaritam baze 2 od ukupnog broja značajki.

6. *XGBClassifier*[27] - varijanta *Gradient Boosting*-a, koristi hiperparametre:
    - *criterion* određuje način mjerenja kvalitete podjele čvorova i može poprimiti vrijednosti Gini indeks (*'gini'*) ili entropija (*'entropy'*).
    - *max_depth* ograničava maksimalnu dubinu stabla, s čim sprječava prekomjerno prilagođavanje, a može imati vrijednosti neograničene dubine (*None*) ili brojčane vrijednosti 10, 20 i 30.
    - *max_features* definira maksimalni broj značajki koje se koriste za određivanje podjele u čvorovima, gdje vrijednost *None* označava sve značajke, dok vrijednosti '*sqrt*' i '*log2*' definiraju broj značajki kao kvadratni korijen ili logaritam baze 2 od ukupnog broja značajki.
    - *class_weight* omogućuje balansiranje klasa, gdje vrijednost *None* znači da sve klase imaju jednaku težinu, a vrijednost '*balanced*' ponderira težine obzirom na učestalost klasa.
7. *KNeighborsClassifier*[71] - koristi metodu najbližih susjeda (engl. *k-Nearest Neighbors*, KNN). Ima sljedeće parametre i vrijednosti:
    - *n_neighbors* definira broj najbližih susjeda uzetih u obzir za klasifikaciju, gdje su brojčane vrijednosti 3, 5, 7, 9 i 11.
    - *weights* određuje način ponderiranja udaljenosti susjeda - *uniform* - jednaka težina i *distance* - veća težina bližim susjedima.
    - *algorithm* definira algoritam pretraživanja - *auto, ball_tree, kd_tree, brute*.
    - *p* određuje udaljenost u Minkowskom prostoru: 1 - *Manhattan*, 2 - *Euclidean*,
    - *metric* dodatno specificira funkciju udaljenosti - *euclidean*, *manhattan*, *minkowski*.
8. *MLPClassifier (Multi-Layer Perceptron)[72]* - koristi sljedeće parametre:

- *hidden_layer_sizes* za definiranje broja i veličine skrivenih slojeva neuronske mreže - (50,) (jedan sloj s 50 neurona), (100,), (50, 50) (dva sloja po 50 neurona) i (100, 100).
- *activation* definira aktivacijsku funkciju - *logistic, tanh, relu*.
- *solver* specificira optimizacijski algoritam - *adam*.
- *alpha* regulira težinu L2 regularizacije - 0.0001, 0.05.
- *learning_rate* kontrolira prilagodbu stope učenja - *constant, adaptive*.
- max_iter definira maksimalni broj iteracija - 500.

Nakon navođenja klasifikacijskih metoda strojnog učenja i njihovih specifičnih parametara, u nastavku su navedene regresijske metode strojnog učenja sa svojim specifičnim parametrima za optimizaciju predikcija kontinuiranih vrijednosti:

1. *LinearRegression*[73] - linearni model koji procjenjuje odnos između nezavisnih varijabli (*X* skupa podataka) i zavisne varijable (*y* ciljne varijable) minimiziranjem srednje kvadratne pogreške. Ima sljedeće parametre:
    - *fit_intercept* s vrijednošću *True* određuje hoće li se uklopiti presjek regresijske jednadžbe.
    - *copy_X* s vrijednošću *True* osigurava kopiranje ulaznog skupa značajki kako bi se spriječile neželjene promjene.
    - *n_jobs* s vrijednošću *None* definira broj korištenih procesorskih jezgri.
    - *positive* s vrijednošću *False* omogućuje restrikciju na pozitivne koeficijente.
2. *Lasso*[74] regresija - proširuju linearnu regresiju primjenom regularizacije. Koristi L1 penalizaciju te ima sljedeće parametre:
    - *alpha* koristi za regulaciju složenosti modela uklanjanjem manje značajnih značajki s rasponom brojčanih vrijednosti 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100, 1000, 10000.
3. *Ridge*[75] regresija - proširuju linearnu regresiju primjenom regularizacije. Koristi L2 penalizaciju te ima sljedeće parametre:
    - *alpha* s rasponom brojčanih vrijednosti 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100, 1000, 10000 smanjuje utjecaj pojedinačnih koeficijenata i time smanjuje varijabilnost modela.

4. *ElasticNet*[76] - kombinira L1 i L2 regularizaciju, s parametrima:
    - *alpha* kontrolira jačinu regularizacije s rasponom brojčanih vrijednosti 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100, 1000, 10000.
    - *l1_ratio* s rasponom brojčanih vrijednosti od 0 do 1 određuje omjer između L1 i L2 penalizacije.
5. *SVR (Support Vector Regression)*[77] - koristi kernelske metode za modeliranje složenih nelinearnih odnosa. Ima sljedeće parametre:
    - *kernel* s vrijednostima *'linear'* ili *'rbf'* određuje transformaciju podataka u višu dimenziju.
    - *C* s vrijednostima 0.1, 1, 10 kontrolira kompromis između složenosti modela i pogreške.
    - *gamma* s vrijednostima '*scale*', '*auto*', 0.1, 1, 10 regulira utjecaj pojedinačnih instanci na funkciju odlučivanja.
6. *KNeighborsRegressor*[78] - temelji se na metodi najbližih susjeda (KNN), s parametrima:
    - *n_neighbors* s vrijednostima 3, 5, 7, 9 ili 11 definira broj susjeda koji doprinose predikciji.
    - *weights* s vrijednostima *'uniform'* ili *'distance'* određuje hoće li svi susjedi imati jednaku važnost ili će oni bliži imati veći utjecaj.
    - *metric* s vrijednostima *'euclidean', 'manhattan', 'minkowski'* definira način izračuna udaljenosti među uzorcima.
7. *XGBRegressor*[27] - koristi gradijentno pojačanje stabala odlučivanja (metoda *XGBoost*). Ima sljedeće parametre:
    - *n_estimators* s vrijednostima 100, 400 ili 800 određuje broj stabala u modelu.
    - *learning_rate* s vrijednostima 0.05, 0.1 ili 0.20 kontrolira brzinu učenja.
    - *max_depth* s vrijednostima 3, 6 ili 9 definira maksimalnu dubinu stabala.
    - *min_child_weight* s vrijednostima 1, 10, 100 postavlja prag za minimalan broj uzoraka u čvorovima kako bi se spriječilo prekomjerno prilagođavanje.

- *tree_method* s vrijednošću *'gpu_hist'* optimizira izvođenje modela korištenjem grafičke kartice računala.

8. *MLPRegressor (Multilayer Perceptron)[79]* - koristi višeslojne neuronske mreže za modeliranje složenih nelinearnih odnosa, sa sljedećim parametrima:
    - *hidden_layer_sizes* s vrijednostima 50, 100, (50, 50), (100, 100) određuje broj slojeva i neurona u mreži.
    - *activation* s vrijednostima *'logistic', 'tanh', 'relu'* definira funkciju aktivacije za neuronske jedinice.
    - *solver* s vrijednošću *'adam'* upravlja optimizacijom težina mreže.
    - alpha s vrijednostima 0.0001 ili 0.05 kontrolira L2 regularizaciju kako bi se spriječilo prekomjerno prilagođavanje.
    - *learning_rate* s vrijednostima *'constant'* ili *'adaptive'* određuje način prilagodbe stope učenja kroz epohe.
    - *max_iter* s vrijednošću 500 postavlja maksimalan broj iteracija za konvergenciju modela.
9. *DecisionTreeRegressor[80]* - koristi stabla odlučivanja za predikciju kontinuiranih vrijednosti, s parametrima:
    - *max_depth* s vrijednostima *None*, 10, 20, 30 određuje maksimalnu dubinu stabla.
    - *max_features* s vrijednostima *None, 'auto', 'sqrt', 'log2'* kontrolira broj značajki dostupnih u svakoj podjeli čvorova.
    - *ccp_alpha* s vrijednostima 0.0, 0.1, 0.2 ili 0.3 primjenjuje podrezivanje temeljeno na ravnoteži troška i kompleksnosti (engl. *cost-complexity pruning*), što omogućuje kontrolu složenosti stabla.
10. *RandomForestRegressor*[26] - koristi više stabala odlučivanja kako bi smanjio varijancu modela, s parametrima:
    - *n_estimators* s vrijednostima 10, 50 ili 100 određuje broj stabala.
    - *max_depth* s vrijednostima *None*, 10, 20 ili 30 regulira složenost modela.
    - *max_features* s vrijednostima *None, 'auto', 'sqrt'* ili *'log2'* postavlja broj značajki koje se koriste pri generiranju svakog stabla.

- *ccp_alpha* s vrijednostima 0.0, 0.1, 0.2 ili 0.3 služi za rezanje stabala (engl. pruning) i smanjuje složenost modela uklanjanjem čvorova koji doprinose samo maloj poboljšanoj točnosti.

11. *GradientBoostingRegressor[81]* - koristi gradijentno pojačanje za optimizaciju predikcija. Ima sljedeće parametre:
    - *n_estimators* s vrijednostima 10, 50 ili 100 definira broj stabala.
    - *learning_rate* s vrijednostima 0.001, 0.01 ili 0.1 određuje brzinu ažuriranja modela.
    - *subsample* s vrijednostima 0.5, 0.75 ili 1.0 postavlja postotak podataka korištenih za treniranje svakog stabla.
    - *max_depth* s vrijednostima 3, 10 ili 20 regulira složenost stabala.
    - *alpha* s vrijednostima 0.1, 0.5 ili 0.9 upravlja s kvantilnom regresijskom funkcijom gubitka (engl. *quantile regression loss*).

Prethodno navedene klasifikacijske i regresijske metode strojnog učenja u fazi Modeliranje i optimizacija modela pokrivaju širok spektar tehnika od klasičnih modela strojnog učenja do naprednih neuronskih mreža i ansamblerskih metoda. Navedeni klasifikacijski i regresijski modeli te njihovi pripadajući hiperparametri pohranjeni su u zasebno razvijene metode *klas_modeli_i_parametri* i *reg_modeli_i_parametri* koje vraćaju unaprijed definirane skupove klasifikacijskih i regresijskih modela s pripadajućim hiperparametrima. Navedene, razvijene, metode automatiziranim putem dohvaćaju modele koji su pogodni za određeni tip zadatka – klasifikacijski ili regresijski, eliminirajući potrebu za ručnim definiranjem modela i njihovih hiperparametara. Prethodno navedene metode se pozivaju u razvijenoj metodi *grid_pretraga*, metodi koja će biti objašnjena u nastavku.

Faza Modeliranje i optimizacija modela predstavlja automatizirani metodološki i razvijeni okvir unutar *MoziaisMl* sustava za odabir i optimizaciju modela strojnog učenja primjenom metode *GridSearchCV[31]*. Cilj i svrha navedene faze je omogućavanje sustavnog pretraživanja i evaluacije različitih modela te odabir optimalnih vrijednosti hiperparametara kako bi se postigle najbolje performanse modela - eksperimentira se s različitim metodama i algoritmima kako bi se pronašlo rješenje koje najbolje odgovara, klasifikacijskom ili regresijskom, zadatku strojnog učenja.

U početku metodološke faze Modeliranje i optimizacija modela, pohranjuju se informacije o modelima i njihovim pripadajućim hiperparametrima u varijable tipa rječnika *modeli* i *parametri*. Nastavno, definiraju se ključni parametri za pohranu rezultata pretrage *grid_pretraga* i *najbolji_rezultati,* omogućujući kasniju analizu modela i njihovih performansi. Kako bi se osigurala dosljednost i mogućnost pohrane rezultata, dodaje se vremenska oznaka, koja omogućuje verzioniranje eksperimenta.

Ključni element faze Modeliranje i optimizacija modela je razvijena metoda *grid_pretraga*, koja provodi s metodom *GridSearchCV[31]* pretragu hiperparametara za sve modele u varijabli, tipa rječnik, *modeli*. Razvijena metoda inicijalizira instancu *GridSearchCV[31]* za svaki model, predavajući joj pripadajuće hiperparametre u varijablu, tipa rječnik, *parametri* i pokreće unakrsnu validaciju na skupu podataka - *X* (ulazni skup podataka) i *y* (ciljana varijabla). Nakon završetka pretrage, razvijena metoda ispisuje najbolji model, najbolje parametre i najbolji ostvareni rezultat, što omogućuje brzu analizu i usporedbu različitih modela jer postizanje brzine jest svrha ovog istraživanja. Na taj način, u navedenoj fazi, osiguran je proces selekcije modela, pri čemu se svi modeli testiraju pod istim uvjetima.

Osim same evaluacije modela, faza Modeliranje i optimizacija modela omogućuje pohranu najboljih rezultata i modela pomoću razvijene metode *pohrana*. Razvijena metoda sprema rezultate optimizacije u serijalizirane datoteke CSV, *pkl[82]* i *json[83]*. Time se omogućuje ponovna upotreba optimiziranih modela bez potrebe za ponovnim izvođenjem pretrage hiperparametara, čime se štedi vrijeme i resursi odnosno postiže brzina - što jest svrha ovog istraživanja.

Kako bi se omogućila strukturna analiza performansi modela, implementirana je razvijena metoda *rezultat_sazetak*, razvijena metoda koja generira sažetak rezultata pretrage modela. Navedena razvijena metoda prolazi kroz sve modele, prikuplja metapodatke poput vremena treniranja, vremena predikcije, broja razdjela u unakrsnoj validaciji i sprema rezultate u CSV, *json* i *pkl* formate. Rezultati su redoslijedno sortirani silaznom putanjom, prvo, prema metrici prediktivna točnost (ili greška), zatim, brzina, što omogućuje brže donošenje odluke pri odabiru modela za konačnu implementaciju u sljedeću fazu naziva Evaluacija.

Zaključno, faza Modeliranje i optimizacija modela predstavlja sveobuhvatan metodološki okvir za optimizaciju modela strojnog učenja, omogućujući automatizirano testiranje, usporedbu i odabir modela kroz strukturirani proces pretrage hiperparametara. Implementacijom programske logike odnosno mehanizama za pohranu i analizu rezultata, navedena faza značajno pojednostavljuje eksperimentalnu fazu razvoja modela, osiguravajući reproducibilnost i efikasnost procesa evaluacije strojnog učenja nad klasifikacijskim i regresijskim zadatcima. Navedena faza koristi osnovnu funkcionalnost pretrage hiperparametara i unakrsne validacije preuzetu iz metode *GridSearchCV[31]*, navedena faza s vlastito razvijenim metodama proširuje i nadograđuje primjenu metode *GridSearchCV[31]* kroz dodatnu automatizaciju, organizaciju rezultata i pohranu modela. Glavna funkcionalnost metode *GridSearchCV[31]*, koja je izravno preuzeta, uključuje:

- Pretragu optimalnih vrijednosti hiperparametara unutar unaprijed definiranog skupa modela i parametara.
- Unakrsnu validaciju kako bi se osiguralo objektivno vrednovanje modela na različitim podskupovima podataka.
- Vraćanje najboljih hiperparametara (parametar metode *GridSearchCV[31]*: *best_params_*), najboljeg modela (parametar metode *GridSearchCV[31]*: *best_estimator_*) i najboljeg postignutog rezultata (parametar metode *GridSearchCV[31]*: *best_score_*).

Međutim, faza Modeliranje i optimizacija modela ima razvijenu dodatnu logiku koja nije dio metode *GridSearchCV[31]*, čime se omogućava bolja organizacija i upravljanje eksperimentima nad modelima metoda i hiperparametara strojnog učenja:

1. Automatizacija višestruke pretrage modela – umjesto ručnog pokretanja metode *GridSearchCV[31]* za svaki model, razvijena metoda, navedene faze, iterira kroz više modela i automatski ih optimizira, čime značajno smanjuje potrebu za ručnom intervencijom.
2. Pohrana i dokumentacija rezultata – rezultati pretrage spremaju se u varijable, tipa rječnik, *grid_pretraga* i *najbolji_rezultati,* a zatim se serijaliziraju u formate CSV, *pkl* i *JSON*. Ova funkcionalnost omogućuje reproducibilnost eksperimenata i

smanjuje potrebu za ponovnim izvođenjem vremenski zahtjevnih pretraga prostora vrijednosti hiperparametara.

3. Generiranje sažetka performansi modela (razvijena metoda *rezultat_sazetak*) – za razliku od metode *GridSearchCV[31]*, koja pruža samo osnovne informacije o rezultatima pretrage, ova metoda dodatno prikuplja vrijeme treniranja, vrijeme evaluacije, broj iteracija i trajanje pretrage, omogućujući detaljnu analizu efikasnosti modela.
4. Klasifikacija i regresija u jednom procesu faze metodološkog okvira – razvijene metode *klas_modeli_i_parametri* i *reg_modeli_i_parametri* omogućuju jednostavno dohvaćanje unaprijed definiranih modela i njihovih hiperparametara, čime se ubrzava postavljanje eksperimenata i osigurava dosljednost u istraživanju ovog rada.

Razvijena programska logika preuzima jezgru funkcionalnosti metode *GridSearchCV[31]*, ali je proširuje stvarajući modularnu, sustavnu i skalabilnu fazu metodološkog okvira za modeliranje i optimizaciju modela strojnog učenja. S automatizacijom, pohranom podataka i generiranjem analitičkih sažetaka, prethodno navedena i stvorena faza omogućuje učinkovito eksperimentiranje i usporedbu modela, osiguravajući optimalnu konfiguraciju za produkcijsku primjenu.

U sljedećoj fazi, naziva Evaluacija, metodološkog okvira primijenit će se samo jedan odabrani model strojnog učenja iz razvijene metode *rezultat_sazetak* s unaprijed definiranim vrijednostima hiperparametara, umjesto raspona vrijednosti koji se koristi u postupku optimizacije faze Modeliranje i optimizacija modela. Primjer, za klasifikacijski zadatak, ako se kroz fazu Modeliranje i optimizaciju modela, u razvijenoj metodi *rezultat_sazetak,* utvrdi kao najbolji odabir, odabire se *RandomForestClassifier* s precizno postavljenim parametrima: *n_estimators=50, criterion='gini', max_depth=20, max_features=None* i *class_weight='balanced'*. Ovakav pristup omogućuje brže izvođenje evaluacije, jer se koristi najbolji model s optimalno konfiguriranim parametrima, čime se izbjegava dodatna računalna složenost povezana s pretraživanjem prostora vrijednosti hiperparametara odnosno postiže se potrebna brzina. Takav, najbolji model s optimalno konfiguriranim parametrima, koristiti će u produkcijskom okružju bez potrebne za stalnim pretraživanjem prostora vrijednosti hiperparametara.

5.2.4. Automatizirana faza evaluacije s latencijom i predikcijom

Nakon faze Modeliranja i optimizacije modela slijedi faza Evaluacije, čiji je primarni cilj procjena performansi modela na neovisnom testnom skupu podataka. Faza Evaluacije omogućuje kvantifikaciju generalizacijske sposobnosti modela odnosno njegovu sposobnost ispravnog predviđanja nepoznatih primjera koji nisu korišteni tijekom faze Modeliranja i optimizacije modela. Ova analiza omogućuje identifikaciju mogućih nedostataka u modelu te njegovu prilagodbu prije implementacije u produkcijsko okružje. U slučaju da su rezultati faze Evaluacije u skladu s očekivanim performansama i zadanim pragovima kvalitete, model se može smatrati dovoljno pouzdanim za operativnu primjenu u produkcijskom okružju.[11, 21, 22, 54]

Evaluacija predstavlja važnu fazu primjene *MoziaisMl* sustava u produkcijskom okružju, u kojoj se ispituje generalizacijska sposobnost, brzina izvršavanja i prediktivna preciznost modela. Nakon inicijalne analize podataka – uključujući njihov prihvat, analizu značajki, transformaciju i prilagodbe skupa podataka te nakon modeliranja i optimizacije modela strojnog učenja – razvijeni se model primjenjuje na testnom skupu podataka. Faza evaluacije omogućuje validaciju učinkovitosti modela, čime se osiguralo da model ispunjava postavljene zahtjeve istraživanja te da je spreman za primjenu u produkcijskom okružju.

U okviru evaluacijske faze primjene *MoziaisMl* sustava, provedena je sustavna analiza modela strojnog učenja, s ciljem kvantifikacije učinkovitosti po pitanju brzine i prediktivne preciznosti. Rezultati faze evaluacije omogućuju identifikaciju potencijalnih ograničenja modela, a shodno tome i preporuka za buduća istraživanja.

Faza evaluacije izvodi se na klasifikacijskim i regresijskim zadatcima analize podataka, pri čemu se ispituje generalizacijska sposobnost modela strojnog učenja. Za klasifikacijske zadatke, primjenjuju se metode klasifikacije na skupovima podataka koji sadrže diskretnu ciljnu varijablu, gdje model predviđa jednu ili više klasa. Nasuprot tome, regresijski zadatci uključuju predikciju kontinuirane ciljne varijable, koristeći modele strojnog učenja optimizirane za aproksimaciju funkcionalnih odnosa između prediktora i izlazne varijable. Evaluacija performansi provodi se primjenom metrika, kao što su brzina izvedbe (vrijeme inferencije), točnost za klasifikacijske modele te vrijeme inferencije i korijenska srednja kvadratna pogreška (RMSE) za regresijske modele.

Za procjenu performansi modela u različitim scenarijima, koristili su se različiti skupovi podataka iz Kaggle baze[55] za analizu, uključujući trening, validacijske i testne skupove podataka. Cilj faze evaluacije je analizirati model u uvjetima koji simuliraju stvarne primjene, čime se ispituje njegova generalizacijska sposobnost i otpornost na varijacije u podatcima. Ovakav analitički proces razvijenog *MoziaisMl* sustava omogućuje kvantifikaciju brzine i točnosti modela te njegove ukupne učinkovitosti, čime se osiguralo da model zadovoljava zadane kriterije kvalitete prije implementacije u produkcijsko okružje.[22]

5.2.5. Implementacija i integracija MoziaisMl sustava u produkcijsko okružje

U fazi Implementacija ključno je pratiti performanse i identificirati degradaciju modela. Nakon implementacije modela u produkcijsko okružje, potrebno je kontinuirano nadzirati njegovu učinkovitost i osigurati pravovremenu prilagodbu kako bi se održala njegova brzina i sposobnost prediktivne preciznosti. Kada se distribucija podataka mijenja u odnosu na one korištene tijekom treniranja, rezultat je postupno smanjenje performansi modela. U slučaju da se takav pad performansi ne prepozna na vrijeme, model može postati nepouzdan i generirati netočne ili pristrane predikcije. Ukoliko analitički informacijski sustav temeljen na metodama strojnog učenja nije unaprijed konfiguriran da detektira ovakve promjene kroz automatizirane alarme i upozorenja, degradacija modela može proći nezapaženo tijekom duljeg vremenskog razdoblja, što može dovesti do ozbiljnih poslovnih posljedica. Uspostava automatiziranog praćenja performansi modela i integracija mehanizama rane detekcije degradacije bitni su čimbenici u osiguravanju dugoročne pouzdanosti modela u dinamičnim i promjenjivim produkcijskim okružjima. [11, 21, 22, 54]

Analitički informacijski sustav temeljen na metodama strojnog učenja iz ovog rada, naziva *MoziaisML* sustavno bilježi događaje i ima razvijene mehanizme za obradu iznimki s ciljem pravovremene detekcije i dijagnosticiranja potencijalnih problema u modelu strojnog učenja, čime osigurava dugoročnu pouzdanost modela u dinamičnom i promjenjivom produkcijskom okružju.

Implementacija optimalnog modela strojnog učenja u produkcijsko okružje popraćena je usporednom analizom s prihvaćenim i popularnim tehnikama redukcije dimenzionalnosti - PCA[14-16], t-SNE[14-17], UMAP[18] i autoenkoderskim

neuronskim mrežama[19]. Faza Implementacija procjenjuje generalizacijske sposobnosti vlastito razvijenih metoda redukcije dimenzionalnosti *Glasanja* i *Ruksaka* na heterogenim skupovima podataka te ispituje otpornosti vlastito razvijenih metoda redukcije dimenzionalnosti *Glasanja* i *Ruksaka* na promjene u strukturama podataka. Strukture podataka su iz različitih domenskih područja i rješavaju različite klasifikacijske i regresijske probleme.

U sljedećem poglavlju predstavljeni su rezultati primjene metodološkog okvira ovog istraživanja, koji je razvijen prema prilagođenim fazama CRISP-DM metodologije. Analizirani rezultati procesiranja podataka omogućuju procjenu učinkovitosti predloženog metodološkog okvira, kao i njegovu primjenjivost u istraživanju ovog rada.

6. PRIMJENA I EVALUACIJA MOZIAISML SUSTAVA NA KLASIFIKACIJSKIM I REGRESIJSKIM ZADATCIMA

U ovom poglavlju predstavljeni su rezultati istraživanja proizašli iz implementacije metodološkog okvira temeljenog na prilagođenim fazama CRISP-DM metodologije. Kroz analizu rezultata, razmatrani se ključni elementi modeliranja, evaluacije i interpretacije podataka, pri čemu se naglasak stavlja na valjanost, pouzdanost i mogućnost generalizacije predloženog metodološkog okvira i programskog rješenja – *MoziaisMl* sustava. Strukturiranom razradom svake faze prilagođenog metodološkog okvira, prikazani su empirijski rezultati koji omogućuju procjenu učinkovitosti predloženog metodološkog okvira te njegovog znanstvenog doprinosa u području teme ovog istraživanja.

6.1. Praktična provedba zadataka u automatiziranom MoziaisMl sustavu

U ovom potpoglavlju predstavljeni su i detaljno analizirani svi skupovi podataka koji se koriste za rješavanje klasifikacijskih i regresijskih zadataka. Svaki skup podataka sustavno je analiziran i evaluiran, pri čemu se u fazi implementacije i održavanja rezultati uspoređuju s tehnikama redukcije dimenzionalnosti - PCA[14-16], t-SNE[14-17], UMAP[18] te autoenkoderskim neuronskim mrežama[19]. Nastavno, prilagođene faze CRISP-DM metodologije primijenjene su na sve analizirane skupove podataka, gdje se kroz sljedeća potpoglavlja detaljno prikazuje njihova razrada iz poglavlja Metodologija, potpoglavlja Prilagodba CRISP-DM faza na stvarnim primjerima skupova podatka [11, 21, 22, 54].

6.1.1. Poslovno definiranje analitičkih zahtjeva i strukturiranje skupova podataka

Navedena faza u *MoziaisML* sustavu uključuje definiranje poslovnog problema, njegovih ključnih metrika uspjeha, prikupljanje i opis podataka te pripremu za pretprocesiranje. [11, 21, 22, 54]

Skupovi podataka za prikaz provjere rezultata smanjenja dimenzionalnosti u okviru hipoteza i istraživačkih ciljeva su iz prijavljenih *Kaggle* natjecanja i repozitorija:

1. *Titanic - Machine Learning from Disaster*[56],
2. *House Prices - Advanced Regression Techniques*[84],
3. *Multi-class Classifiers: Which Vaccine Was It?*[85].

6.1.1.1.Utvrđivanje domenskog problema i formulacija tipa problema

Titanic skup podataka

Analiza Titanic skupa podataka, izvornog naziva s *Kaggle*-a, *Titanic - Machine Learning from Disaster* skupa podataka temelji se na prediktivnom modeliranju preživljavanja putnika koristeći dva podskupa podataka s informacijama o pojedinim putnicima, uključujući njihovu dob, spol, socio-ekonomski status i druge relevantne varijable. Prvi skup podataka, *train.csv*, sadrži zapise za 891 putnika te uključuje poznate ishode odnosno informaciju o preživljavanju svakog pojedinca, što predstavlja referentnu vrijednost (engl. *ground truth*). S druge strane, *test.csv* obuhvaća slične značajke, ali ne sadrži informaciju o ishodu preživljavanja, čime se postavlja analitički izazov prediktivne klasifikacije. Poslovni problem iz navedenog skupa podataka odnosi se na razvoj modela strojnog učenja koji će, temeljem obrazaca prepoznatih u *train.csv* skupu podataka, moći predvidjeti sudbinu preostalih 418 putnika iz *test.csv* skupa. [56]

Skup podataka s cijenama nekretnina

Analiza skupa podataka s cijenama nekretnina, izvornog naziva s *Kaggle*-a, *House Prices - Advanced Regression Techniques* temelji se na *Ames Housing* skupu podataka, koji je sastavio Dean De Cock s ciljem unapređenja vještina u inženjerstvu značajki i naprednim regresijskim tehnikama. Navedeni skup podataka predstavlja suvremenu i proširenu alternativu *Boston Housing* skupu podataka, čineći ga pogodnim za primjenu i evaluaciju različitih modela strojnog učenja. Sadrži ukupno 79 ulaznih/nezavisnih varijabli koje detaljno opisuju gotovo sve aspekte stambenih objekata u Amesu, Iowa. Izazov ovog skupa podataka sastoji se u predikciji konačne prodajne cijene svake nekretnine (ciljna varijabla), pri čemu su modeli izloženi međuodnosima

varijabli i zahtjevima regresijske analize. Poslovni problem iz navedenog skupa podataka odnosi se na razvoj modela strojnog učenja koji će, temeljem obrazaca prepoznatih u *train.csv* skupu podataka, moći predvidjeti tržišnu cijenu nekretnina iz proširenog *test.csv[86]*, koji uključuje 50.459 redova, skupa. [84]

Skup podataka s cjepivima

Analiza skupa podataka s cjepivima, izvornog naziva s *Kaggle*-a, *Multi-class Classifiers: Which Vaccine Was It?* temelji se na zadatku izgradnje višeklasnog klasifikacijskog modela strojnog učenja koji će identificirati koji je tip cjepiva primijenjen na svakom mišu unutar skupa podataka. Prvi skup podataka, *train.csv*, sadrži zapise za 149 miša te uključuju podatke koji prate niz ciljanih organa i fizioloških parametara, kako bi se steklo dublje razumijevanje učinka virusa i imunološkog odgovora organizma. Drugi skup podataka je prošireni *test.csv[87]* koji uključuje 50.000 miševa bez sadržaja informacije o primijenjenom cjepivu nad njima odnosno bez ciljne varijable *y*. Poslovni problem iz navedenog skupa podataka odnosi se na primjenu višeklasne klasifikacije strojnog učenja za razvoj algoritma koji predviđa koja je vrsta cjepiva primijenjena na svakog miša.[85]

6.1.1.2. Utvrđivanje ciljeva podatkovne analize u kontekstu višekriterijske optimizacije

Titanic skup podataka

Cilj analize *Titanic* skupa podataka je modelirati i optimizirati model strojnog učenja koji će, temeljem identificiranih obrazaca iz trening skupa podataka, dodijeliti binarnu oznaku (0 ili 1) svakom putniku u testnom skupu podataka, pri čemu 0 označava nepreživljavanje, a 1 preživljavanje.[56]

Skup podataka s cijenama nekretnina

Cilj analize *House* skupa podataka je predikcija prodajne cijene svake pojedine nekretnine. Za svaki identifikator (*Id*) u skupu podataka za testiranje, potrebno je procijeniti vrijednost varijable *SalePrice*, koja predstavlja konačnu tržišnu cijenu objekta. Ova predikcija temelji se na analizi skupa značajki koje opisuju karakteristike nekretnina,

gdje se primjenjuju napredne regresijske tehnike kako bi se osigurala što veća točnost modela.[84]

Skup podataka s cjepivima

Cilj analize *Vaccine* skupa podataka je razviti i implementirati odgovarajući algoritam višeklasne klasifikacije koji će precizno predvidjeti skrivena klasifikacijska obilježja na temelju ulaznih atributa u skupu podataka.[85]

6.1.1.3.Odabir evaluacijskih metrika za klasifikaciju i regresiju

Titanic skup podataka

Ključna metrika uspješnosti prediktivnog modela u zadatku klasifikacije *Titanic* skupa podataka jest, prva po rangu, brzina izvedbe i točnost, koja predstavlja omjer ispravno predviđenih ishoda preživljavanja putnika u odnosu na ukupan broj instanci u testnom skupu podataka. [56]

Skup podataka s cijenama nekretnina

Ključna metrika evaluacije modela, u zadatku *House* skupa podataka, je, prva po rangu, brzina izvedbe i RMSE između logaritamske vrijednosti predviđene prodajne cijene i logaritamske vrijednosti stvarne prodajne cijene. Primjena logaritamske transformacije omogućava ujednačeno ponderiranje pogrešaka bez obzira na to odnose li se na nekretnine s visokom ili niskom cijenom. Na taj način osigurava se da odstupanja u predikciji cijena imaju jednak utjecaj na ukupnu procjenu modela, neovisno o apsolutnoj vrijednosti nekretnine.[84]

Skup podataka s cjepivima

Ključna metrika za evaluaciju, u zadatku klasifikacije *Vaccine* skupa podataka jest, prva po rangu, brzina izvedbe i točnost. Točnost modela procjenjuje se usporedbom predviđenih i stvarnih oznaka cjepiva, a definirana je kao omjer ispravno klasificiranih uzoraka i ukupnog broja uzoraka.[85]

6.1.1.4.Analiza ulaznih značajki domenskih skupova podataka

Opis i analiza varijabli obuhvaćaju detaljniji prikaz skupa podataka te uključuju deskriptivne i statističke informacije o varijablama/značajkama. Navedeni proces u prvoj fazi prilagođenog metodološkog okvira prikazuje osnovne metrike, poput aritmetičke sredine, medijana, standardne devijacije, minimalnog/maksimalnog raspona i interkvartilnog raspona, čime se omogućuje dublje razumijevanje distribucije podataka i njihovih svojstava.

Titanic skup podataka

Osnovne informacije o skupu podataka *Titanic* prikazane su tablicom u nastavku.

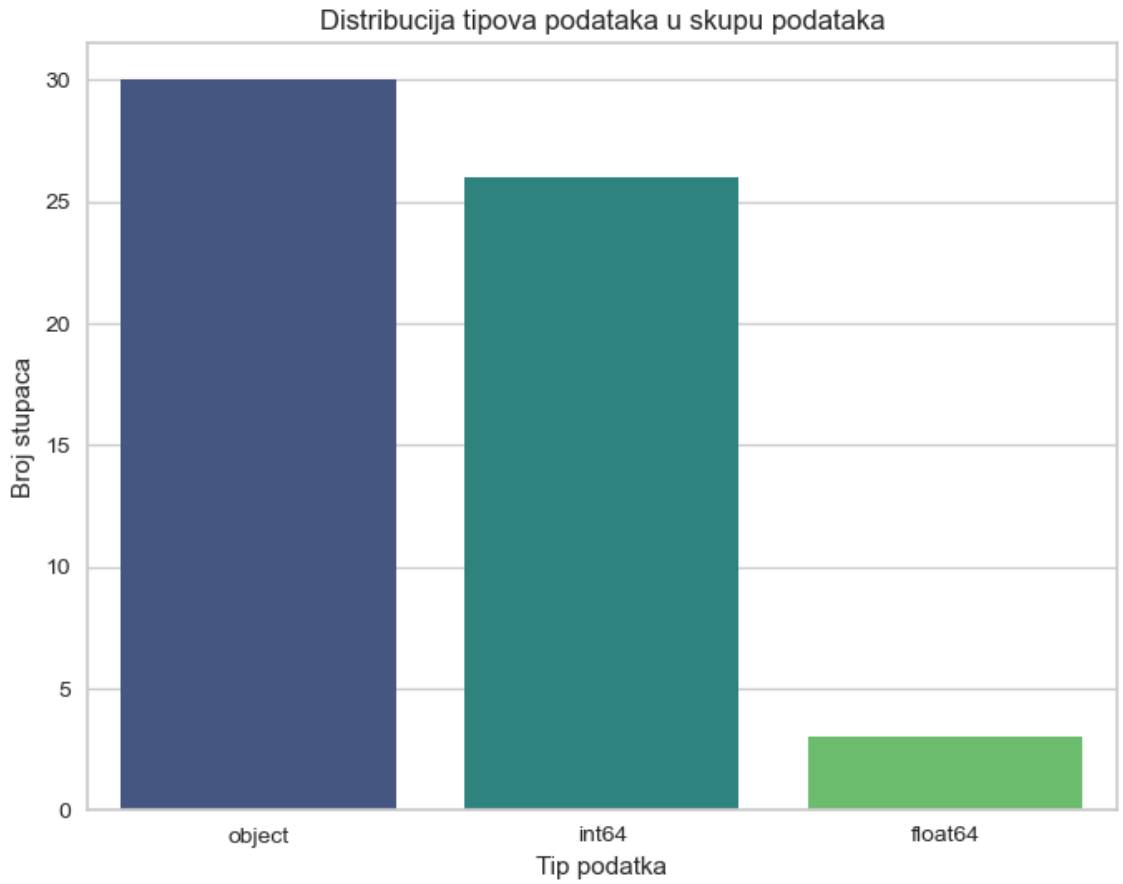
Rbr.	Stupac	Tip podatka
1	PassengerId	int64
2	Survived	int64
3	Pclass	int64
4	Name	object
5	Sex	object
6	Age	float64
7	SibSp	int64
8	Parch	int64
9	Ticket	object
10	Fare	float64
11	Cabin	object
12	Embarked	object

Tablica 16: Skup podataka Titanic

Iz osnovnih informacija *Titanic* skupa podataka, vidljivo je kako se sastoji od 12 atributa, od kojih 11 predstavljaju nezavisne varijable (X), dok je jedna varijabla zavisna, ciljna varijabla (y). Ciljna varijabla *Survived* klasifikacijske je prirode i predstavlja binarnu odluku o preživljavanju, a ulazne značajke sadrže informacije koje omogućuju predikciju tog ishoda. [56]

Skup podataka s cijenama nekretnina

Osnovne informacije o skupu podataka, korištenom u *House* skupu podataka prikazane su slikovnim prikazom u nastavku.



Slika 37:Distribicija tipova podatka u skupu podatka s cijenama nekretnina (House)

Iz osnovnih informacija o skupu podataka, korištenom u *House* skupu podataka vidljivo je kako se sastoji od 81 stupca – *Id* stupac, 79 stupaca su ulazne/nezavisne varijable te je 1 stupac zavisna odnosno ciljna varijabla. Stupčasti graf pokazuje distribuciju tipova podataka, pri čemu se ističe, na grafu, broj stupaca koji su pohranjeni kao kategorijske varijable (engl. *object*). Kategorijske varijable predstavljaju opisne atribute, poput kategorija nekretnina, vrste materijala i sličnih opisa nekretnina. Zatim, numeričke varijable podijeljene su na cjelobrojne (engl. *int64*) i realne (engl. *float64*) vrijednosti. Cjelobrojne varijable predstavljaju diskretne numeričke značajke, kao što su godine izgradnje, broj soba ili broj parkirnih mjesta. Realne vrijednosti u varijablama ukazuju na atribute koji su kontinuiranog tipa, poput površine zemljišta ili cjenovnih vrijednosti.

Skup podataka s cjepivima

Osnovne informacije o skupu podataka *Vaccine* prikazane su tablicom u nastavku.

Rbr.	Stupac	Tip podatka
1	Mouse Number	int64
2	Vaccine	object
3	Maxd1delta Penh	float64
4	STD Penh	float64
5	Weight Delta	float64
6	Construct	object
7	Dose	object

Tablica 17: Skup podataka o cjepivima (Vaccine)

Iz osnovnih informacija *Vaccine* skupa podataka, vidljivo je kako se sastoji od 7 atributa, od kojih 6 predstavljaju nezavisne varijable (*X*), dok je jedna varijabla zavisna, ciljna varijabla (*y*). Ciljna varijabla "*Vaccine*" višeklasifikacijske je prirode i predstavlja odluku o primijenjenoj vrsti cjepiva na svakom mišu, a ulazne značajke sadrže informacije koje omogućuju predikciju tog ishoda.[85] Atributi obuhvaćaju sljedeće podatke, koji omogućuju analizu reakcije miševa na primijenjene tretmane cjepivom[85]:

- Identifikator miša (engl. *Mouse Number*): Jedinstvena oznaka dodijeljena svakom mišu.
- Vrsta cjepiva (engl. *Vaccine*): Kategorijska varijabla koja označava vrstu primijenjenog cjepiva na miša. Miševi bez primijenjenog cjepiva označeni su kategorijom *Naive*. Ostale tri kategorije uključuju *IN* (intrnazalno cjepivo), *SC* (subkutano cjepivo) i *IIAV* (cjepivo na bazi nanoparticula primijenjeno subkutano).
- Maksimalna promjena u respiratornim parametrima (engl. *Max1delta Penh*): Penh predstavlja dimenzionalan broj koji kvantificira promjene u plućnom tkivu miševa. Vrijednost *Max1delta Penh-a* izračunava se kao omjer početnog mjerenja na prvi dan pokusa i maksimalne promjene zabilježene tijekom pokusa.
- Standardna devijacija Penh vrijednosti (engl. *STD Penh*): Mjera varijabilnosti promjena plućnog tkiva miševa kroz 14-dnevni eksperimentalni period.
- Promjena tjelesne mase (engl. *Weight Change*): Razlika u tjelesnoj masi miševa između početka i završetka 14-dnevnog perioda praćenja.

- Virusni konstrukt (engl. *Construct*): Specifični konzervirani proteinski segment virusa gripe primijenjenog na miša.
- Doza cjepiva (engl. *Dose*): Količina primijenjenog cjepiva na pojedinog miša.

6.1.2. Automatizirano inženjerstvo značajki i primjena metoda Glasanja i Ruksaka u MoziaisMl sustavu

Proces obrade i transformacije podataka obuhvaća razvijene metode usmjerene na pripremu skupa podataka u format prikladan za treniranje modela strojnog učenja. Ovaj postupak uključuje čišćenje podataka, rukovanje nestalim vrijednostima, razdvajanje tekstualnih varijabli, kodiranje kategoriziranih varijabli, normalizaciju i skaliranje značajki, gdje se osigurava efikasnost modela pri učenju iz podataka. [11, 21, 22, 54]

6.1.2.1. Struktura i funkcionalnost algoritma latencije u MoziaisMl sustavu

Razvijeni automatizirani cjevovod pretprocesiranja podataka u *MoziaisMl* sustavu obuhvaća niz razvijenih metoda, navedenih u poglavlju Metodologija, potpoglavlje faza Priprema podataka, usmjerenih na pripremu skupa podataka. Sustav *MoziaisMl* uključuje metodu za detekciju i obradu nestalih vrijednosti (*NA, N/A, NaN, NULL* i prazne vrijednosti), metodu za identifikaciju i eliminaciju dupliciranih unosa, metodu za razdvajanje tekstualnih podataka te metodu za filtriranje značajki koje nisu prisutne u treniranom, validacijskom i testnom skupu podataka. Nadalje, u automatiziranom cjevovodu pretprocesiranja *MoziaisMl* sustava implementirana je metoda za kategorizaciju značajki u brojčani format te metoda za normalizaciju i skaliranje značajki, čime se osigurava konzistentnost podataka i poboljšava učinkovitost modela strojnog učenja u fazi Modeliranje i optimizacija modela.

Titanic skup podataka

Rezultat rada razvijenog automatiziranog cjevovoda *MoziaisMl* sustava prikazan je u nastavku. Razvijena metoda pretprocesiranja podataka za utvrđivanje *NA, N/A, NaN, NULL* i praznih vrijednosti unutar stupaca skupa podataka dala je sljedeći ukupan broj *NA, N/A, NaN, NULL* i praznih vrijednosti u treniranim podatcima:

$$N_{ukupno} = \sum_{c \in C} \sum_{i=i}^{m} \left(X_{ic} = NA,\ N/A,\ NaN,\ NULL, "" \right)$$
$$N_{ukupno} = 866 \tag{59}$$

U prethodnoj formuli C je skup svih stupaca u skupu podataka, m broj redaka u skupu podataka, X_{ic} vrijednost u i-tom retku i c-tom stupcu, a (X_{ic} = *NA, N/A, NaN, NULL*, prazna vrijednost) je *boolean* funkcija koja vraća 1 ako je vrijednost *NA, N/A, NaN, NULL*, prazna vrijednost, u protivnom 0. Ukupan broj nedostajućih vrijednosti kroz sve stupce u treniranim podatcima je 866, a broj *NA, N/A, NaN, NULL*, praznih vrijednosti po stupcima u treniranim podatcima je:

$$\begin{aligned}
N_c &= \sum_{i=1}^{m} \left(X_{ic} = NA,\ N/A,\ NaN,\ NULL, "" \right) \\
N_{Age} &= \sum_{i=1}^{m} \left(X_{i,Age} = NA,\ N/A,\ NaN,\ NULL, "" \right) = 177 \\
N_{Cabin} &= \sum_{i=1}^{m} \left(X_{i,Cabin} = NA,\ N/A,\ NaN,\ NULL, "" \right) = 687 \\
N_{Embarked} &= \sum_{i=1}^{m} \left(X_{i,Emabrked} = NA,\ N/A,\ NaN,\ NULL, "" \right) = 2
\end{aligned} \tag{60}$$

U prethodnoj formuli N_c ukupan broj *NA, N/A, NaN, NULL*, praznih vrijednosti u stupcu c odnosno stupcima *Age*, *Cabin* i *Embarked*. Omjer *NA, N/A, NaN, NULL* i praznih vrijednosti u odnosu na ukupan broj redaka (varijabla m=*891*) u treniranom skupu podatka prikazan je u nastavku:

$$\begin{aligned}
P_c &= \frac{N_c}{m} \\
P_{Age} &= \left(\frac{177}{891} \right) \times 100 = 19,86\% \\
P_{Cabin} &= \left(\frac{687}{891} \right) \times 100 = 77,12\% \\
P_{Embarked} &= \left(\frac{2}{891} \right) \times 100 = 0,22\%
\end{aligned} \tag{61}$$

Ukupan broj nedostajućih vrijednosti kroz sve stupce u testnim podatcima je 414, a dobiven je:

$$N_{ukupno} = \sum_{c \in C} \sum_{i=i}^{m} \left(X_{ic} = NA,\ N/A,\ NaN,\ NULL, "" \right)$$
$$N_{ukupno} = 414 \tag{62}$$

Ukupan broj *NA, N/A, NaN, NULL*, praznih vrijednosti po stupcima u testnim podatcima je:

$$\begin{aligned} N_c &= \sum_{i=1}^{m} \left(X_{ic} = NA,\ N/A,\ NaN,\ NULL, "" \right) \\ N_{Age} &= \sum_{i=1}^{m} \left(X_{i,Age} = NA,\ N/A,\ NaN,\ NULL, "" \right) = 86 \\ N_{Fare} &= \sum_{i=1}^{m} \left(X_{i,Fare} = NA,\ N/A,\ NaN,\ NULL, "" \right) = 1 \\ N_{Cabin} &= \sum_{i=1}^{m} \left(X_{i,Cabin} = NA,\ N/A,\ NaN,\ NULL, "" \right) = 327 \end{aligned} \tag{63}$$

Omjer *NA, N/A, NaN, NULL* i praznih vrijednosti u odnosu na ukupan broj redaka (varijabla *m=418*) u treniranom skupu podatka prikazan je u nastavku:

$$\begin{aligned} P_c &= \frac{N_c}{m} \\ P_{Age} &= \left(\frac{86}{418} \right) \times 100 = 20,57\% \\ P_{Fare} &= \left(\frac{1}{418} \right) \times 100 = 0,24\% \\ P_{Cabin} &= \left(\frac{327}{418} \right) \times 100 = 78,23\% \end{aligned} \tag{64}$$

Razvijena metoda pretprocesiranja podataka za utvrđivanje *NA, N/A, NaN, NULL* i praznih vrijednosti i kvantifikaciju prisutnosti *NA, N/A, NaN, NULL* i praznih vrijednosti u skupu podataka prosljeđuje sljedećoj razvijenoj metodi pretprocesiranja podataka za obradu *NA, N/A, NaN, NULL* i praznih vrijednosti.

Razvijena metoda pretprocesiranja podataka za obradu *NA, N/A, NaN, NULL* i praznih vrijednosti kreira za svaki stupac *c* iz skupa značajki *C*, kreira se novi binarni stupac $c\_was\_missing$ koji označava prisutnost *NA, N/A, NaN, NULL* i praznih vrijednosti u originalnom stupcu. Razvijena metoda pretprocesiranja podataka za utvrđivanje detektira

NA, N/A, NaN, NULL i praznih vrijednosti, razvijena metoda pretprocesiranja podataka za obradu zamjenjuje *NA, N/A, NaN, NULL* i prazne vrijednosti se unaprijed definiranim imputacijskim vrijednostima *v* (na ovom skupu podatka *v=0*):

$$
\begin{aligned}
X_{ic} &= \begin{cases} v, & ako\ je\ X_{i,c} = NA,\ N/A,\ NaN,\ NULL,"" \\ X_{ic}, & u\ protivnom \end{cases} \\
X_{i,Age} &= \begin{cases} 0, & ako\ je\ X_{i,Age} = NA,\ N/A,\ NaN,\ NULL,"" \\ X_{i,Age}, & u\ protivnom \end{cases} \\
X_{i,Cabin} &= \begin{cases} 0, & ako\ je\ X_{i,Cabin} = NA,\ N/A,\ NaN,\ NULL,"" \\ X_{i,Cabin}, & u\ protivnom \end{cases} \\
X_{i,Embarked} &= \begin{cases} 0, & ako\ je\ X_{i,Embarked} = NA,\ N/A,\ NaN,\ NULL,"" \\ X_{i,Embarked}, & u\ protivnom \end{cases}
\end{aligned}
\tag{65}
$$

Metoda pretprocesiranja podataka za detekciju dupliciranih vrijednosti razvijena je tako da u početnom koraku identificira i izdvoji stupce koji služe kao jedinstveni identifikatori podataka, u *Titanic* skupu podataka: *PassengerId* i *Name*. Nakon toga, provodi analizu preostalih značajki isključujući identifikatorske stupce te računa ukupan broj dupliciranih redaka *D*. Proces identifikacije dupliciranih zapisa temelji se na usporedbi svih redaka unutar filtriranog skupa podataka, čime se osigurava precizno prepoznavanje redundantnih unosa, a time i poboljšava kvaliteta podataka za daljnju analizu i modeliranje:

$$
\begin{aligned}
D &= \sum_{i=1}^{m} \left(\exists j \neq i : X_{iC_{dupl}} = X_{jC_{dupl}} \right) \\
D &= 15
\end{aligned}
\tag{66}
$$

U prethodnoj formuli $X_{iC_{dupl}}$ je vektor vrijednosti za sve značajke u skupu C_{dupl} za redak *i*, a $\left(\exists j \neq i : X_{iC_{dupl}} = X_{jC_{dupl}} \right)$ je boolean funkcija koja vraća 1 ako postoji barem jedan duplikat.

Skup podataka s cijenama nekretnina

Rezultat rada razvijenog automatiziranog cjevovoda *MoziaisMl* sustava, na *House* skupu podataka, prikazan je u nastavku. Razvijena metoda pretprocesiranja podataka za utvrđivanje *NA, N/A, NaN, NULL* i praznih vrijednosti unutar stupaca skupa podataka

dala je sljedeći ukupan broj *NA, N/A, NaN, NULL* i praznih vrijednosti u treniranim podatcima:

$$N_{ukupno} = \sum_{c \in C} \sum_{i=i}^{m} \left(X_{ic} = NA,\ N/A,\ NaN,\ NULL, "" \right)$$
$$N_{ukupno} = 6965 \tag{67}$$

Ukupan broj nedostajućih vrijednosti kroz sve stupce u treniranim podatcima je 6.965, dok u testnim podatcima je bilo 7.000 nedostajućih vrijednosti:

$$N_{ukupno} = \sum_{c \in C} \sum_{i=i}^{m} \left(X_{ic} = NA,\ N/A,\ NaN,\ NULL, "" \right)$$
$$N_{ukupno} = 7000 \tag{68}$$

Ukupan broj dupliciranih redaka *D*, unutar skupa podataka, isključujući identifikatorske stupce, je bio 0. Izračunat je na sljedeći način:

$$D = \sum_{i=1}^{m} \left(\exists j \neq i : X_{iC_{dupl}} = X_{jC_{dupl}} \right)$$
$$D = 0 \tag{69}$$

Skup podataka s cjepivima

Rezultat rada razvijenog automatiziranog cjevovoda *MoziaisMl* sustava, na *Vaccine* skupu podataka kroz metode za utvrđivanje *NA, N/A, NaN, NULL* i praznih vrijednosti unutar stupaca i broj dupliciranih redaka bio je 0.

Unutar faze Pripreme podataka, implementirane su i dodatne razvijene metode pretprocesiranja koje omogućuju strukturiranje i transformaciju podataka u format pogodan za treniranje modela strojnog učenja. Među razvijenim metodama je metoda razdvajanja tekstualnih vrijednosti, koja koristi razmak (' ') kao razdjelnicu za dekompoziciju višedijelnih atributa u zasebne značajke, čime se poboljšava kvaliteta reprezentacije podataka. Zatim, implementirana je razvijena metoda filtriranja značajki koja uklanja atribute koji nisu prisutni u skupu podataka korištenom za treniranje i testiranje, čime se osigurava konzistentnost podataka u cijelom eksperimentalnom okruženju. Nadalje, razvijena je metoda kategorizacije numeričkih varijabli, koja transformira atributivne značajke u numerički format koristeći unaprijed definiranu granicu od 12 kategorija, čime se olakšava njihova integracija u modele strojnog učenja. Na kraju, razvijena metoda normalizacije i skaliranja značajki omogućuje standardizaciju

distribucije podataka, gdje se osigurava bolja usporedivost varijabli tijekom procesa treniranja.

6.1.2.2.Automatizirana eksplorativna analiza ulaznog skupa podataka

Eksplorativna analiza podataka (engl. *Exploratory Data Analysis*, EDA) je proces, isto, u drugoj fazi Priprema podataka prilagođenog metodološkog okvira, tijekom kojeg su detaljno ispitana svojstva i struktura skupa podataka. Procesom eksplorativne analize podataka otkrivaju se nastale promjene uslijed korištenja automatiziranog cjevovoda pretprocesiranja podataka u *MoziaisMl* sustavu, distribucija varijabli, njihov međusobni odnos te potencijalni utjecaj na ciljnu varijablu (*y*).[88]

Titanic skup podataka

Nakon što je razvijeni automatizirani cjevovod pretprocesiranja podataka u *MoziaisMl* sustavu pretprocesirao skup podataka sa svim razvijenim metodama, osnovne informacije pretprocesiranog skupa podataka *Titanic* prikazane su tablicom u nastavku.

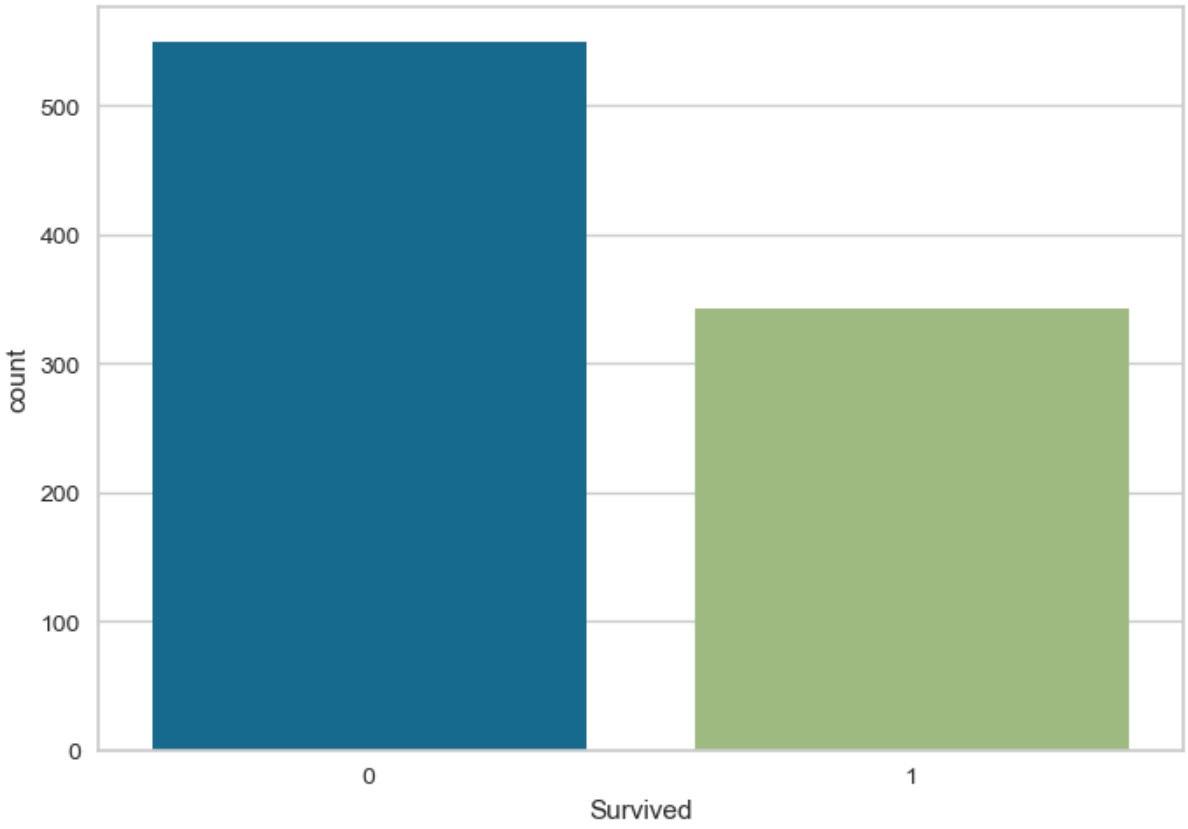
Rbr	Stupac	Tip podatka
1	Survived	int64
2	Pclass	float64
3	Sex	float64
4	Age	float64
5	SibSp	float64
6	Parch	float64
7	Ticket	float64
8	Fare	float64
9	Cabin	float64
10	Age_was_missing	float64
11	Cabin_was_missing	float64
12	Name0	float64
13	Name1	float64
14	Name2	float64
15	Name3	float64
16	Name4	float64
17	Name5	float64
18	Name6	float64
19	Name7	float64
20	Ticket0	float64
21	Ticket1	float64
22	Ticket2	float64
23	Cabin0	float64
24	Cabin1	float64
25	Cabin3	float64
26	Embarked_C	float64
27	Embarked_Q	float64
28	Embarked_S	float64
29	Name8_0	float64
30	Cabin2_0	float64
31	Cabin2_B55	float64
32	Cabin2_B63	float64
33	Cabin2_C27	float64

Tablica 18: Pretprocesirani skup podataka Titanic

Iz osnovnih informacija pretprocesiranog *Titanic* skupa podataka, vidljivo je kako se skup podataka sada sastoji od 33 atributa/stupca, od kojih 32 predstavljaju nezavisne varijable (X), dok je jedna varijabla zavisna, ciljna varijabla (y). Ciljna varijabla "*Survived*" klasifikacijske je prirode i predstavlja binarnu odluku o preživljavanju (0-

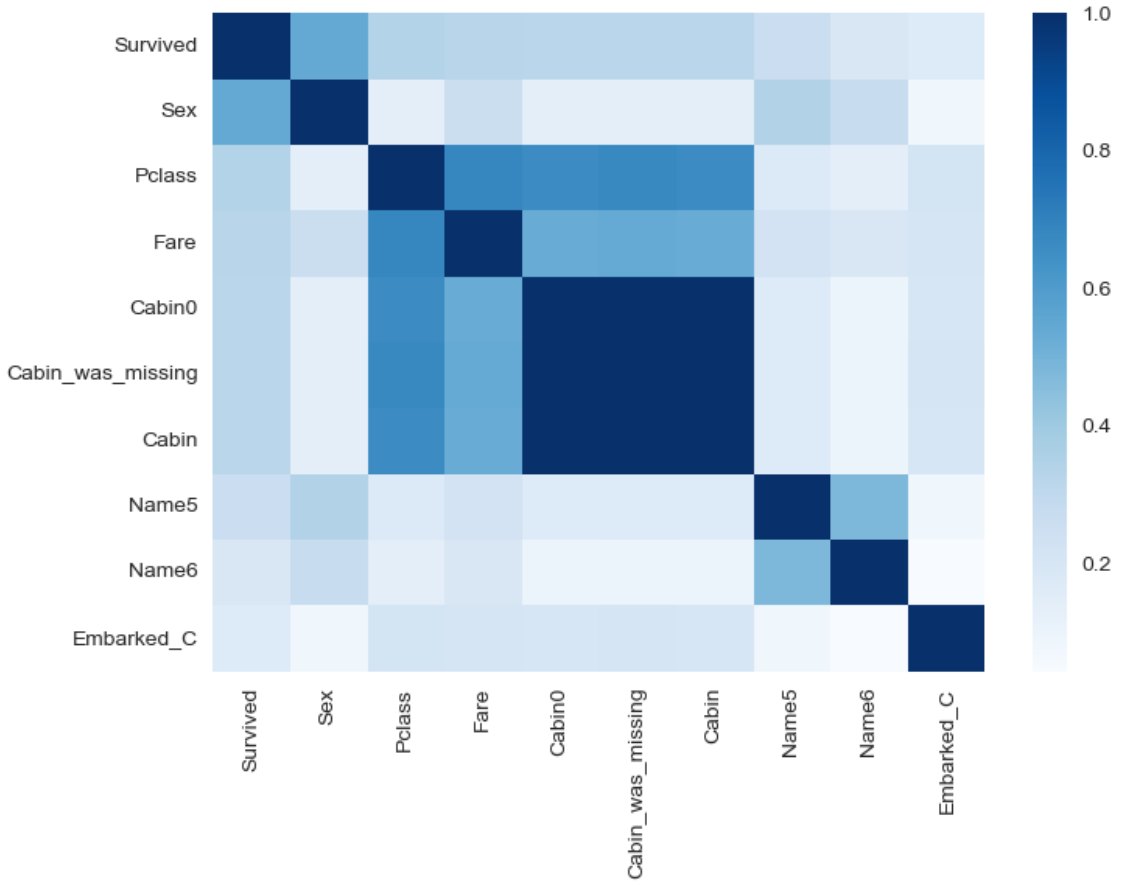
nepreživljavanje, 1-preživljavanje), a ulazne značajke sadrže informacije koje omogućuju predikciju tog ishoda. [56]

Analiza distribucije klasa unutar treniranog skupa podataka otkriva značajnu neravnotežu između dviju kategorija ishoda. Konkretno, klasa koja označava nepreživljavanje (0) obuhvaća 61,62% ukupnih instanci, dok klasa koja označava preživljavanje (1) čini 38,39% podataka. Ovakva distribucija može imati značajan utjecaj na treniranje modela strojnog učenja, gdje modeli mogu biti skloniji favoriziranju dominantne klase. Stoga, ovakva distribucija klasa je stavila na test treću fazu okvira, Modeliranje i optimizacija modela, gdje automatizirano sustavno pretraživanje i evaluacija različitih modela te odabir njihovih optimalnih vrijednosti hiperparametara, na kraju, je i primijenilo tehnike za balansiranje skupa podataka odnosno ponderiranje klasa odabirom metode *RandomForestClassifier[25]* s parametrom *class_weight='balanced'*, kako bi se poboljšala sposobnost modela da točno klasificira i manje zastupljenu klasu (0 preživljavanje).



Slika 38: Balans klasa

Analiza korelacije ciljne varijable *Survived* s atributima iz ulaznog skupa podataka *X* ukazuje na značaj vlastito razvijenih metoda za pretprocesiranje podataka. Konkretno, nakon primjene pretprocesiranja, novokreirane značajke *Cabin0* (0,32), *Cabin_was_missing* (0,32), *Name5* (0,26), *Name6* (0,19) i *Embarked_C* (0,17) ostvaruju statistički značajnu korelaciju s ciljnom varijablom. Nadalje, najizraženije korelacije s varijablom *Survived* pokazuju značajke *Sex* (0,54) i *Pclass* (0,40), dok izvorna značajka *Cabin*, u usporedbi s novokreiranim atributima *Cabin0* i *Cabin_was_missing*, ostvaruje nižu korelacijsku vrijednost, čime se potvrđuje opravdanost generiranja novih varijabli u procesu pretprocesiranja podataka te samim tim i opravdanost razvoja vlastitih metoda za pretprocesiranje.



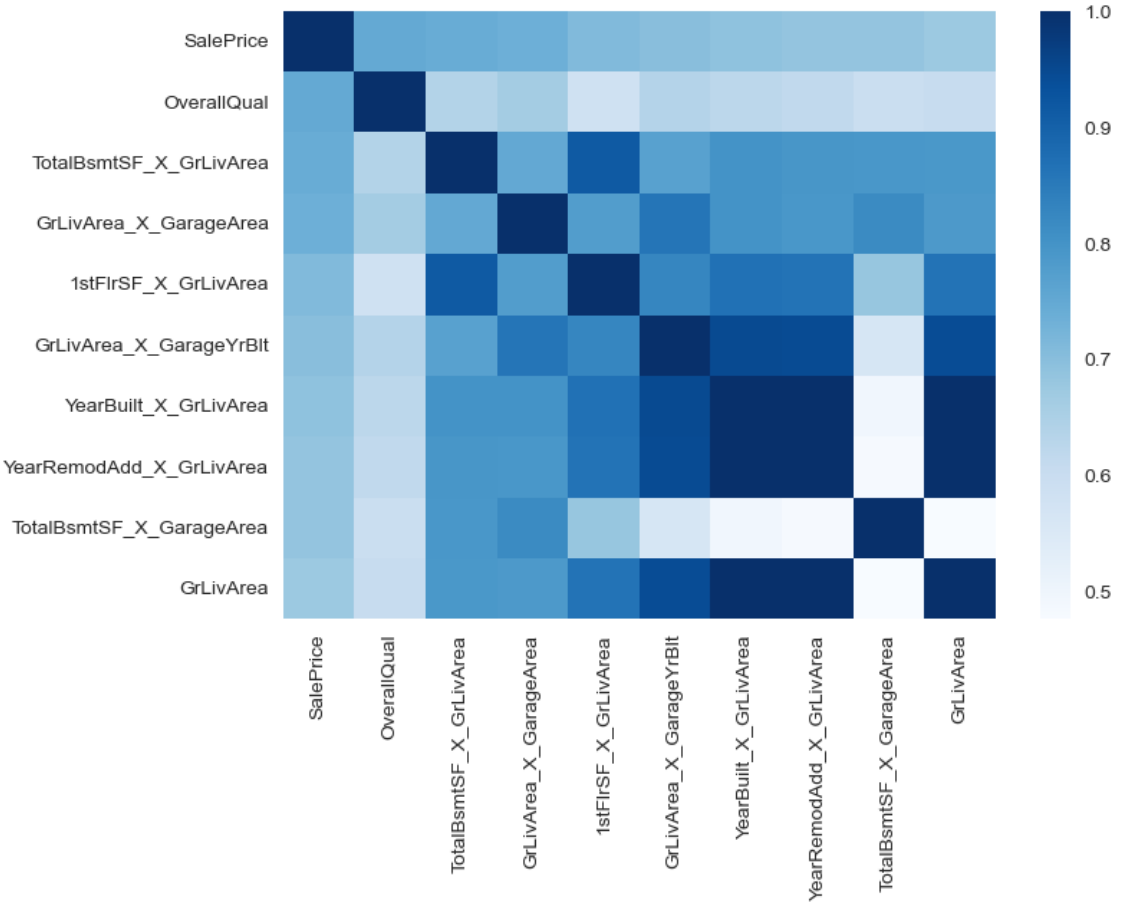
Slika 39: Korelacije atributa sa ciljnom varijablom Survived

Matrica se temelji na Pearsonovom koeficijentu korelacije, čija vrijednost je između -1 i 1. Pozitivne vrijednosti ukazuju na pozitivnu korelaciju, dok negativne vrijednosti sugeriraju negativnu korelaciju odnosno povezanost među atributima. Nijanse plave boje označavaju intenzitet korelacije između različitih atributa, gdje tamnije nijanse

ukazuju na snažniju korelaciju (bliže 1), dok svjetlije nijanse sugeriraju slabiju povezanost.

Skup podataka s cijenama nekretnina

Nakon što je razvijeni automatizirani cjevovod pretprocesiranja podataka u *MoziaisMl* sustavu pretprocesirao skup podataka sa svim razvijenim metodama, novi skup broji 341 značajku za predikciju ciljne varijable (*y*).



Slika 40: Korelacije atributa s ciljnom varijablom SalePrice

Analiza korelacije ciljne varijable prodajne cijene (*SalePrice)* s atributima iz ulaznog skupa podataka *X* ukazuje na intenzitet korelacije sa sljedećim atributima:

- *OverallQual* (0,7488) – Kvaliteta gradnje nekretnine odnosno njezinih materijala ima značajnu pozitivnu povezanost s prodajnom cijenom.

- *TotalBsmtSF_X_GrLivArea* (0,7423) – Umnožak ukupne površine podruma i stambene površine prizemlja pokazuje visoku korelaciju s prodajnom cijenom, što sugerira da veće nekretnine s prostranim podrumima postižu višu prodajnu cijenu.
- *GrLivArea_X_GarageArea* (0,7341) – Povezanost između stambene površine (*GrLivArea*) i garažnog prostora (*GarageArea*) pokazuje snažnu povezanost s prodajnom cijenom.
- *1stFlrSF_X_GrLivArea* (0,7098) – Povezanost površine prizemlja i ukupne stambene površine, isto, pokazuje snažnu povezanost s prodajnom cijenom.
- *GrLivArea_X_GarageYrBlt* (0,7002) – Stambena površina zajedno s godinom izgradnje garaže ukazuje na određen utjecaj na prodajnu cijenu.
- *YearBuilt_X_GrLivArea* (0,6922) i *YearRemodAdd_X_GrLivArea* (0.6851) – navedeni atributi godina izgradnje i godina renovacije sugeriraju da takve nekretnine u kombinaciji s većim stambenom površinom imaju višu prodajnu cijenu.
- *TotalBsmtSF_X_GarageArea* (0,6849) – Povezanost ukupne površine podruma i garaže potvrđuje, isto, utjecaj na prodajnu cijenu.
- *GrLivArea* (0,6760) – Ukupna površina stambenog prostora, isto, pokazuje povezanost s prodajnom cijenom.

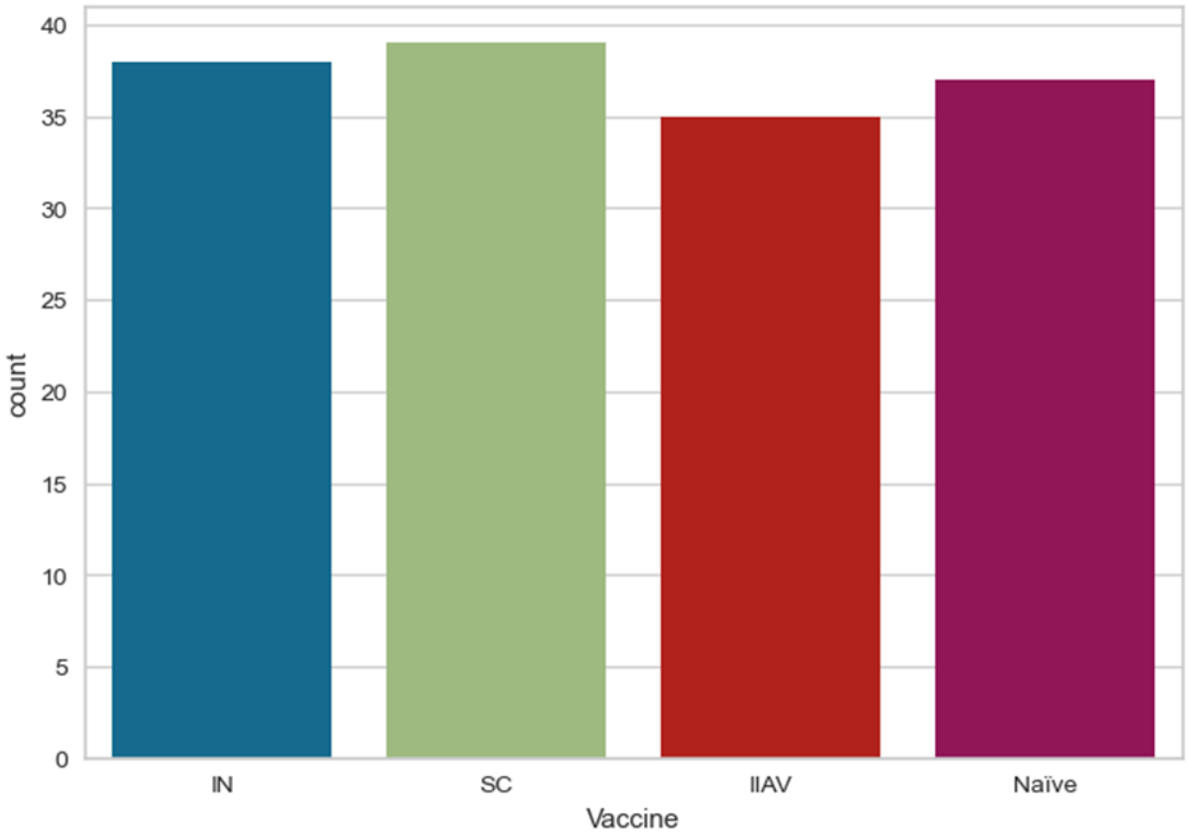
Skup podataka s cjepivima

Nakon što je razvijeni automatizirani cjevovod pretprocesiranja podataka u *MoziaisMl* sustavu pretprocesirao skup podataka sa svim razvijenim metodama, osnovne informacije pretprocesiranog skupa podataka *Vaccine* prikazane su tablicom u nastavku.

Rbr.	Atribut	Tip podatka
1	Vaccine	int32
2	Maxd1delta Penh	float64
3	STD Penh	float64
4	Weight Delta	float64
5	Construct	float64
6	Dose	float64
7	Dose0	float64
8	Dose1	float64

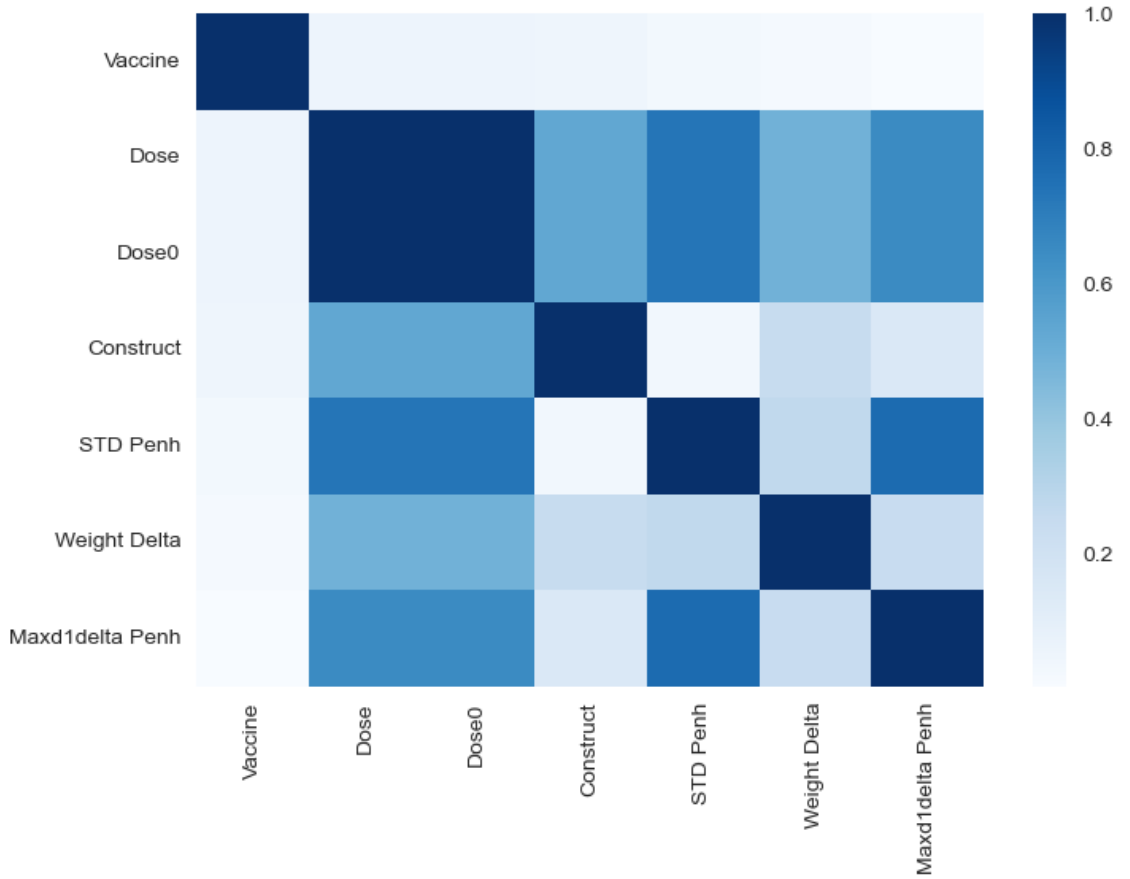
Tablica 19: Pretprocesirani skupa podataka s cjepivima

Iz osnovnih informacija pretprocesiranog *Vaccine* skupa podataka, vidljivo je kako se skup podataka sada sastoji od 8 atributa/stupca, od kojih 7 predstavljaju nezavisne varijable (*X*), dok je jedna varijabla zavisna, ciljna varijabla (*y*). Ciljna varijabla *Vaccine* višeklasifikacijske je prirode i predstavlja 4-klasnu odluku o primijenjenom cjepivu (IIAV, IN, Naive(nije primljeno cjepivo), SC), a ulazne značajke sadrže informacije koje omogućuju predikciju primijenjenog cjepiva.[85]



Slika 41: Balans klasa

Analiza distribucije klasa unutar skupa podataka pokazuje razmjerno ravnomjernu i ujednačenu raspodjelu između različitih vrsta cjepiva (IN, SC, IIAV) i kontrolne skupine (Naive). Subkutano cjepivo (SC) čini najveći udio uzoraka s 26,17%, dok je Nanopartikulno subkutano cjepivo (IIAV) prisutno u 23,49% slučajeva. Intrnazalno cjepivo (IN) obuhvaća 25,50%, dok kontrolna skupina (Naive), predstavlja miševe koji nisu primili cjepivo, čini 24,83% uzoraka.



Slika 42: Korelacije atributa s ciljnom varijablom Vaccine

Analiza korelacije s varijablom *Vaccine* ukazuje kako varijable doze primijenjenog cjepiva (Dose i Dose0) pokazuju blagu pozitivnu korelaciju s cjepivom (*Vaccine*), s vrijednošću 0,0543, što navodi da postoji određena, ali slaba veza između doze i vrste primijenjenog cjepiva. Treća varijabla konzervirani proteinski segment virusa (*Construct*) pokazuje korelaciju od 0,0520, što upućuje na vrlo blagu povezanost između specifične varijante virusa korištene u istraživanju i vrste cjepiva primijenjenog na miševima. Varijabla standardna devijacija promjena u plućnom tkivu miševa tijekom 14-dnevnog pokusa (*STD Penh*) ima još slabiju korelaciju s cjepivom (0,0283), što ukazuje na nisku povezanost između plućnih promjena i vrste primijenjenog cjepiva. Četvrta varijabla koja mjeri promjene tjelesne mase miševa tijekom pokusa (*Weight Delta*), isto pokazuje vrlo nisku korelaciju s vrstom cjepiva (0,0205), što znači da vrsta primijenjenog cjepiva nema značajan utjecaj na promjenu mase ispitanih miševa. Peta varijabla bilježi najnižu korelaciju - 0,0028, što implicira gotovo nepostojeću povezanost između maksimalne promjene u plućnom odgovoru i vrste primijenjenog cjepiva.

6.1.3. Višekriterijska optimizacija modela s fokusom na predikciju i vrijeme treniranja

Autonomno i automatiziranim putem *MoziaisMl* sustava bira i optimizira modele prema zadanim postavkama metoda i hiperparametara, koji su navedeni u poglavlju Metodologija, potpoglavlje faza Modeliranje i optimizacija modela. [11, 21, 22, 54]

6.1.3.1. Automatizirani odabir modela i hiperparametara u MoziaisMl sustavu

Titanic skup podataka

Cilj predikcije je dodijeliti binarnu oznaku (0 ili 1) svakom putniku, pri čemu oznaka 0 označava nepreživljavanje, a oznaka 1 preživljavanje[56], stoga se predikcija definira kao zadatak binarne klasifikacije. U svrhu modeliranja i optimizacije modela na *Titanic*[56] skupu podataka, primjenjivale su se klasifikacijske metode strojnog učenja, uz optimizaciju hiperparametara klasifikacijskih modela, kako bi se osigurala maksimalna brzina i točnost modela.

6.1.3.2. Evaluacija izvedbenih i prediktivnih hiperparametara modela

Titanic skup podataka

Rezultati i analiza su prikazani tablicom u nastavku.

Rbr.	Model	Najbolji rezultat	Vrijeme izvršavanja (s)	Najbolji parametri	Broj kombinacija
1	RandomForestClassifier	0,8384	30,65	'class_weight': 'balanced', 'criterion': 'entropy', 'max_depth': None, 'max_features': 'log2', 'n_estimators': 100	144
2	GradientBoostingClassifier	0,8384	62,04	'criterion': 'friedman_mse', 'learning_rate': 0.2, 'loss': 'exponential', 'max_depth': 30, 'max_features': 'sqrt', 'n_estimators': 50	864
3	XGBClassifier	0,8283	3,51	'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': None	48

4	MLPClassifier	0,8148	63,73	'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (100,), 'learning_rate': 'adaptive', 'max_iter': 500, 'solver': 'adam'	48
5	SVC	0,8114	0,7	'C': 1, 'gamma': 'scale', 'kernel': 'rbf'	30
6	KNeighborsClassifier	0,8025	7,85	'algorithm': 'auto', 'metric': 'manhattan', 'n_neighbors': 11, 'p': 1, 'weights': 'distance'	240
7	DecisionTreeClassifier	0,8002	1,12	'class_weight': None, 'criterion': 'gini', 'max_depth': 10, 'max_features': None	48
8	GaussianNB	0,6622	0,02	'priors': None, 'var_smoothing': 1e-08	3

Tablica 20:Modelirani i optimizirani modeli s hiperparametrima

Faza Modeliranja i optimizacije modela provedena putem *MoziaisMl* sustava autonomno je odabrala algoritam *RandomForestClassifier[25]* uz parametrizaciju *class_weight='balanced'*, što je opravdano obzirom na prisutnu neuravnoteženost klasa unutar skupa podataka. Evaluacija modela pokazala je točnost klasifikacije od 84%, gdje je vrijeme izvršavanja iznosilo 30,65 sekundi. Kao alternativni model razmatran je *GradientBoostingClassifier*[70], koji je ostvario jednaku točnost, ali uz značajno veće vrijeme izvršavanja od 62,04 sekunde, što predstavlja više od 100% povećanja u odnosu na prvi model. S obzirom na kriterije, ovog istraživanja, optimizacije vremena izvedbe odnosno brzine i točnosti, za fazu Evaluacije odabran je model *RandomForestClassifier[25]* s navedenim hiperparametrima.

Skup podataka s cijenama nekretnina

Rezultati i analiza su prikazani tablicom u nastavku.

Rbr.	Model	Najbolji rezultat	Vrijeme izvršavanja (s)	Najbolji parametri	Broj kombinacija
1	DecisionTreeRegressor	0,7783	7,4	'ccp_alpha': 0.3, 'max_depth': 10, 'max_features': None	64
2	KNeighborsRegressor	0,7493	1,91	'metric': 'manhattan', 'n_neighbors': 9, 'weights': 'distance'	30
3	ElasticNet	0,6043	35,87	'alpha': 0.01, 'l1_ratio': 0.2	110
4	Lasso	0,5885	2,59	'alpha': 1e-05	10
5	Ridge	0,5761	0,6	'alpha': 10	10
6	LinearRegression	0,5511	1,16	'copy_X': True, 'fit_intercept': True, 'n_jobs': None, 'positive': False	1
7	MLPRegressor	-2,1817	155,53	'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (100, 100), 'learning_rate': 'constant', 'max_iter': 500, 'solver': 'adam'	48
8	SVR	-	10815,74	-	-
9	XGBRegressor	-	10815,74	-	-
10	RandomForestRegressor	-	10815,74	-	-
11	GradientBoostingRegressor	-	10815,74	-	-

Tablica 21: Modelirani i optimizirani modeli s hiperparametrima

Faza Modeliranja i optimizacije modela provedena putem *MoziaisMl* sustava autonomno je odabrala najuspješniju metodu prema evaluacijskim metrikama -

DecisionTreeRegressor[80], model koji je ostvario najbolji rezultat od 0,7783, uz relativno kratko vrijeme izvršavanja od 7,4 sekunde. Metoda *DecisionTreeRegressor*[80] koristi ccp_alpha=0.3 i maksimalnu dubinu stabla postavljenu na 10, što ukazuje na regulaciju složenosti modela radi postizanja bolje generalizacije. Druga metoda, u rangiranju po uspješnosti, je *KNeighborsRegressor*[78], s rezultatom 0,7493 i izuzetno brzim vremenom izvršavanja od 1,91 sekundu. Metoda *KNeighborsRegressor*[78] koristi *Manhattan* metriku udaljenosti i 9 susjeda u predikciji, uz ponderirane težine (*distance*). Metoda *ElasticNet*[76] je ostvarila rezultat 0,6043, ali s puno duljim vremenom izvršavanja od 35,87 sekundi. Metode *Lasso*[74] (0,5885) i *Ridge*[75] (0,5761) su isto ostvarili pristojne rezultate, s kraćim vremenima izvršavanja od 2,59 i 0,6 sekundi. Metoda *LinearRegression*[73] ostvarila je rezultat od 0,5511, uz najkraće vrijeme izvršavanja (1,16 sekundi). Jedina metoda koja je postigla negativan rezultat je *MLPRegressor*[79] (-2,1817), što ukazuje na preveliku složenost modela u odnosu na skup podataka, pretjerano prilagođavanje treniranju ili problem konvergencije neuronske mreže. Vrijeme izvršavanja metode *MLPRegressor*[79] bilo je 155,53 sekunde, što dodatno potvrđuje neefikasnost kod ovog specifičnog skupa podataka. Četiri metode - *SVR*[77], *XGBRegressor*[27], RandomForestRegressor[26] i GradientBoostingRegressor[81] - nisu dovršile evaluaciju, a vrijeme izvršavanja navedenih metoda bilo je preko 3 sata (10.815,74 sekundi). Prethodno navedene metode zahtijevale su značajno više računalnih resursa te je zbog toga prekinuto njihovo izvršavanje.

Faza Modeliranja i optimizacije modela provedena putem *MoziaisMl* sustava upućuje da su metode *DecisionTreeRegressor*[80] i *KNeighborsRegressor*[78] najbolji izbori u parametrima ovog istraživanja - omjera točnosti i brzine izvršavanja. Metode *ElasticNet*[76], *Lasso*[74] i *Ridge*[75] pružaju alternativna rješenja koja mogu biti korisna u slučajevima kada je regularizacija modela ključna. Metode koje imaju neadekvatne performanse i nisu završile evaluaciju zahtijevaju dodatne optimizacije i optimizaciju računalnih resursa kako bi bile modelirane i optimizirane za upotrebu u ovom skupu podataka.

Skup podataka s cjepivima

Rezultati i analiza su prikazani tablicom u nastavku.

Rbr	Model	Najbolji rezultat	Vrijeme izvršavanja (s)	Najbolji parametri	Broj kombinacija
1	MLPClassifier	0,4027	18,46	'activation': 'logistic', 'alpha': 0.0001, 'hidden_layer_sizes': (50,), 'learning_rate': 'adaptive', 'max_iter': 500, 'solver': 'adam'	48
2	RandomForestClassifier	0,3959	11,44	'class_weight': 'balanced', 'criterion': 'gini', 'max_depth': 30, 'max_features': None, 'n_estimators': 50	144
3	DecisionTreeClassifier	0,3954	1,03	'class_weight': None, 'criterion': 'gini', 'max_depth': 10, 'max_features': None	48
4	SVC	0,3897	0,1	'C': 10, 'gamma': 'auto', 'kernel': 'rbf'	30
5	GaussianNB	0,3230	0,01	'priors': None, 'var_smoothing': 1e-08	3
6	KNeighborsClassifier	0,3216	3,61	'algorithm': 'auto', 'metric': 'euclidean', 'n_neighbors': 5, 'p': 1, 'weights': 'distance'	240
7	XGBClassifier	-	2,69	'class_weight': None, 'criterion': 'gini', 'max_depth': None, 'max_features': None	48
8	GradientBoostingClassifier	-	10225,34	-	-

Tablica 22: Modelirani i optimizirani modeli s hiperparametrima

Faza Modeliranja i optimizacije modela provedena putem *MoziaisML* sustava autonomno je odabrala kao najbolji model prema postignutim rezultatima - *MLPClassifier[72]*, koji je ostvario najbolju metriku točnosti od 0,4027 uz vrijeme izvršavanja od 18,46 sekundi. Zatim slijedi *RandomForestClassifier*[25], koji je postigao

točnost od 0,3959, ali uz značajno kraće vrijeme izvršavanja od 11,44 sekunde, što ukazuje na znatno brže izvođenje modeliranja i optimizacije parametara u usporedbi s neuronskim mrežama. Metoda *DecisionTreeClassifier*[64] isto pokazuje konkurentne performanse s točnošću 0,3953, ali s kraćim vremenom izvršavanja u odnosu na neuronske mreže. Metoda *Support Vector Classifier* (SVC)[69] ostvarila je rezultat od 0,3896, što je nešto niže u usporedbi s prethodno navedenim metodama, dok metode *GaussianNB*[67] i *KNeighborsClassifier*[71] su postigle nižu točnost od 0,3229 i 0,3216, što ukazuje na to da se *GaussianNB*[67] i *KNeighborsClassifier*[71] nisu najbolje prilagodile strukturi podataka. Metoda *XGBClassifier*[27] nije polučila rezultate, dok je *GradientBoostingClassifier*[70] ostao neizvršen, nakon čak 3 sata dugotrajne optimizacije hiperparametara.

Faza Modeliranja i optimizacije modela provedena putem *MoziaisMl* sustava upućuje na *MLPClassifier[72]* kao najbolju metodu s danim hiperparametrima s obzirom na postignutu točnost, iako *RandomForestClassifier*[25] nudi kompromis između točnosti i učinkovitosti vremena izvršavanja. Metode *GaussianNB*[67] i *KNeighborsClassifier*[71] s postignutim slabim rezultatima pokazale su se kao neadekvatne za ovaj specifičan skup podataka.

6.1.4. Kvantitativna evaluacija učinka primijenjenih metoda Glasanja i Ruksaka

Navedena faza uključuje procjenu točnosti i performansi modela strojnog učenja prije donošenja odluke o njegovoj implementaciji u produkcijsko okružje. Evaluacija modela temelji se na analizi metrika uspješnosti, definiranih u ovom istraživanju - brzini i točnosti odnosno greški. Fazom Evaluacije se osigurava generalizacijska sposobnost modela na neviđenim podatcima. [11, 21, 22, 54]

6.1.4.1.Usporedna analiza modela uz i bez primjene metoda Glasanja i Ruksaka

<u>Titanic skup podataka</u>

U okviru istraživanja razvijena metoda *Glasanja* temeljila se na odabiru 20% značajki skupa podataka, gdje je odabrano 6 značajki odnosno komponenti: *Sex*, *Name2*, *Name1*, *Pclass*, *Fare* i *Ticket0*. Razvijena metoda *Ruksaka* odabrala je 5 ključnih komponenti odnosno predmeta za Ruksak: *Name1*, *Ticket*, *Age*, *Sex* i *Pclass*. Rezultati

faze Evaluacije, na neviđenim podatcima odnosno testnom skupu podataka, pokazali su da su obje razvijene metode značajno ubrzale proces izvršavanja modela *RandomForestClassifier[25]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, uz minimalne gubitke u točnosti razvijene metode *Glasanja* u odnosu na izvedbu modela, *RandomForestClassifier[25]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, bez primjene redukcije dimenzionalnosti. Razvijena metoda *Ruksaka* ima značajno brže vrijeme izvršavanja i točnost predikcije u odnosu na izvedbu modela bez primjene redukcije dimenzionalnosti.

Usporedni rezultati metoda *Glasanja* i *Ruksaka*, u odnosu na izvedbu modela bez primjene redukcije dimenzionalnosti, su prikazani tablicom u nastavku.

Razvijena metoda	**Vrijeme izvršavanja**	**Točnost**
Glasanje	-19,78%	-0,71%
Ruksak	-28,68%	+12,54%

Tablica 23: Usporedni rezultati metoda Glasanja i Ruksaka u odnosu na izvedbu modela bez primjene redukcije dimenzionalnosti

Eksperimentalni rezultati ukazuju na to da su obje razvijene metode omogućile značajno ubrzanje inferencijskog izvršavanja modela, dok su gubici u točnosti klasifikacije bili zanemarivi u odnosu na model koji nije uključivao redukciju dimenzionalnosti. Prethodno navedeni rezultati potvrđuju korisnost razvijenih metoda u poboljšanju učinkovitosti modela strojnog učenja bez značajnog kompromitiranja njegove prediktivne sposobnosti.

Razvijena metoda *Glasanja* omogućila je 19,78% ubrzanje, odnosno smanjenje, inferencijskog vremena izvršavanja, dok je gubitak u točnosti bio zanemariv, iznosivši tek 0,71% u odnosu na model bez primijenjene redukcije dimenzionalnosti.

Razvijena metoda *Ruksak* značajno poboljšava učinkovitost modela *RandomForestClassifier[25]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, omogućivši za 28,68% ubrzanje inferencijskog vremena izvršavanja, uz istovremeno povećanje točnosti klasifikacije za 12,54% u odnosu na model bez primijenjene redukcije dimenzionalnosti.

Skup podataka s cijenama nekretnina

U okviru provedenog istraživanja nad predmetnim skupom podataka razvijena metoda *Glasanja* temeljila se na odabiru 10% značajki iz skupa podataka odnosno u cijelom broju – 34 značajke odnosno komponente, razvijena metoda *Ruksaka* s 57 predmeta odnosno komponenti u visoko dimenzionalnom skupu podatka od ukupno 341 značajke nakon pretprocesiranja s fazom Priprema podataka. Odabrane značajke i predmeti odnosno komponente, prikazane su u tablici u nastavku.

Razvijena metoda *Glasanja* (34 značajke/komponente)	Razvijena metoda *Ruksaka* (57 predmeta/komponenti)
'A16_iter_score', 'GarageCars', '2ndFlrSF', 'YearBuilt_X_GarageYrBlt', 'LotArea_X_GrLivArea', 'YearRemodAdd_X_GrLivArea', 'TotalBsmtSF_X_GrLivArea', '1stFlrSF_X_GrLivArea', 'GrLivArea_X_GarageYrBlt', 'GrLivArea_X_GarageArea', 'YearBuilt_X_YearRemodAdd', 'OverallQual', 'A15_iter_score', 'YearBuilt', 'BsmtQual_Ex', 'LotArea_X_BsmtFinSF1', '1stFlrSF_X_GarageArea', 'A14_iter_score', 'TotalBsmtSF_X_GarageArea', 'YearRemodAdd', 'LotFrontage_X_GrLivArea', 'BsmtFinSF1_X_GrLivArea', 'BsmtFinSF1_X_GarageArea', '2ndFlrSF_X_GarageYrBlt', 'YearRemodAdd_X_GarageYrBlt',	*'anomaly_score_IsolationForest', 'anomaly_IsolationForest', 'TotalBsmtSF_X_GrLivArea', 'SaleCondition_Partial', 'SaleCondition_Normal', 'SaleType_WD', 'SaleType_New', 'Fence_MnPrv', 'PavedDrive_Y', 'PavedDrive_N', 'GarageCond_TA', 'GarageQual_TA', 'GarageQual_Fa', 'GarageType_BuiltIn', 'Functional_Typ', 'KitchenQual_Fa', 'KitchenQual_Ex', 'Electrical_SBrkr', 'Electrical_FuseF', 'Electrical_FuseA', 'HeatingQC_Fa', 'Heating_GasA', 'BsmtFinType2_Unf', 'BsmtExposure_Gd', 'BsmtExposure_Av', 'BsmtCond_TA', 'BsmtCond_Fa', 'BsmtQual_Fa', 'BsmtQual_Ex', 'Foundation_Slab', 'Foundation_BrkTil', 'ExterCond_TA', 'ExterCond_Fa', 'ExterQual_Fa', 'ExterQual_Ex', 'MasVnrType_Stone', 'RoofStyle_Hip',*

'LotArea_X_GarageArea', 'YearBuilt_X_GrLivArea', 'TotalBsmtSF', 'TotalBsmtSF_X_2ndFlrSF', '2ndFlrSF_X_GarageArea', 'GarageArea', 'BsmtQual_Gd', 'A12_iter_score', 'GrLivArea'	*'HouseStyle_1.5Fin', 'BldgType_TwnhsE', 'BldgType_Duplex', 'BldgType_1Fam', 'Condition1_Norm', 'Alley_Grvl', 'MSZoning_RM', 'MSZoning_FV',* ***'Fence_was_missing', 'BsmtExposure_was_missing', 'Alley_was_missing'****, 'GarageCars', 'KitchenAbvGr', '2ndFlrSF', 'CentralAir', 'OverallCond', 'OverallQual', 'LotArea', 'LotFrontage', 'MSSubClass'*

Tablica 24: Odabrane značajke/predmeti odnosno komponente razvijenih metoda Glasanja i Ruksaka

Vidljivo je iz odabranih značajki, kako je razvijena metoda *Ruksaka* pri odabiru značajki u niždimenzionalnom prostoru dala prednost novokreiranim varijablama koje su nastale kao rezultat razvijenih metoda pretprocesiranja podataka iz faze Priprema podataka. Među izdvojenim značajkama nalaze se *Fence_was_missing*, *BsmtExposure_was_missing* i *Alley_was_missing*, što ukazuje na visoku informativnu vrijednost ovih značajki u procesu predikcije.

Potrebno je istaknuti da značajka *BsmtExposure_was_missing*, koja se odnosi na ekspoziciju podruma – tehnički termin koji opisuje položaj podrumskih zidova u odnosu na vanjski teren – označena kao značajna unatoč tome što predstavlja samo indikator odsutnosti vrijednosti u izvornoj značajki *BsmtExposure* u skupu podatka. Također, prepoznavanje značajnosti varijabli *Alley_was_missing* (koja označava, u kontekstu nekretnina, odsutnost uske sporedne ili bočne prometnice) te *Fence_was_missing* (indikator nepostojanja ograde) upućuje na to da nedostajuće vrijednosti u ovom eksperimentu nad analizom podataka nose ključne informacije koje doprinose poboljšanju prediktivne sposobnosti modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela.

Navedene značajke potvrđuju valjanost metodološkog pristupa koji, u fazi Priprema podataka, uključuje razvijene metode za pretprocesiranje podataka odnosno razvijene metode za inženjerstvo značajki tijekom pretprocesiranja podataka. Razvijene metode, još jednom su pokazale kako omogućuju optimizaciju reprezentacije važnosti

značajki u skupu podataka u svrhu preciznije predikcije modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, čime se dodatno opravdava primjena razvijenih metoda za pretprocesiranje podataka u *MoziaisMl* sustavu.

Rezultati faze Evaluacije, na podatcima, pokazali su da su obje razvijene metode značajno ubrzale proces inferencijskog izvršavanja modela s metodom *DecisionTreeRegressor*[80] i odabranim hiperparametrima iz faze Modeliranje i optimizacija modela.

Usporedni rezultati metoda *Glasanja* i *Ruksaka*, u odnosu na izvedbu modela bez primjene redukcije dimenzionalnosti, su prikazani tablicom u nastavku.

Razvijena metoda	**Vrijeme izvršavanja**	**RMSE**
Glasanje	- 147,61%	+1,68%
Ruksak	- 110,27%	- 9,08%

Tablica 25: Usporedni rezultati metoda Glasanja i Ruksaka u odnosu na izvedbu modela bez primjene redukcije dimenzionalnosti

Usporedni rezultati, prikazani u tablici, su učinci izvedbe dviju razvijenih metoda *Glasanja* i *Ruksaka* u odnosu na model *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, bez primjene redukcije dimenzionalnosti, uzimajući u obzir dvije ključne metrike ovog istraživanja: vrijeme inferencijskog izvršavanja i točnost modela. Razvijene metode *Glasanja* i *Ruksaka* značajno su smanjile inferencijsko vrijeme izvršavanja, odnosno korištenje računalnih resursa, u usporedbi s modelom bez primjene redukcije dimenzionalnosti. Razvijena metoda *Glasanja* ostvarila je smanjenje inferencijskog vremena izvršavanja za 147,61%, dok je razvijena metoda *Ruksaka* rezultirala smanjenjem od 110,27%. Rezultati dviju razvijenih metoda *Glasanja* i *Ruksaka* ukazuju na to da su obje razvijene metode značajno povećale računalnu učinkovitost odnosno smanjenje korištenja računalnih resursa, pri čemu je razvijena metoda *Glasanja* bila učinkovitija u smanjenju inferencijskog vremena izvršavanja. Analiza preciznosti modela odnosno smanjenja RMSE greške pokazuje različite rezultate između razvijenih metoda *Glasanja* i *Ruksaka*. Metoda *Glasanja* zabilježila je porast RMSE greške od 1,68%, što upućuje na minimalni gubitak

prediktivne sposobnosti uz značajnu optimizaciju brzine izvođenja. Suprotno tome, metoda *Ruksaka* ostvarila je smanjenje RMSE greške od 9,08%, što sugerira da ova metoda ne samo da poboljšava računalnu učinkovitost odnosno korištenje računalnih resursa, već i povećava prediktivnu preciznost modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela.

Usporedni rezultati ukazuju na to da su razvijene metode *Glasanja* i *Ruksaka* omogućile značajnu optimizaciju inferencijskog vremena izvođenja modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela. Međutim, dok razvijena metoda *Glasanja* omogućuje najveće ubrzanje inferencijskog izvođenja uz minimalan porast RMSE greške, razvijena metoda *Ruksaka* istovremeno osigurava značajno poboljšanje prediktivne preciznosti, uz nešto manju, ali i dalje značajnu optimizaciju inferencijskog vremena izvršavanja u odnosu na model *DecisionTreeRegressor*[80], s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, bez primjene redukcije dimenzionalnosti.

Skup podataka s cjepivima

U okviru provedenog istraživanja nad predmetnim skupom podataka razvijena metoda *Glasanja* temeljila se na odabiru 50% značajki iz skupa podataka. U slučajevima kada broj značajki nije cijeli broj (primjer, 50% od 7 značajki iznosi 3,5, a uzima cijeli broj, bez zaokruživanja - 3), razvijena metoda se prilagođava zaokružujući broj odabranih značajki na 3. S druge strane, razvijena metoda *Ruksaka* odabrala je tri predmeta odnosno značajke. Obje razvijene metode rezultirale su identičnim odabranim značajkama: *Maxd1delta Penh*, *TD Penh* i *Weight Delta*. Rezultati faze Evaluacije, na podatcima, pokazali su da su obje razvijene metode značajno ubrzale proces izvršavanja modela s metodom *MLPClassifier[72]* i utvrđenim hiperparametrima iz faze Modeliranje i optimizacija modela.

Usporedni rezultati metoda *Glasanja* i *Ruksaka*, u odnosu na izvedbu modela bez primjene redukcije dimenzionalnosti, su prikazani tablicom u nastavku.

Razvijena metoda	Vrijeme izvršavanja	Točnost
Glasanje i Ruksak	- 157,97%	+ 0,06%

Tablica 26: Usporedni rezultati metoda Glasanja i Ruksaka u odnosu na izvedbu modela bez primjene redukcije dimenzionalnosti

Eksperimentalni rezultati jasno ukazuju na značajno poboljšanje učinkovitosti modela strojnog učenja prilikom primjene razvijenih metoda *Glasanja* i *Ruksaka*. Konkretno, razvijene metode omogućile su 157,97% ubrzanje, odnosno smanjenje, inferencijskog vremena izvršavanja, a postojao je i dobitak u točnosti klasifikacije, viša točnost za 0,06% u odnosu na model bez primijenjene redukcije dimenzionalnosti. Dobiveni rezultati potvrđuju da su razvijene metode omogućile optimizaciju računalnih resursa i vremenske složenosti modela, pri čemu je očuvana te malo i poboljšana prediktivna sposobnost, što ih čini korisnima u kontekstu modeliranja i optimizacije nad višeklasifikacijskim skupovima podataka.

6.1.5. Implementacija i testiranje generalizacijske robusnosti MoziaisMl sustava

Nakon odabira optimalnog modela, slijedi njegova integracija u produkcijsko okružje, uz usporednu analizu učinkovitosti u odnosu na popularne i prihvaćene tehnike redukcije dimenzionalnosti, poput PCA[14-16], t-SNE[14-17], UMAP[18] i autoenkoderskih neuronskih mreža[19]. Cilj ove usporedne evaluacije jest procjena generalizacijskih sposobnosti razvijenih metoda *Glasanja* i *Ruksaka* na podatkovnim skupovima iz različitih domena u odnosu na popularne i prihvaćene tehnike redukcije dimenzionalnosti, kao i robusnost razvijenih metoda *Glasanja* i *Ruksaka* u stvarnim uvjetima primjene.

6.1.5.1. Evaluacija stabilnosti i robusnosti MoziaisMl sustava u odnosu na druge metode redukcije dimenzionalnosti

Titanic skup podataka

Usporedna analiza učinkovitosti razvijenih metoda *Glasanja* i *Ruksaka* s popularnim tehnikama redukcije dimenzionalnosti PCA[14-16], t-SNE[14-17], UMAP [18] i autoenkoderskim neuronskim mrežama [19], prikazana je tablicom u nastavku.

Tehnike redukcije dimenzionalnosti	Razvijena metoda Glasanja (6 komponenti/značajki)		Razvijena metoda Ruksaka (5 komponenti/značajki)	
	Vrijeme izvršavanja(ms)	Točnost	Vrijeme izvršavanja(ms)	Točnost
PCA	+33,43%	-8,12%	+38,75%	-15,38%
t-SNE	-20,44%	-2,87%	+30,91%	-19,79%
UMAP	+19,99%	-8,12%	+43,41%	-19,04%
Autoenkoderske neuronske mreže	+15,57%	+15,69%	+27,62%	+15,47%

Tablica 27: Usporedna analiza učinkovitosti razvijenih metoda Glasanja i Ruksaka s popularnim tehnikama redukcije dimenzionalnosti

Prikazani rezultati, u prethodnoj tablici, uspoređuju korištene i prihvaćene tehnike redukcije dimenzionalnosti – PCA, UMAP, Autoenkoderske mreže i t-SNE - u odnosu na razvijene metode *Glasanja* (s 6 značajki odnosno komponenti) i *Ruksaka* (s 5 predmeta odnosno komponenti), pri čemu se promatraju promjene u inferencijskom vremenu izvršavanja i točnosti modela.

U usporedbi s metodom *Glasanja* (s 6 značajki odnosno komponenti), metoda PCA, s 6 značajki odnosno komponenti, je rezultirala povećanjem inferencijskog vremena izvršavanja za 33,43%, što ukazuje na značajnije korištenje računalnih resursa. Istovremeno, smanjenje točnosti od 8,12% ukazuje da PCA metoda gubi relevantne informacije tijekom redukcije dimenzionalnosti značajki u niždimenzionalni prostor. Metoda t-SNE, s 6 značajki odnosno komponenti, pokazala je smanjenje inferencijskog vremena izvršavanja za 20,44%, što je pozitivna karakteristika metode u upotrebi modela *RandomForestClassifier[25]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, ali uz istovremeni pad točnosti od 2,87%, što ukazuje na gubitak količine informacija iz niždimenzionalnog prostora. Metoda UMAP, s 6 značajki odnosno komponenti, je polučila negativno povećanje inferencijskog vremena izvršavanja za 19,99%, dok je točnost smanjena za 8,12%, što ukazuje na slične nedostatke kao kod metode PCA, ali uz nešto bolje performanse u brzini izvršavanja. Metoda Autoenkoderskih neuronskih mreža značajno, s 6 značajki odnosno komponenti, je povećala točnost za 15,69%, što predstavlja pozitivnu karakteristiku, dok je inferencijsko vrijeme izvršavanja povećano za 15,57% - što je negativna karakteristika,

rezultat Autoenkoderskih neuronskih mreža ukazuje na poboljšanje prediktivne sposobnosti modela *RandomForestClassifier[25]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela uz povećano korištenje računalnih resursa.

U usporedbi s metodom *Ruksaka* (s 5 predmeta odnosno komponenti), metoda PCA, s 5 komponenti, je povećala inferencijsko vrijeme izvršavanja za 38,75%, uz istovremeni pad točnosti za 15,38%, što ukazuje na neefikasnost metode u ovom klasifikacijskom skupu podatka. Metoda t-SNE, s 5 komponenti, je povećala inferencijsko vrijeme izvršavanja za 30,91%, uz istovremeno, značajno, smanjenje točnosti od 19,79%, što ukazuje na negativan utjecaj gubitka informacija iz niždimenzionalnog prostora. Metoda UMAP, s 5 komponenti, je povećala inferencijsko vrijeme izvršavanja za 43,41%, dok je točnost pala za 19,04%, što ukazuje na manju učinkovitost metode UMAP u ovom klasifikacijskom skupu podatka. Metoda Autoenkoderskih neuronskih mreža, s 5 komponenti, je povećala inferencijsko vrijeme izvršavanja za 27,62%, ali uz istovremeno poboljšanje točnosti za 15,47%, što Autoenkoderske neuronske mreže čini najuspješnijom metodom u smislu očuvanja informacija iz niždimenzionalnog prostora kod korištenja modela *RandomForestClassifier[25]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela.

Prikazani rezultati pokazuju da su metode PCA, UMAP i t-SNE rezultirale smanjenjem točnosti, dok su istovremeno, većinom povećale inferencijsko vrijeme izvršavanja, izuzev metode t-SNE u usporedbi s razvijenom metodom *Glasanja*, koja je smanjila inferencijsko vrijeme izvršavanja. S druge strane, metoda Autoenkoderskih neuronskih mreža je jedina metoda koja je značajno poboljšala točnost, ali uz veće inferencijsko vrijeme izvršavanja odnosno uz cijenu značajnijeg korištenja računalnih resursa. Razvijene metode *Glasanja* i *Ruksaka* pokazale su bolji balans između inferencijskog vremena izvršavanja i točnosti u usporedbi s korištenim i prihvaćenim tehnikama redukcije dimenzionalnosti – PCA, UMAP, Autoenkoderske mreže i t-SNE.

Skup podataka s cijenama nekretnina

Usporedna analiza učinkovitosti razvijenih metoda *Glasanja* i *Ruksaka* s popularnim tehnikama redukcije dimenzionalnosti PCA[14-16], t-SNE[14-17], UMAP [18] i autoenkoderskim neuronskim mrežama [19], prikazana je tablicom u nastavku.

Tehnike redukcije dimenzionalnosti	Razvijena metoda Glasanja (34 komponente/značajke)		Razvijena metoda Ruksaka (57 komponenti/značajki)	
	Vrijeme izvršavanja(ms)	RMSE	Vrijeme izvršavanja(ms)	RMSE
PCA	+20,78%	+29,15%	+3,97%	+31,84%
t-SNE	-	-	-	-
UMAP	-	-	-	-
Autoenkoderske neuronske mreže	-32,87%	+33,56%	-56,54%	+ 38.32%

Tablica 28: Usporedna analiza učinkovitosti razvijenih metoda Glasanja i Ruksaka s popularnim tehnikama redukcije dimenzionalnosti

Usporednom analizom učinkovitosti, prikazanom u tablici, procjenjuje se učinak korištenih i prihvaćenih tehnika redukcije dimenzionalnosti – PCA, UMAP, Autoenkoderske mreže i t-SNE - u odnosu na razvijene metode *Glasanja* (s 34 značajke odnosno komponente) i *Ruksaka* (s 57 predmeta odnosno komponenti). Pritom se vrednuju promjene u inferencijskom vremenu izvršavanja i RMSE-u, pri čemu se niže RMSE vrijednosti smatraju poželjnima jer ukazuju na veću preciznost modela, dok niže vrijednosti inferencijskog vremena izvršavanja upućuju na bolju računalnu učinkovitost kod korištenja dostupnih računalnih resursa.

U usporedbi s razvijenom metodom *Glasanja* (s 34 značajke odnosno komponente), metoda PCA, s 34 značajke odnosno komponente, rezultira s povećanjem inferencijskog vremena izvršavanja za 20,78%, odnosno upućuje na značajnije korištenje računalnih resursa u odnosu na razvijenu metodu *Glasanja*. Istovremeno, RMSE pogreška je porasla za 29,15%, što ukazuje na degradaciju modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, u pogledu točnosti predikcije. Metoda Autoenkoderskih neuronskih mreža, s 34 značajke odnosno komponente, je značajno smanjila inferencijsko vrijeme izvršavanja za 32,87%, što ukazuje na poboljšanu računalnu učinkovitost odnosno manje

korištenje računalnih resursa. Međutim, RMSE pogreška se povećala za 33,56%, što ukazuje na gubitak informacija u niždimenzionalnom prostoru i slabije predikcijske performanse modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela.

U usporedbi s razvijenom metodom *Ruksaka* (s 57 predmeta odnosno komponenti), metoda PCA, s 57 značajki odnosno komponenti, je rezultirala povećanjem inferencijskog vremena izvršavanja za 3,97%, što je prihvatljivo povećanje iskorištavanja računalnih resursa, ali je istovremeno prouzročila povećanje RMSE pogreške za 31,84%, što ukazuje na smanjenu prediktivnu sposobnost modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela. Metoda Autoenkoderskih neuronskih mreža, s 57 značajki odnosno komponenti, je rezultirala sa značajnim poboljšanjem računalne učinkovitosti odnosno smanjenja iskorištavanja računalnih resursa, smanjujući inferencijsko vrijeme izvršavanja za 56,54%, ali uz istovremeno povećanje RMSE pogreške za 38,32%, što upućuje na degradaciju modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela u smislu preciznosti predikcije.

Metode t-SNE i UMAP, u ovom regresijskom skupu podataka nisu učinkovite metode za redukciju dimenzionalnosti jer je njihovo inferencijsko izvršavanje trajalo više od 3 sata, nakon čega je prekinuta obrada podataka s t-SNE i UMAP metodama.

Usporedna analiza učinkovitosti ukazuje na to da metoda PCA, s 34 i 57 komponenti, značajno povećava RMSE pogrešku, čime smanjuje prediktivnu preciznost modela, dok metoda Autoenkoderskih neuronskih mreža, s 34 i 57 komponenti, smanjuje inferencijsko vrijeme izvršavanja odnosno upotrebu računalnih resursa, ali uz znatno povećanje RMSE pogreške, što ukazuje na kompromis između, superiorne, računalne učinkovitosti i preciznosti modela. Razvijene metode *Glasanja* i *Ruksaka*, iako nisu uvijek najbrže, osigurale su nižu RMSE pogrešku u usporedbi s korištenim i prihvaćenim tehnika redukcije dimenzionalnosti – PCA, UMAP, Autoenkoderske mreže i t-SNE, što potvrđuje njihovu robusnost u očuvanju informacija kroz odabir ključnih značajki u niždimenzionalnom prostoru za preciznost predikcije modela *DecisionTreeRegressor*[80] s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela.

Skup podataka s cjepivima

Usporedna analiza učinkovitosti razvijenih metoda *Glasanja* i *Ruksaka* s popularnim tehnikama redukcije dimenzionalnosti PCA[14-16], t-SNE[14-17], UMAP [18] i autoenkoderskim neuronskim mrežama [19], prikazana je tablicom u nastavku.

Tehnike redukcije dimenzionalnosti (3 komponente)	Razvijena metoda Glasanja i Ruksaka	
	Vrijeme izvršavanja	Točnost
PCA[14-16]	- 9.98%	- 38,46%
t-SNE[14-17]	-	-
UMAP [18]	- 51,23%	- 41%
Autoenkoderske neuronske mreže [19]	- 15,09%	- 41%

Tablica 29: Usporedna analiza učinkovitosti razvijenih metoda Glasanja i Ruksaka s popularnim tehnikama redukcije dimenzionalnosti

Usporednom analizom učinkovitosti, prikazanom u tablici, uočava se kako korištene i prihvaćene tehnike redukcije dimenzionalnosti – PCA, UMAP, Autoenkoderske mreže i t-SNE - utječu na vrijeme izvršavanja i točnost modela, u usporedbi s razvijenim metodama *Glasanja* i *Ruksaka*. Metoda PCA pokazala je smanjenje inferencijskog vremena izvršavanja za 9,98%, što ukazuje na poboljšanje u brzini modela *MLPClassifier[72]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela. Međutim, točnost PCA metode je značajno smanjena, i to za 38,46%, što ukazuje na gubitak informacija relevantnih za odabrani višeklasifikacijski zadatak. Unatoč ubrzanju modela, ovakav pad točnosti čini metodu PCA nepovoljnijim izborom u odnosu na razvijene metode *Glasanja* i *Ruksaka*.

Nastavno, metoda UMAP dovela je, isto, do smanjenja inferencijskog vremena izvršavanja za 51,23%, što ukazuje na značajno smanjenje korištenja računalnih resursa potrebnih za analizu podataka. Istovremeno, točnost metode UMAP smanjena je za 41%,

što sugerira gubitak bitnih značajki za predikciju, nakon redukcije dimenzionalnosti. Rezultati ukazuju na to da metoda UMAP u ovom višeklasifikacijskom zadatku nije učinkovita metoda optimizacije modela *MLPClassifier[72]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela.

Autoenkoderske neuronske mreže također su rezultirale smanjenjem inferencijskog vremena izvršavanja za 15,09%, dok je točnost smanjena za 41%, što sugerira da Autoenkoderske neuronske mreže nisu uspjele očuvati relevantne značajke u višeklasifikacijskom skupu podataka, dok su, s druge strane, u smanjenom obujmu koristile računalne resurse.

Metoda t-SNE, iako pokazuje dobre rezultate u vizualizaciji podataka, u ovom višeklasifikacijskom skupu podataka nije učinkovita metoda za redukciju dimenzionalnosti jer je njezino inferencijsko izvršavanje trajalo više od 3 sata, nakon čega je prekinuta analiza podataka s t-SNE metodom.

Na temelju usporedne analize učinkovitosti, zaključuje se da su razvijene metode *Glasanja* i *Ruksaka* superiornije u odnosu na korištene i prihvaćene tehnike redukcije dimenzionalnosti – PCA, UMAP, Autoenkoderske mreže i t-SNE jer omogućuju zadržavanje više informacija uz niže inferencijsko vrijeme izvršavanja. Metode PCA, UMAP i Autoenkoderske mreže iako poboljšavaju brzinu modela *MLPClassifier[72]* s odabranom parametrizacijom iz faze Modeliranje i optimizacija modela, s druge strane, značajno smanjuju njegovu prediktivnu sposobnost.

6.2. Sažeti rezultati MoziaisMl sustava

Na temelju prikazanih rezultata, u tablici u nastavku, dana je sažeta diskusija rezultata koja obuhvaća analizu i interpretaciju učinkovitosti razvijenih metoda redukcije dimenzionalnosti unutar sustava *MoziaisMl*, s fokusom na usporedbu između cjelokupne dimenzionalnosti skupa podataka, metode *Glasanja* i metode *Ruksaka*. Rezultati su prikazani za tri različita skupa podataka: Titanic (binarna klasifikacija s velikim disbalansom klasa), House Prices (regresija) i Vaccine (višeklasna klasifikacija).

Skup podataka	Titanic	House Prices	Vaccine
Skup podataka cjelokupne dimenzije	Survived, Pclass, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin, Age_was_missing, Cabin_was_missing, Name0, Name1, Name2, Name3, Name4, Name5, Name6, Name7, Ticket0, Ticket1, Ticket2, Cabin0, Cabin1, Cabin3, Embarked_C, Embarked_Q, Embarked_S, Name8_0, Cabin2_0, Cabin2_B55, Cabin2_B63, Cabin2_C27	MSSubClass, LotFrontage, LotArea, Street, Utilities, Neighborhood, OverallQual, OverallCond, YearBuilt, YearRemodAdd, Exterior1st, Exterior2nd, MasVnrArea, BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, CentralAir, 1stFlrSF, 2ndFlrSF, LowQualFinSF, GrLivArea, BsmtFullBath, BsmtHalfBath, FullBath, HalfBath, BedroomAbvGr, KitchenAbvGr, TotRmsAbvGrd, Fireplaces, GarageYrBlt, GarageCars, GarageArea, WoodDeckSF, OpenPorchSF, EnclosedPorch, 3SsnPorch, ScreenPorch, PoolArea, MiscVal, MoSold, YrSold, SalePrice, LotFrontage_was_missing, Alley_was_missing, MasVnrType_was_missing, MasVnrArea_was_missing, BsmtQual_was_missing, BsmtCond_was_missing, BsmtExposure_was_missing, BsmtFinType1_was_missing, BsmtFinType2_was_missing, FireplaceQu_was_missing, GarageType_was_missing, GarageYrBlt_was_missing, GarageFinish_was_missing, GarageQual_was_missing, GarageCond_was_missing, PoolQC_was_missing, Fence_was_missing, MiscFeature_was_missing, MSZoning1, Exterior1st0, Exterior1st1, Exterior2nd0, MSZoning_C (all), MSZoning_FV, MSZoning_RH, MSZoning_RL, MSZoning_RM, Alley_0, Alley_Grvl, Alley_Pave,	Vaccine, Maxd1delta Penh, STD Penh, Weight Delta, Construct, Dose, Dose0, Dose1

		LotShape_IR1, LotShape_IR2, LotShape_IR3, LotShape_Reg, LandContour_Bnk, LandContour_HLS, LandContour_Low, LandContour_Lvl, LotConfig_Corner, LotConfig_CulDSac, LotConfig_FR2, LotConfig_FR3, LotConfig_Inside, LandSlope_Gtl, LandSlope_Mod, LandSlope_Sev, Condition1_Artery, Condition1_Feedr, Condition1_Norm, Condition1_PosA, Condition1_PosN, Condition1_RRAe, Condition1_RRAn, Condition1_RRNe, Condition1_RRNn, Condition2_Artery, Condition2_Feedr, Condition2_Norm, Condition2_PosA, Condition2_PosN, BldgType_1Fam, BldgType_2fmCon, BldgType_Duplex, BldgType_Twnhs, BldgType_TwnhsE, HouseStyle_1.5Fin, HouseStyle_1.5Unf, HouseStyle_1Story, HouseStyle_2.5Unf, HouseStyle_2Story, HouseStyle_SFoyer, HouseStyle_SLvl, RoofStyle_Flat, RoofStyle_Gable, RoofStyle_Gambrel, RoofStyle_Hip, RoofStyle_Mansard, RoofStyle_Shed, RoofMatl_CompShg, RoofMatl_Tar&Grv,	

		RoofMatl_WdShake, RoofMatl_WdShngl, MasVnrType_0, MasVnrType_BrkCmn, MasVnrType_BrkFace, MasVnrType_Stone, ExterQual_Ex, ExterQual_Fa, ExterQual_Gd, ExterQual_TA, ExterCond_Ex, ExterCond_Fa, ExterCond_Gd, ExterCond_Po, ExterCond_TA, Foundation_BrkTil, Foundation_CBlock, Foundation_PConc, Foundation_Slab, Foundation_Stone, Foundation_Wood, BsmtQual_0, BsmtQual_Ex, BsmtQual_Fa, BsmtQual_Gd, BsmtQual_TA, BsmtCond_0, BsmtCond_Fa, BsmtCond_Gd, BsmtCond_Po, BsmtCond_TA, BsmtExposure_0, BsmtExposure_Av, BsmtExposure_Gd, BsmtExposure_Mn, BsmtExposure_No, BsmtFinType1_0, BsmtFinType1_ALQ, BsmtFinType1_BLQ, BsmtFinType1_GLQ, BsmtFinType1_LwQ, BsmtFinType1_Rec, BsmtFinType1_Unf, BsmtFinType2_0, BsmtFinType2_ALQ, BsmtFinType2_BLQ, BsmtFinType2_GLQ, BsmtFinType2_LwQ, BsmtFinType2_Rec, BsmtFinType2_Unf, Heating_GasA, Heating_GasW, Heating_Grav, Heating_Wall, HeatingQC_Ex, HeatingQC_Fa, HeatingQC_Gd, HeatingQC_Po, HeatingQC_TA,	

		Electrical_FuseA, Electrical_FuseF, Electrical_FuseP, Electrical_SBrkr, KitchenQual_Ex, KitchenQual_Fa, KitchenQual_Gd, KitchenQual_TA, Functional_Maj1, Functional_Maj2, Functional_Min1, Functional_Min2, Functional_Mod, Functional_Sev, Functional_Typ, FireplaceQu_0, FireplaceQu_Ex, FireplaceQu_Fa, FireplaceQu_Gd, FireplaceQu_Po, FireplaceQu_TA, GarageType_0, GarageType_2Types, GarageType_Attchd, GarageType_Basment, GarageType_BuiltIn, GarageType_CarPort, GarageType_Detchd, GarageFinish_0, GarageFinish_Fin, GarageFinish_RFn, GarageFinish_Unf, GarageQual_0, GarageQual_Fa, GarageQual_Gd, GarageQual_Po, GarageQual_TA, GarageCond_0, GarageCond_Ex, GarageCond_Fa, GarageCond_Gd, GarageCond_Po, GarageCond_TA, PavedDrive_N, PavedDrive_P, PavedDrive_Y, PoolQC_0, PoolQC_Ex, PoolQC_Gd, Fence_0, Fence_GdPrv, Fence_GdWo, Fence_MnPrv,	

		Fence_MnWw, MiscFeature_0, MiscFeature_Gar2, MiscFeature_Othr, MiscFeature_Shed, SaleType_COD, SaleType_CWD, SaleType_Con, SaleType_ConLD, SaleType_ConLI, SaleType_ConLw, SaleType_New, SaleType_Oth, SaleType_WD, SaleCondition_Abnorml, SaleCondition_AdjLand, SaleCondition_Alloca, SaleCondition_Family, SaleCondition_Normal, SaleCondition_Partial, MSZoning0_C, MSZoning0_FV, MSZoning0_RH, MSZoning0_RL, MSZoning0_RM, Exterior2nd1_0, Exterior2nd1_Cmn, Exterior2nd1_Sdng, Exterior2nd1_Shng	
Metoda Glasanja - odabrane značajke	Sex, Name2, Name1, Pclass, Fare, Ticket0	A16_iter_score, GarageCars, 2ndFlrSF, YearBuilt_X_GarageYrBlt, LotArea_X_GrLivArea, YearRemodAdd_X_GrLivArea, TotalBsmtSF_X_GrLivArea, 1stFlrSF_X_GrLivArea, GrLivArea_X_GarageYrBlt, GrLivArea_X_GarageArea, YearBuilt_X_YearRemodAdd, OverallQual, A15_iter_score, YearBuilt, BsmtQual_Ex, LotArea_X_BsmtFinSF1, 1stFlrSF_X_GarageArea, A14_iter_score, TotalBsmtSF_X_GarageArea, YearRemodAdd, LotFrontage_X_GrLivArea,	Maxd1delta Penh, TD Penh, Weight Delta

		BsmtFinSF1_X_GrLivArea, BsmtFinSF1_X_GarageArea, 2ndFlrSF_X_GarageYrBlt, YearRemodAdd_X_GarageYr Blt, LotArea_X_GarageArea, YearBuilt_X_GrLivArea, TotalBsmtSF, TotalBsmtSF_X_2ndFlrSF, 2ndFlrSF_X_GarageArea, GarageArea, BsmtQual_Gd, A12_iter_score, GrLivArea	
Metoda Ruksaka - odabrane značajke	Name1, Ticket, Age, Sex, Pclass	anomaly_score_IsolationForest , anomaly_IsolationForest, TotalBsmtSF_X_GrLivArea, SaleCondition_Partial, SaleCondition_Normal, SaleType_WD, SaleType_New, Fence_MnPrv, PavedDrive_Y, PavedDrive_N, GarageCond_TA, GarageQual_TA, GarageQual_Fa, GarageType_BuiltIn, Functional_Typ, KitchenQual_Fa, KitchenQual_Ex, Electrical_SBrkr, Electrical_FuseF, Electrical_FuseA, HeatingQC_Fa, Heating_GasA, BsmtFinType2_Unf, BsmtExposure_Gd, BsmtExposure_Av, BsmtCond_TA, BsmtCond_Fa, BsmtQual_Fa, BsmtQual_Ex, Foundation_Slab, Foundation_BrkTil, ExterCond_TA, ExterCond_Fa, ExterQual_Fa, ExterQual_Ex, MasVnrType_Stone, RoofStyle_Hip, HouseStyle_1.5Fin, BldgType_TwnhsE, BldgType_Duplex, BldgType_1Fam, Condition1_Norm, Alley_Grvl, MSZoning_RM, MSZoning_FV,	

		Fence_was_missing, BsmtExposure_was_missing, Alley_was_missing, GarageCars, KitchenAbvGr, 2ndFlrSF, CentralAir, OverallCond, OverallQual, LotArea, LotFrontage, MSSubClass	
Vrijeme inferencije (ms) - bazna metoda/model bez redukcije dimenzionalnosti	57,53	172,68	1220,54
Vrijeme inferencije (ms) – Metoda Glasanja	47,18	26,02	
Vrijeme inferencije (ms) – Metoda Ruksaka	43,10	49,94	143,29
Točnost/RMSE - bazna metoda/model bez redukcije dimenzionalnosti	0,3397	41544,31	0,3761
Točnost/RMSE – Metoda Glasanja	0,3373	42249,71	
Točnost/RMSE – Metoda Ruksaka	0,3852	37933,15	0,3763

Tablica 30: Sažeti rezultati MoziaisMl sustava

Za svaki skup podataka prikazane su: korištene značajke, vrijeme izvođenja (inferencijsko vrijeme u milisekundama) i metričke vrijednosti točnosti (za klasifikacijske zadatke) odnosno greške RMSE (za regresijske zadatke).

Kod skupa podataka Titanic, primjećuje se da obje razvijene metode redukcije dimenzionalnosti - *Glasanja* i *Ruksaka* - značajno smanjuju vrijeme inferencije u usporedbi s baznom metodom/modelom (47,18 ms i 43,10 ms naspram 57,53 ms), dok točnost ostaje ista ili čak blago poboljšana (0,3373 za metodu Glasanja te 0,3852 za

metodu Ruksaka u odnosu na 0,3397 bazne metode/modela). Time se potvrđuje učinkovitost predloženih metoda u smanjenju latencije uz očuvanje prediktivne preciznosti.

Kod regresijskog skupa House Prices, također dolazi do značajnog smanjenja inferencijskog vremena — osobito kod metode *Glasanja* (26,02 ms u odnosu na 172,68 ms bazne metode/modela), uz očuvanje performansi slične razine (RMSE je 42249,71 u odnosu na 41544,31), dok metoda *Ruksaka* dodatno poboljšava RMSE na 37933,15, iako uz nešto veće inferencijsko vrijeme izvođenja nego kod metode *Glasanja* (49,94 ms). Pritom, rezultat ukazuje na dodatnu prednost metode Ruksaka u slučajevima gdje je preciznost primarni kriterij, a ne samo latencija.

Skup Vaccine pokazuje slične obrasce rezultata – metoda *Glasanja* i *Ruksaka* postižu značajno smanjenje inferencijskog vremena izvođenja (143,29 ms naspram 1220,54 ms bazne metode/modela), uz zadržavanje točnosti na istoj razini (0,3763). Ovi rezultati potvrđuju učinkovitost metoda *Glasanja* i *Ruksaka* u višeklasifikacijskim scenarijima.

Zaključno, prikazani sažeti rezultati *MoziaisMl* sustava, u prethodnoj tablici, potvrđuju cilj i svrhu istraživanja prema kojoj vlastito razvijene metode redukcije dimenzionalnosti unutar *MoziaisMl* sustava doprinose smanjenju inferencijskog vremena izvođenja predikcije bez značajnog kompromisa u prediktivnoj točnosti. Ovakav višekriterijski pristup evaluaciji - gdje latencija ima veći primat nad neznatnim dobicima u točnosti/greški - omogućuje optimalnu primjenu *MoziaisMl* sustava u vremenski osjetljivim uvjetima različitih poslovnim domena produkcijskih okružja.

6.3. Funkcija cilja

Funkcija cilja provodi evaluaciju performansi algoritma latencije unutar *MoziaisMl* sustava, uzimajući u obzir ključne parametre koji utječu na brzinu izvršavanja i preciznost predikcija. Funkcija prima inferencijsko vrijeme izvršavanja algoritma, gdje se prvi rang važnosti postavlja na kraće inferencijsko vrijeme izvršavanja. Nadalje, uzima se u obzir utjecaj latencije algoritma na točnost predikcija, gdje je cilj minimizirati degradaciju predikcijske preciznosti uzrokovanu brzinom izvršavanja.

Uz prethodno navedeno, funkcija uključuje težinski faktor alpha, koji omogućuje prilagodbu kompromisa između brzine izvršavanja i očuvanja predikcijske točnosti, pri čemu se njegova vrijednost nalazi unutar intervala 0 i 1. Obzirom na prvi rang kriterija inferencijskog vremena izvršavanja odnosno brzine, autor ovog rada je parametar alpha (α) postavio na 0,7 ($\alpha = 0.7$).

S funkcijom cilja omogućena je kvantitativna usporedba između modela bez redukcije dimenzionalnosti, tehnika redukcije dimenzionalnosti PCA[14-16], t-SNE[14-17], UMAP [18] i autoenkoderskim neuronskim mrežama [19] te razvijenih metoda *Glasanja* i *Ruksaka*.

Kao izlazna vrijednost, funkcija vraća optimizacijsku vrijednost koja kvantificira ukupnu učinkovitost algoritma latencije, pri čemu niža vrijednost označava bolji balans između brzine i preciznosti.

Kvantitativna usporedba inferencijskog vremena izvršavanja i točnosti između modela bez redukcije dimenzionalnosti, tehnika redukcije dimenzionalnosti PCA[14-16], t-SNE[14-17], UMAP [18] i autoenkoderskim neuronskim mrežama [19] te razvijenih metoda *Glasanja* i *Ruksaka*, prikazana je tablicama u nastavku.

Titanic skup podataka

U nastavku je prikazana funkcija cilja za razvijenu metodu *Glasanja* pri uzimanju 6 značajki odnosno komponenti u niždimenzionalnom prostoru.

Rbr.	Metoda	Vrijeme izvršavanja (ms)
1.	t-SNE	38,43
2.	Metoda Glasanja	47,18
3.	Autoenkoderske neuronske mreže	55,14
4.	Model bez redukcije dimenzionalnosti	57,53
5.	UMAP	57,65
6.	PCA	66,12

Tablica 31: Vrijeme izvršavanja metoda kod 6 značajki/komponenti

Parametri minimalnog i maksimalnog praga vremena izvršavanja za funkciju cilja, definirani su kao referentne vrijednosti za normalizaciju podataka, gdje se osigurava konzistentna usporedivost između različitih scenarija izvršavanja. Pritom za minimalni prag su 38,43 milisekundi (ms), dok su za maksimalni 66,12 milisekunde (ms).

Rbr.	Metoda	Točnost
1.	Autoenkoderske neuronske mreže	0,3947
2.	Model bez redukcije dimenzionalnosti	0,3397
3.	Metoda Glasanja	0,3373
4.	t-SNE	0,3278
5.	PCA	0,3110
6.	UMAP	0,3110

Tablica 32:Točnost predikcije između metoda kod 6 značajki/komponenti

Parametri najmanjeg i najvećeg utjecaja na preciznost za funkciju cilja, definirani su kao referentne vrijednosti koje omogućuju kalibraciju modela u odnosu na raspon utjecaja na predikcijske performanse točnosti. Pritom za najmanji utjecaj su 0,394736842, dok su za najveći 0,311004785.

Funkcija cilja za razvijenu metodu *Glasanja*, sastoji se od 3 procesne radnje, prikazane u nastavku:

1. Ulazni podatci:
    - Minimalni prag vremena izvršavanja – 38,43 ms
    - Maksimalni prag vremena izvršavanja – 66,12 ms
    - Najmanji utjecaj na preciznost - 0,394736842
    - Najveći utjecaj na preciznost - 0,311004785
    - Vrijeme izvršavanja razvijene metode Glasanja - 47,18 ms
    - Predikcija točnosti/preciznosti razvijene metode Glasanja - 0,337320574
2. Normalizacija podataka: svaka vrijednost se skalira u raspon od 0 do 1 kako bi imali istu važnost
    - Vrijeme izvršavanja

$$T_{norm} = \frac{47,18 - 38,43}{66,12 - 38,43} = \frac{8,75}{27,69} = 0,316 \tag{70}$$

    - Predikcija točnosti/preciznosti

$$I_{norm} = \frac{0,337320574 - 0,394736842}{0,311004785 - 0,394736842} = \frac{-0,055023923}{-0,083732057} = 0,657 \tag{71}$$

3. Funkcija cilja

$$F(47.18, 0.337320574) = 0,7 * 0,316 + (1-0,7) * 0,657 \\ = 0,2212 + 0,1971 = 0,418 \quad (72)$$

Parametar vrijeme izvršavanja ima normaliziranu vrijednost od 0,316, dok utjecaj inferencijskog vremena izvršavanja na preciznost iznosi 0,657. S obzirom na to da težinski faktor α=0,7 daje veći značaj inferencijskom vremenu izvršavanja, prioritetizira se brzina u odnosu na očuvanje preciznosti predikcija. Analizirajući pojedinačne parametre funkcije cilja, vidljivo je da je inferencijsko vrijeme izvršavanja u srednjem rasponu normalizirane skale, što pokazuje kako razvijena metoda *Glasanja* nije među najbržima, ali se još uvijek nalazi u prihvatljivim granicama. Istovremeno, relativno visoka normalizirana vrijednost utjecaja inferencijskog vremena izvršavanja na preciznost (0,657) sugerira da brzina ima značajan negativan učinak na točnost odnosno preciznost predikcija. Prethodno navedeno, govori kako razvijena metoda *Glasanja*, unatoč optimizaciji brzine, nije u potpunosti otporna na probleme vezane uz degradaciju preciznosti uzrokovanu inferencijskim vremenom izvršavanja. Vraćena optimizacijska metrika odnosno vraćeni rezultat funkcije cilja od 0,418 pokazuje da razvijena metoda *Glasanja* postiže umjereno zadovoljavajući balans između brzine i preciznosti, ali s tendencijom prema kompromisu koji može rezultirati gubitkom točnosti zbog većeg utjecaja inferencijskog vremena izvršavanja.

Nadalje, prikazana je funkcija cilja za razvijenu metodu *Ruksaka* pri uzimanju 5 značajki odnosno komponenti u nižedimenzionalnom prostoru.

Rbr.	Metoda	Vrijeme izvršavanja (ms)
1.	Metoda Ruksaka	43,10
2.	Autoenkoderske neuronske mreže	56,92
3.	Model bez redukcije dimenzionalnosti	57,53
4.	t-SNE	58,86
5.	PCA	63,82
6.	UMAP	67,00

Tablica 33: Vrijeme izvršavanja metoda kod 5 značajki/komponenti

Parametri minimalnog i maksimalnog praga vremena izvršavanja za funkciju cilja, definirani su kao referentne vrijednosti za normalizaciju podataka, gdje se osigurava konzistentna usporedivost između različitih scenarija izvršavanja. Pritom za minimalni prag su 43,1 milisekundi (ms), dok su za maksimalni 67,00 milisekundi (ms).

Rbr.	Metoda	Točnost
1.	Autoenkoderske neuronske mreže	0,4498
2.	Metoda Ruksaka	0,3852
3.	Model bez redukcije dimenzionalnosti	0,3397
4.	PCA	0,3301
5.	UMAP	0,3182
6.	t-SNE	0,3158

Tablica 34: Točnost predikcije između metoda kod 5 značajki/komponenti

Parametri najmanjeg i najvećeg utjecaja na preciznost za funkciju cilja, definirani su kao referentne vrijednosti koje omogućuju kalibraciju modela u odnosu na raspon utjecaja na predikcijske performanse točnosti. Pritom za najmanji utjecaj su 0,449760766, dok su za najveći 0,315789474.

Funkcija cilja za razvijenu metodu *Ruksaka*, sastoji se od 3 procesne radnje, prikazane u nastavku:

1. Ulazni podatci:
    - Minimalni prag vremena izvršavanja – 43,10 ms
    - Maksimalni prag vremena izvršavanja – 67,00 ms
    - Najmanji utjecaj na preciznost - 0,449760766
    - Najveći utjecaj na preciznost - 0,315789474
    - Vrijeme izvršavanja razvijene metode Ruksaka - 43,10 ms
    - Predikcija točnosti/preciznosti razvijene metode Ruksaka - 0,385167464
2. Normalizacija podataka: svaka vrijednost se skalira u raspon od 0 do 1 kako bi imali istu važnost
    - Vrijeme izvršavanja

$$T_{norm} = \frac{43,10 - 43,10}{67,0 - 43,10} = \frac{0}{23,90} = 0,000 \tag{73}$$

    - Predikcija točnosti/preciznosti

$$I_{norm} = \frac{0,385167464 - 0,449760766}{0,315789474 - 0,449760766} = \frac{-0,064593302}{-0,133971292} = 0,482 \tag{74}$$

3. Funkcija cilja

$$F(43.10, 0.385167464) = 0,7 * 0,000 + (1 - 0,7) * 0,482 \\ = 0,000 + 0,1446 = 0,145 \quad (75)$$

Normalizirana vrijednost inferencijskog vremena izvršavanja iznosi 0,000, upućuje na to da razvijena metoda *Ruksaka* postiže najbolji mogući rezultat u odnosu na definirani raspon inferencijskog vremena izvršavanja, dok normalizirana vrijednost utjecaja razvijene metode *Ruksaka* na točnost/preciznost od 0,482 sugerira da razvijena metoda *Ruksaka* ima umjereno ograničen negativan učinak na predikcijske performanse točnosti, ali se ne nalazi u kritičnom rasponu degradacije predikcijske performanse točnosti. S obzirom na to da težinski faktor α=0,7 pridaje veću važnost minimizaciji inferencijskog vremena izvršavanja u odnosu na preciznost, ukupna vrijednost funkcije cilja 0,145 ukazuje na vrlo povoljan balans između dvaju kriterija – inferencijskog vremena izvršavanja i točnosti. Relativno niska vrijednost funkcije cilja sugerira da je razvijena metoda *Ruksaka* optimizirana tako da postiže iznimno nisko inferencijsko vrijeme izvršavanja, uz istovremeni umjereni kompromis u održavanju točnosti predikcija unutar prihvatljivih granica.

Skup podataka s cijenama nekretnina

U nastavku je prikazana funkcija cilja za razvijenu metodu *Glasanja* pri uzimanju 34 značajke odnosno komponenti u niždimenzionalnom prostoru.

Rbr.	Metoda	Vrijeme izvršavanja (ms)
1.	Autoenkoderske neuronske mreže	18,68
2.	Metoda Glasanja	26,02
3.	PCA	32,06
4.	Model bez redukcije dimenzionalnosti	172,68
5.	t-SNE	Nema rezultata
6.	UMAP	Nema rezultata

Tablica 35: Vrijeme izvršavanja metoda kod 34 značajke/komponente

Parametri minimalnog i maksimalnog praga vremena izvršavanja za funkciju cilja, definirani su kao referentne vrijednosti za normalizaciju podataka, gdje se osigurava konzistentna usporedivost između različitih scenarija izvršavanja. Pritom za minimalni prag su 18,68 milisekundi (ms), dok su za maksimalni 172,68 milisekundi (ms).

Rbr.	Metoda	RMSE
1.	Model bez redukcije dimenzionalnosti	41544,31
2.	Metoda Glasanja	42249,71
3.	PCA	56669,73
4.	Autoenkoderske neuronske mreže	59289,90
5.	t-SNE	Nema rezultata
6.	UMAP	Nema rezultata

Tablica 36: RMSE greška između metoda kod 34 značajke/komponente

Parametri najmanjeg i najvećeg utjecaja na preciznost za funkciju cilja, definirani su kao referentne vrijednosti koje omogućuju kalibraciju modela u odnosu na raspon utjecaja na predikcijske performanse RMSE greške. Pritom za najmanji utjecaj su 41544,31, dok su za najveći 59289,90.

Funkcija cilja za razvijenu metodu *Glasanja*, sastoji se od 3 procesne radnje, prikazane u nastavku:

1. Ulazni podatci:
    - Minimalni prag vremena izvršavanja – 18,68 ms
    - Maksimalni prag vremena izvršavanja – 172,68 ms
    - Najmanji utjecaj na preciznost - 41544,31
    - Najveći utjecaj na preciznost - 59289,90
    - Vrijeme izvršavanja razvijene metode Glasanja – 26,02 ms
    - Predikcija RMSE greške razvijene metode Glasanja - 42249,71
2. Normalizacija podataka: svaka vrijednost se skalira u raspon od 0 do 1 kako bi imali istu važnost
    - Vrijeme izvršavanja

$$T_{norm} = \frac{26,02 - 18,68}{172,68 - 18,68} = \frac{7,35}{154,00} = 0,048 \tag{76}$$

    - Predikcija RMSE greške

$$I_{norm} = \frac{42249,71 - 41544,31}{59289,90 - 41544,31} = \frac{705,39}{17745,59} = 0,040 \tag{77}$$

3. Funkcija cilja

$$F(26.02, 42249.71) = 0,7 * 0,048 + (1 - 0,7) * 0,040 \\ = 0,0336 + 0,012 = 0,046 \quad (78)$$

Parametar inferencijsko vrijeme izvršavanja ima normaliziranu vrijednost od 0,048, dok utjecaj inferencijskog vremena izvršavanja na preciznost iznosi 0,040. Obje vrijednosti su iznimno niske u odnosu na referentni raspon mogućih vrijednosti, što implicira visoko optimiziranu izvedbu razvijene metode *Glasanja* kako u pogledu brzine izvršavanja, tako i u pogledu minimalnog negativnog utjecaja latencije na predikcijske performanse. S obzirom na to da težinski faktor faktor α=0,7 daje veći značaj brzini izvršavanja, činjenica da je normalizirana vrijednost inferencijskog vremena izvršavanja gotovo na donjoj granici ukazuje na to da je razvijena metoda *Glasanja* iznimno efikasna u pogledu predikcije preciznosti odnosno niske RMSE greške sa svojim značajkama. Istovremeno, niska normalizirana vrijednost utjecaja latencije na RMSE grešku upućuje na to da latencija ne uzrokuje značajno povećanje RMSE greške, čime se dodatno potvrđuje kvaliteta razvijene metode *Glasanja*. Nastavno, vrijednost funkcije cilja od 0,046 ukazuje na izuzetno povoljan balans između brzine i niske RMSE greške, što znači da razvijena metoda *Glasanja*, ne samo da postiže iznimno nisko inferencijsko vrijeme izvršavanja odnosno brzinu, već istovremeno je očuvala nisku razinu RMSE greške.

Nadalje, prikazana je funkcija cilja za razvijenu metodu *Ruksaka* pri uzimanju 57 značajki odnosno komponenti u nižedimenzionalnom prostoru.

Rbr.	Metoda	Vrijeme izvršavanja (ms)
1.	Autoenkoderske neuronske mreže	27,93
2.	Metoda Ruksaka	49,94
3.	PCA	51,96
4.	Model bez redukcije dimenzionalnosti	172,68
5.	t-SNE	Nema rezultata
6.	UMAP	Nema rezultata

Tablica 37: Vrijeme izvršavanja metoda kod 57 značajki/komponenti

Parametri minimalnog i maksimalnog praga vremena izvršavanja za funkciju cilja, definirani su kao referentne vrijednosti za normalizaciju podataka, gdje se osigurava konzistentna usporedivost između različitih scenarija izvršavanja. Pritom, za minimalni prag su 27,93 milisekunde (ms), dok su za maksimalni 172,68 milisekundi (ms).

Rbr.	Metoda	RMSE
1.	Metoda Ruksaka	37933,15
2.	Model bez redukcije dimenzionalnosti	41544,31
3.	PCA	52298,32
4.	Autoenkoderske neuronske mreže	55919,27
5.	t-SNE	Nema rezultata
6.	UMAP	Nema rezultata

Tablica 38: Točnost predikcije između metoda kod 57 značajki/komponenti

Parametri najmanjeg i najvećeg utjecaja na preciznost za funkciju cilja, definirani su kao referentne vrijednosti koje omogućuju kalibraciju modela u odnosu na raspon utjecaja na predikcijske performanse RMSE greške. Pritom, za najmanji utjecaj su 37933,15, dok su za najveći 55919,27.

Funkcija cilja za razvijenu metodu *Ruksaka*, sastoji se od 3 procesne radnje, prikazane u nastavku:

1. Ulazni podatci:
    - Minimalni prag vremena izvršavanja – 27,93 ms
    - Maksimalni prag vremena izvršavanja – 172,68 ms
    - Najmanji utjecaj na preciznost - 37933,15
    - Najveći utjecaj na preciznost - 55919,27
    - Vrijeme izvršavanja razvijene metode Ruksaka - 49,94 ms
    - Predikcija RMSE greške razvijene metode Ruksaka - 37933,15
2. Normalizacija podataka: svaka vrijednost se skalira u raspon od 0 do 1 kako bi imali istu važnost
    - Vrijeme izvršavanja

$$T_{norm} = \frac{49,94 - 27,93}{172,68 - 27,93} = \frac{22,01}{144,75} = 0,152 \tag{79}$$

    - Predikcija RMSE greške

$$I_{norm} = \frac{37933,15 - 37933,15}{55919,27 - 37933,15} = \frac{0,00}{17986,12} = 0,000 \tag{80}$$

3. Funkcija cilja

$$F(49.94, 37933.15) = 0,7 * 0,152 + (1 - 0,7) * 0,000 \\ = 0,152 + 0,000 = 0,152 \quad (81)$$

Normalizirana vrijednost inferencijskog vremena izvršavanja iznosi 0,152, upućuje na to da se razvijena metoda *Ruksaka* postiže relativno brzo inferencijsko vrijeme izvršavanja u odnosu na definirani raspon inferencijskog vremena izvršavanja, dok normalizirana vrijednost utjecaja razvijene metode *Ruksaka* na RMSE grešku od 0,000 sugerira da razvijena metoda *Ruksaka* nema nikakav negativan učinak na RMSE grešku. S obzirom na to da težinski faktor α=0,7 pridaje veću važnost minimizaciji inferencijskog vremena izvršavanja u odnosu na RMSE grešku, ukupna vrijednost funkcije cilja 0,152 ukazuje na visoko učinkovito izvršavanje razvijene metode *Ruksaka*, iako nije najbrže moguće. Međutim, ključna prednost ovog rezultata razvijene metode *Ruksaka* proizlazi iz potpunog eliminiranja utjecaja latencije na RMSE grešku, što znači da razvijena metoda *Ruksaka* ostvaruje minimalnu moguću RMSE, bez povećanja RMSE greške uzrokovane bržim inferencijskim vremenom izvršavanja predikcije. Relativno niska vrijednost funkcije cilja naglašava na vrlo povoljan balans između brzine i RMSE greške, pri čemu je postignuto relativno nisko inferencijsko vrijeme izvršavanja uz istovremeno potpuno očuvanje niske RMSE greške.

Skup podataka s cjepivima

U nastavku je prikazana funkcija cilja za razvijenu metodu *Glasanja* i *Ruksaka* pri uzimanju 3 značajke/predmeta odnosno komponente u nižedimenzionalnom prostoru.

Rbr.	Metoda	Vrijeme izvršavanja (ms)
1.	UMAP	84,85
2.	Autoenkoderske neuronske mreže	123,19
3.	PCA	129,67
4.	Metoda Ruksaka	143,29
5.	Metoda Glasanja	143,29
6.	Model bez redukcije dimenzionalnosti	1220,54
7.	t-SNE	Nema rezultata

Tablica 39: Vrijeme izvršavanja metoda kod 3 značajke/predmeta/komponente

Parametri minimalnog i maksimalnog praga vremena izvršavanja za funkciju cilja, definirani su kao referentne vrijednosti za normalizaciju podataka, gdje se osigurava

konzistentna usporedivost između različitih scenarija izvršavanja. Pritom za minimalni prag su 84,85 milisekundi (ms), dok su za maksimalni 1220,54 milisekunde (ms).

Rbr.	Metoda	Točnost
1.	Metoda Ruksaka	0,3763
2.	Metoda Glasanja	0,3763
3.	Model bez redukcije dimenzionalnosti	0,3761
4.	PCA	0,2549
5.	Autoenkoderske neuronske mreže	0,2483
6.	UMAP	0,2483
7.	t-SNE	Nema rezultata

Tablica 40:Točnost predikcije između metoda kod 3 značajke/predmeta/komponente

Parametri najmanjeg i najvećeg utjecaja na preciznost za funkciju cilja, definirani su kao referentne vrijednosti koje omogućuju kalibraciju modela u odnosu na raspon utjecaja na predikcijske performanse točnosti. Pritom, za najmanji utjecaj su 0,376321839, dok su za najveći 0,248275862.

Funkcija cilja za razvijenu metodu *Glasanja* i *Ruksaka*, sastoji se od 3 procesne radnje, prikazane u nastavku:

1. Ulazni podatci:
    - Minimalni prag vremena izvršavanja – 84,85 ms
    - Maksimalni prag vremena izvršavanja – 1220,54 ms
    - Najmanji utjecaj na preciznost - 0,376321839
    - Najveći utjecaj na preciznost - 0,248275862
    - Vrijeme izvršavanja razvijene metode Glasanja i Ruksaka - 143,29 ms
    - Predikcija točnosti/preciznosti razvijene metode Glasanja i Ruksaka - 0,376321839
2. Normalizacija podataka: svaka vrijednost se skalira u raspon od 0 do 1 kako bi imali istu važnost
    - Vrijeme izvršavanja

$$T_{norm} = \frac{143,29 - 84,85}{1220,54 - 84,85} = \frac{58,44}{1135,69} = 0,051 \tag{82}$$

    - Predikcija točnosti/preciznosti

$$I_{norm} = \frac{0,376321839 - 0,376321839}{0,248275862 - 0,376321839} = \frac{0,000}{-0,128} = 0,000 \tag{83}$$

3. Funkcija cilja

$$\begin{aligned} F(143.29, 0.376321839) &= 0,7 * 0,051 + (1 - 0,7) * 0,000 \\ &= 0,0357 + 0,000 = 0,036 \end{aligned} \tag{84}$$

Parametar inferencijsko vrijeme izvršavanja ima normaliziranu vrijednost od 0,051, dok utjecaj inferencijskog vremena izvršavanja na preciznost iznosi 0,000. S obzirom na to da težinski faktor α=0,7 daje veći značaj inferencijskom vremenu izvršavanja, prioritetizira se brzina u odnosu na očuvanje preciznosti predikcija. Analizirajući pojedinačne parametre funkcije cilja, vidljivo je da je inferencijsko vrijeme izvršavanja u srednjem rasponu normalizirane skale, što pokazuje kako razvijene metode *Glasanja* i *Ruksaka* nisu među najbržima, ali niska normalizirana vrijednost inferencijskog vremena izvršavanja (0,051) ukazuje da razvijene metode *Glasanja* i *Ruksaka* postižu iznimno dobru učinkovitost u vremenskoj domeni, iako još uvijek postoji mogućnost daljnjeg ubrzanja prema minimumu u zadanom rasponu vremena svih metoda redukcije dimenzionalnosti. Istovremeno, rezultati ukazuju na to da razvijene metode *Glasanja* i *Ruksaka* nemaju nikakav negativan učinak na predikcijske performanse, što znači da su razvijene metode *Glasanja* i *Ruksaka* očuvale maksimalnu točnost predikcija, dok istovremeno postižu vrlo nisko inferencijsko vrijeme izvršavanja u odnosu na zadani raspon vrijednosti. Ključna prednost razvijenih metoda *Glasanja* i *Ruksaka* je u činjenici da u potpunosti eliminiraju negativan utjecaj na preciznost, čime se postiže idealna predikcijska točnost bez narušavanja performansi. Vrijednost funkcije cilja od 0,036 ukazuje na izuzetno povoljan balans između brzine i preciznosti, pri čemu razvijene metode *Glasanja* i *Ruksaka* uspješno minimiziraju inferencijsko vrijeme izvršavanja, dok istovremeno osiguravaju maksimalnu predikcijsku točnost. Rezultat funkcije cilja sugerira vrlo visoku razinu robusnosti razvijenih metoda *Glasanja* i *Ruksaka*, gdje su oba kriterija – brzina i točnost - ostvarena u gotovo idealnim uvjetima.

6.4. Parni t-test i bootstrap metoda

Potpoglavlje parni t-test[89] i bootstrap metoda[90, 91] predstavlja rezultate statističke značajnosti prve i druge hipoteze ovog doktorskog rada na temelju provedenog

parnog t-testa prije i poslije primjene razvijenih metoda - *Glasanja* i *Ruksaka* te zasebno bootstrap metode[90, 91] za prvu hipotezu.

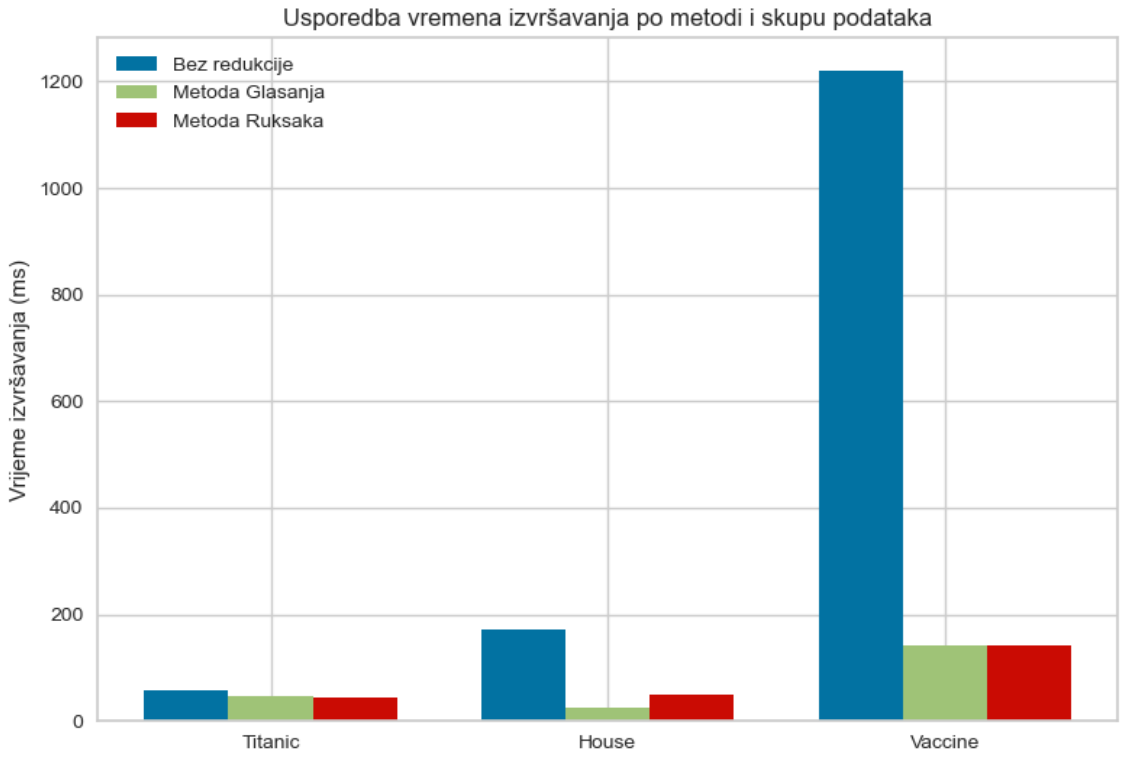
Prva hipoteza (H1) polazi od pretpostavke da je razvijena prilagođena verzija CRISP-DM metodološkog okvira s ugrađenim algoritmom latencije statistički značajno brža u izvršavanju predikcija za najmanje 15% u odnosu na verziju bez prilagodbe. Parni t-test[89] za usporedbu metode *Glasanja* s baznom metodom/modelom bez redukcije dimenzionalnosti pokazuje sljedeće rezultate: t=1,2273 i p=0,3446, dok za usporedbu metode *Ruksaka* s baznom metodom/modelom bez redukcije dimenzionalnosti pokazuje rezultate: t=1,1988 i p=0,3534. S obzirom na to da su obje p-vrijednosti znatno veće od standardne razine značajnosti (0,05), ne odbacuje se hipoteza prema kojoj razlike u prosječnim vremenima izvršavanja nisu statistički značajne. Vizualizacija rezultata, na slici u nastavku, s deskriptivnom razlikom u vremenu jasno pokazuje smanjenje vremena izvršavanja odnosno brže izvršavanje, ali statistička značajnost parnog t-testa na danom skupnom uzorku odabranih skupova podataka (tri promatrana skupa podataka) nije dovoljna da potvrdi H1 s visokom pouzdanošću.

Nepostojanje potvrde statističke značajnosti H1 je očekivano jer H1 se ne odnosi na bilo kakvu razliku među srednjim vrijednostima, već na postojanje minimalnog, unaprijed definiranog praga poboljšanja od najmanje 15 % u inferencijskom vremenu izvršavanja. Prethodno navedeno, standardni t-test (metoda *ttest_rel*[92]) ne podržava jer ispituje postoji li statistički značajna razlika između srednjih vrijednosti dvaju povezanih uzoraka, odnosno između inferencijskog vremena izvršavanja prije i nakon primjene razvijenih metoda *Glasanja* i *Ruksaka*. U tom slučaju hipoteze H1, t-test mjeri statističku značajnost opće razlike između uzoraka (bazne metode/modela bez redukcije dimenzionalnosti i metoda *Glasanja* i *Ruksaka*), ali ne mjeri izravno ostvaruje li se poboljšanje koje prelazi postavljeni prag od 15%.[89]

Stoga se za provjeru H1 primijenila bootstrap metoda za izračun intervala povjerenja koja je omogućila izračun stvarnog postotnog smanjenja inferencijskog vremena te kvantifikaciju povjerenja u taj rezultat putem 95% intervala povjerenja (95% CI). Na taj način dobiva se ne samo procjena srednje vrijednosti poboljšanja, već i mjera sigurnosti te procjene. Drugim riječima, statistička granica unutar koje se s određenom pouzdanošću očekuje stvarna vrijednost smanjenja inferencijskog vremena

izvršavanja. Ako donja granica intervala pouzdanosti prelazi postavljeni prag od -15%, H1 se smatra potvrđenom, jer se time pokazuje da je postignuto poboljšanje statistički značajno i veće od minimalno očekivanog (-15%).[90, 91]

Rezultati bootstrap metode pokazali su da je srednja postotna promjena inferencijskog vremena izvršavanja iznosila -57,47%, što jasno ukazuje da su metode *Glasanja* i *Ruksaka* u prosjeku gotovo dvostruko brže od bazne metode/modela bez redukcije dimenzionalnosti. Nadalje, donja granica 95 % intervala povjerenja iznosila je -61,66%, dok je gornja granica iznosila -54,03%. Time je cijeli interval povjerenja u cijelosti ispod postavljenog praga od -15%, čime se potvrđuje statistička značajnost hipoteze H1.

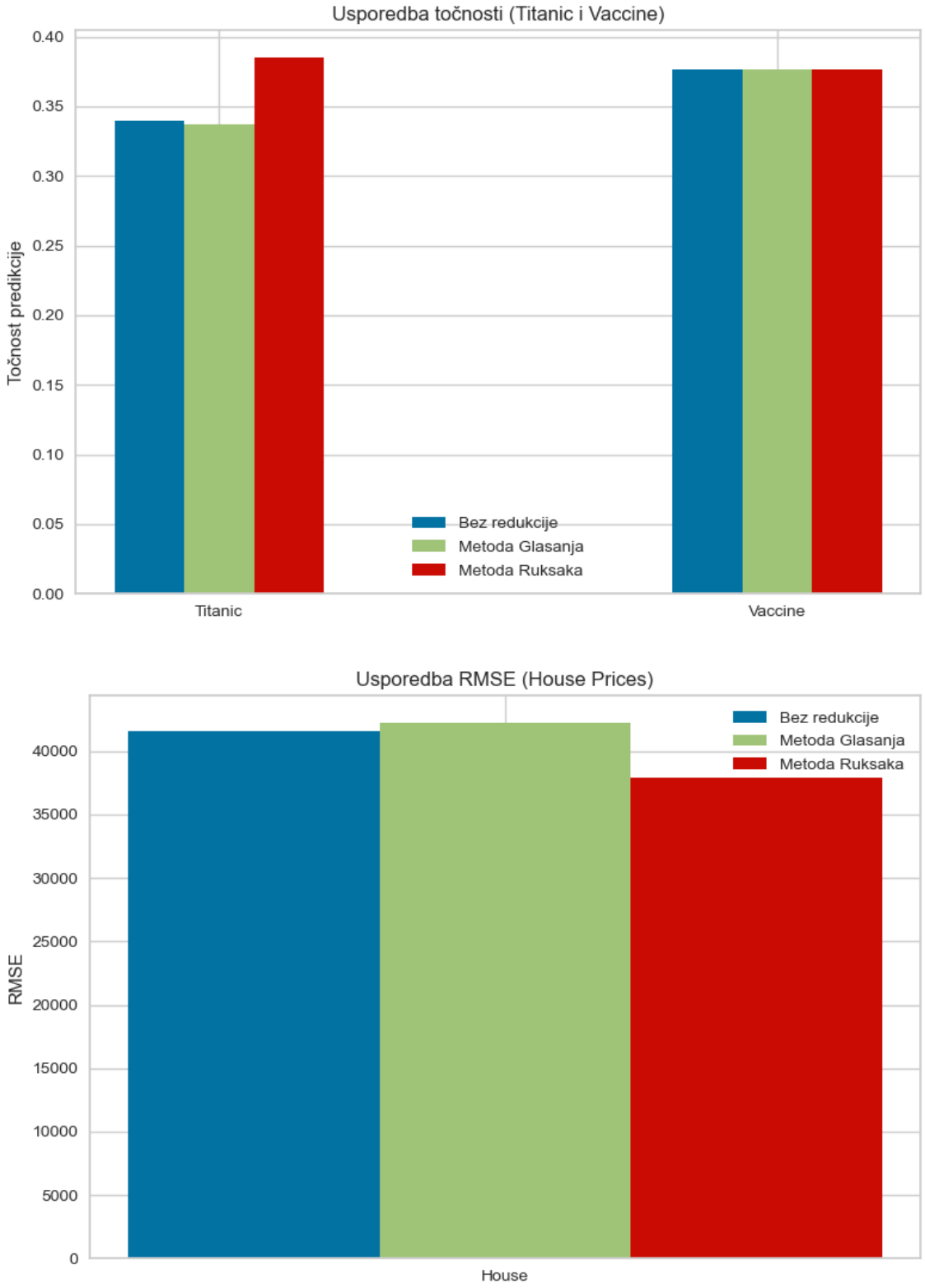


Slika 43: Usporedba vremena izvršavanja po metodi i skupu podataka

Na slici iznad uočava se velika apsolutna razlika u vremenu izvršavanja u korist razvijenih metoda *Glasanja* i *Ruksaka*, osobito kod House Prices i Vaccine skupa podataka.

Druga hipoteza (H2) polazi od pretpostavke da uvođenje algoritma latencije ne dovodi do smanjenja točnosti većem od 4%. Usporedni rezultati parnog t-testa za metodu *Glasanja* s baznom metodom/modelom bez redukcije dimenzionalnosti su t=−1,00, i

p=0,4227 te metodu *Ruksaka* s baznom metodom/modelom bez redukcije dimenzionalnosti su t=1,00, i p=0,4227. Ni u jednom slučaju ne postoji statistički značajna razlika u točnosti kod Titanic, Vaccine ili RMSE-a kod House Prices skupa podataka. Činjenica da su p-vrijednosti i ovdje znatno veće od 0,05 podupire prihvaćanje hipoteze H2, odnosno potvrđuje da nema statistički značajnog smanjenja točnosti predikcija nakon primjene razvijenih metoda *Glasanja* i *Ruksaka* u CRISP-DM okviru. Vizualizacija rezultata dodatno potvrđuje rezultate parnog t-testa gdje grafikoni, na slici u nastavku, s prikazom točnosti ili RMSE-a pokazuju vrlo slične vrijednosti između svih metoda, uz minimalne oscilacije koje parni t-test ne prepoznaje kao značajne.



Slika 44: Usporedba točnosti i RMSE-a

Hipoteza H1 nije potvrđena parnim t-testom na danom uzorku zbog velike varijance među testiranim skupovima podataka i ograničenja t-testa u mjerenju

poboljšanja od 15%, statistička značajnost potvrđena je primjenom bootstrap metode, koja je omogućila mjerenje postotne promjene (-15%) i pouzdaniji interval povjerenja. S druge strane, H2 potvrđuje, s parnim t-testom, rezultate razvijenih metoda *Glasanja* i *Ruksaka*, čime se dokazuje da algoritam latencije u CRISP-DM okviru ne umanjuje predikcijsku učinkovitost, što je od važnog značaja za primjenu u različitim poslovnim domenama produkcijskih okružja.

6.5. Doprinosi istraživanja

Većina prethodnih istraživanja, uključujući radove autora Li, et al. [39] (VolcanoML), Gosiewska, et al. [40] (SAFE), Yakovlev, et al. [41] (Oracle AutoML), Fister, et al. [42] (NiaAML), EldeebiElshawi [43] (BigFeat), Kedziora, et al. [44] (meta-učenje) i Wu, et al. [45] (ChaCha algoritam), usmjerena su na rješavanje različitih izazova unutar AutoML sustava: skalabilnost pretraživačkih prostora, interpretabilnost modela, robusnost prema adverzijalnim napadima, računalna učinkovitost i meta-učenje. Međutim, u većini analiziranih radova izostaje integrirana metodološka povezanost između faza pripreme podataka, redukcije dimenzionalnosti i optimizacije modela s obzirom na latencijske zahtjeve i vremensku složenost izvođenja modela.

U tom kontekstu, istraživanje ovog doktorskog rada i razvoj sustava *MoziaisMl* donosi novi pristup i znanstveni doprinos kroz implementaciju prilagođene verzije CRISP-DM metodološkog okvira, u kojoj ne samo da se automatizira cijeli cjevovod strojnog učenja, već se i primjenjuju razvijene metode redukcije dimenzionalnosti (*Glasanja* i *Ruksaka*), koje nadilaze problem ekspanzije značajki identificiran kod autora EldeebiElshawi [43]. Metode *Glasanja* i *Ruksaka* omogućuju automatizirani i dinamički odabir značajki uz očuvanje interpretabilnosti i pouzdanosti predikcije, čime se doprinosi transparentnosti, robusnosti i generalizacijskoj sposobnosti s *MoziaisMl* sustavom.

Također, rad autora Kedziora, et al. [44] naglašava izostanak integracije vremenskih aspekata pri evaluaciji modela, *MoziaisMl* sustav ima integriran algoritam latencije s primarnom komponentom vremenskog aspekta unutar višekriterijske optimizacije – inferencijsko vrijeme kao prvi kriterij te zatim točnost/greška kao drugi kriterij. Time se omogućuje procjena metodoloških okvira koje uvažavaju ne samo točnost modela, već i

inferencijsko vrijeme izvođenja predikcije kao bitan čimbenik u produkcijskim sustavima.

Zaključno, implementacija prilagodljivih mehanizama automatizirane detekcije neuravnoteženosti klasa, razdvajanje kompleksnih tekstualnih atributa te utvrđivanje i obrada nedostajućim i dupliciranim vrijednostima u fazi pretprocesiranja osiguravaju robusnost i prilagodljivost AutoML sustava različitim poslovnim domenama, što u radovima autora poput Yakovlev, et al. [41] (Oracle AutoML) i EldeebiElshawi [43] (BigFeat) nije sustavno razrađeno.

6.5.1. Implikacije za praksu

MoziaisMl sustav izdvaja se po tome što pruža replikabilno i vremenski optimizirano rješenje za automatizirani dizajn modela strojnog učenja u stvarnim produkcijskim okružjima. Razvijene metode redukcije dimenzionalnosti *Glasanja* i *Ruksaka* omogućuju značajno smanjenje složenosti modela bez gubitka točnosti, čime se olakšava njihovo izvođenje u uvjetima ograničenih računalnih resursa.

Korištenjem algoritma latencije unutar višekriterijske evaluacije, *MoziaisMl* sustav omogućuje optimizaciju kompromisa između brzine izvođenja i prediktivne učinkovitosti, čime se unaprjeđuje ravnoteža između predikcije i učinkovitog operativnog upravljanja infrastrukturnim resursima.

Nastavno, prilagođeni CRISP-DM okvir dokumentira svaki korak analitičkog procesa, što pridonosi auditabilnosti i transparentnosti rada *MoziaisMl* sustava.

Pritom, provedeno istraživanje i razvijeni *MoziaisMl* sustav ne samo da popunjava prethodno utvrđene praznine u istraživanju iz relevantne znanstvene literature, već i nudi prilagođene metodološke faze i programski kod automatiziranog i generalizacijski robusnog procesa temeljenog na metodama strojnog učenja unutar ovog doktorskog rada.

U praktičnom smislu, u okviru digitalne transformacije poslovanja banke, posebice u odjelima zaduženim za podršku poslovanju, *MoziaisMl* sustav može nadograditi informacijske sustave uredskog poslovanja i arhiviranja automatizacijom analitičkih procesa nad dokumentima koji ulaze u takve sustave.

Nakon digitalizacije dokumenata (primjer, zahtjevi za kredit, izvadci, ovjerene isprave) tokom koje se izvrši strojna obrada teksta i ekstrakcija značajki u CSV, *MoziaisMl* automatski identificira tip dokumenta, izdvaja ključne značajke (primjer, ime klijenta, OIB, iznos, datum, vrstu kredita) te ih pretprocesira u strukturirani oblik prikladan za daljnju analizu.

Potom se pokreće prilagođena treća faza modeliranja i optimizacije modela, gdje *MoziaisMl* automatski trenira više klasifikacijskih modela, optimizira njihove parametre te odabire najbolji klasifikacijski model s pripadajućim parametrima.

Zatim u sljedećoj fazi, evaluira performanse prema višekriterijskoj funkciji cilja - latencija i točnost. U praksi, to znači da se dokumenti klasificiraju s visokom točnošću u kategorije kao što su Zahtjev za kredit, Ugovor o kreditu, Izjava jamca, Dodatna dokumentacija, itd., pri čemu se zadržava vrijeme predikcije ispod određenog praga (primjer, 80ms), što je ključno za integraciju u produkcijsko okružje.

Nadalje, *MoziaisMl* automatski primjenjuje vlastite metode redukcije dimenzionalnosti - metoda *Glasanja* i *Ruksaka* - kako bi se smanjila složenost ulaznih značajki, čime se ubrzava predikcija modela i povećava stabilnost bez gubitka točnosti. Ovo je osobito važno u scenarijima gdje stotine OCR-om (engl. *Optical Character Recognition*) dobivenih značajki treba svesti na najrelevantnije za donošenje klasifikacijske odluke.

MoziaisMl putem API-a može se integrirati s arhivskim informacijskim sustavom te automatski predložiti klasifikacijsku oznaku i poslovnu funkciju dokumenta, čime se smanjuje potreba za ručnim razvrstavanjem od strane službenika. Nakon završene klasifikacije i dodjele poslovne funkcije, *MoziaisMl* može omogućiti da dokumenti automatski budu uvršteni u odgovarajuće predmete, uz pridružene metapodatke i klasifikaciju, a svi rezultati (vrijeme obrade, tip modela, evaluacija) se pohranjuju za auditorsku i regulatornu sljedivost.

Konkretno, *MoziaisMl* može omogućiti automatiziranu klasifikaciju dokumentacije s visokom točnošću (primjer, iznad 95%), inferencijsko vrijeme predikcije smanjiti (primjer, ispod 80ms), čime je povećana latencijska učinkovitost, dok službenici samo identificiraju nedostajuće ili nepravilno klasificirane dokumente, što značajno ubrzava obradu dokumentacije i smanjuje operativni rizik banke. Time, implementacija

MoziaisMl-a u bankarski dokumentacijski i arhivski sustav rezultira povećanjem točnosti, brzinom obrade i smanjenjem operativnih troškova, ali i omogućava standardizirano, zakonski usklađeno i revizijski provjerljivo upravljanje dokumentima, što je ključno u sektorima podložnim strogim regulatornim zahtjevima poput bankarstva.

Od ostalih praktičnih primjena, može se izdvojiti automatizirana analiza podataka o prodaji, zaliha i ponašanju kupaca, pri čemu značajke poput vrste artikala, dana u tjednu i lokacije trgovine (u samom gradu i državi) *MoziaisMl* može automatski pretprocesirati, odabrati najrelevantnije - metodom Glasanja ili Ruksaka - te zatim generira predikcije – sve bez potrebe za intervencijom stručnjaka u području podatkovne analize. Time se može značajno skratiti vrijeme potrebno za donošenje odluka o optimizaciji zaliha na skladištu, što ima direktnu korelaciju s nižim operativnim troškovima. Zatim, u zdravstvu, *MoziaisMl* može omogućiti automatsku izgradnju modela za predikciju zdravstvenih rizika temeljenih na povijesti liječenja pacijenata. Budući da *MoziaisMl* automatski detektira, čisti i transformira podatke, može pomoći liječničkom osoblju ubrzati proces donošenja odluka.

Zaključno, određenije i opsežnije implikacije, u kontekstu građevinarstva, maloprodajne industrije, zdravstva ili drugih sektora, se mogu potvrditi nakon validacije rezultata vanjskih izvora nestrukturiranih podataka koje stručnjaci iz različitih poslovnih domena svakodnevno analiziraju.

6.6. Ograničenja i preporuke za buduća istraživanja

Provedeno istraživanje u ovom radu predstavlja prilagođeni CRISP-DM metodološki okvir u kojem su, kroz automatizirani algoritam latencije, razvijene metode pretprocesiranja značajki odnosno metode inženjerstva značajki te metode za redukciju dimenzionalnosti značajki.

Provedeno istraživanje predstavlja znanstveni doprinos u okviru automatiziranog dizajna modela strojnog učenja, s posebnim naglaskom na unaprjeđenje faza unutar prilagođenog CRISP-DM metodološkog okvira. U fazi Pripreme podataka, implementirane su metode pretprocesiranja koje uključuju sustavno prepoznavanje i označavanje nedostajućih vrijednosti - *NA, N/A, NaN, NULL*, prazne vrijednosti. Umjesto

konvencionalne imputacije nedostajućih vrijednosti u skupovima podataka, predloženo je generiranje dodatnih indikatorskih varijabli koje bilježe prisutnost praznina kroz značajke i zapise skupova podataka, čime se dodatna informacija zadržava u podatcima *MoziaisMl* sustava. Nadalje, razvijen je postupak za automatsku detekciju i procesiranje dupliciranih vrijednosti, kao i metoda za razdvajanje kompleksnih tekstualnih atributa kroz dinamičko generiranje novih interpretabilnih značajki, čime se povećava informacijska vrijednost značajki skupova podataka.

U okviru iste faze, značajan doprinos ostvaren je uvođenjem vlastitih metoda redukcije dimenzionalnosti, konkretno metode *Glasanja* i *Ruksaka*. Metoda *Glasanja* se temelji na agregaciji rang listi važnosti značajki iz više modela (*RandomForest[25, 26]*, *XGBoost[27]*, *Mutual Information[28, 29]* i *Recursive feature elimination[30]*), dok metoda *Ruksaka* koristi optimizacijski pristup baziran na problemu ruksaka za selekciju značajki s najvećim informativnim doprinosom uzimajući u obzir njihovu varijancu kao oblik troška. Navedene metode omogućuju reduciranje dimenzionalnosti bez značajnog gubitka prediktivne sposobnosti, a istovremeno doprinose smanjenju inferencijskog vremena izvođenja modela, što je od ključne važnosti za produkcijska okružja s visokim zahtjevima za latencijsku optimizaciju.

U fazi Modeliranja i optimizacije, znanstveni doprinos očituje se u detekciji neuravnoteženosti klasa, čime se osigurava povećanje osjetljivosti modela na manje zastupljene klase. Time se poboljšava evaluacija i smanjuje pristranost modela, a ujedno i doprinosi robusnosti i generalizacijskoj sposobnosti *MoziaisMl* sustava.

Završno, u fazi Implementacije, analiziraju se generalizacijske sposobnosti *MoziaisMl* sustava, gdje rezultati potvrđuju metodološku održivost i znanstvenu značajnost predloženih metoda redukcije dimenzionalnosti. Validacijom na različitim skupovima podataka demonstrirano je da metode *Glasanja* i *Ruksaka* osiguravaju stabilne i kompetitivne performanse klasifikacijskih i regresijskih modela, uz očuvanje visoke točnosti i značajno kraće inferencijsko vrijeme izvođenja. Time je potvrđena njihova primjenjivost u različitim poslovnim domenama, kao i njihova održivost u znanstvenim istraživanjima koja zahtijevaju replikabilnost, skalabilnost, robusnost, generalizacijsku sposobnost i vremensku efikasnost.

Iako provedeno istraživanje i ostvareni rezultati predstavljaju doprinos s prilagođenom verzijom CRISP-DM metodološkog okvira i razvojem *MoziaisMl* sustava – trenutne verzije 1.0, istraživanje ima određena ograničenja. Trenutna verzija prilagođenog CRISP-DM metodološkog okvira i *MoziaisMl* sustava služi kao temelj daljnjih istraživanja, a budući razvoj trebao bi se proširiti ne samo na rješavanje klasifikacijskih i regresijskih zadataka već i na modeliranje vremenskih serija.

Uvođenjem vremenskih serija omogućila bi se dodatna evaluacija učinkovitosti razvijenih metoda pretprocesiranja podataka i inženjerstva značajki, kao i razvijenih metoda redukcije dimenzionalnosti – *Glasanja* i *Ruksaka*. Poseban naglasak bio bi na analizi mogućnosti transformacije vremenskih podataka u niždimenzionalni prostor, uz optimizaciju inferencijskog vremena izvršavanja odnosno brzinu, gdje bi se očuvala prediktivna preciznost modela.

Također, treba naglasiti da su svi korišteni skupovi podataka preuzeti s platforme *Kaggle*, što može utjecati na validaciju rezultata vanjskih izvora nestrukturiranih podataka domenskih stručnjaka. U budućim istraživanjima planira se proširenje eksperimenta na dodatne izvore podataka koji obuhvaćaju domenski nestrukturirane ili vremenski varijantne podataka, kako bi se dodatno potvrdila robusnost i primjenjivost razvijenih metodoloških pristupa te generalizacijska sposobnost razvijenog *MoziaisMl* sustava.

Ograničenje istraživanja očituje se i u tome što uz evaluacijske metrike nije uvedena analiza potrošnje računalnih resursa - CPU (engl. *Central Processing Unit*) /GPU (engl. *Graphics Processing Unit*) iskoristivost te zauzeće RAM (engl. *Random-Access Memory*) memorije - za svaku metodu, čime se u istraživanju propustilo kvantificirati oportunitetni trošak izvođenja pojedinih metoda. U budućim istraživanjima planira se integracija ove analitičke metrike kako bi se omogućila cjelovitija procjena učinkovitosti razvijenog *MoziaisMl* sustava u odnosu na resursnu potrošnju računala.

Na kraju, buduće verzije prilagođenog CRISP-DM metodološkog okvira i *MoziaisMl* sustava mogle bi uključivati adaptivne strategije selekcije značajki s ciljem optimizacije preciznosti modela. Na temelju unaprijed definiranih pragova prediktivne točnosti, sustav bi mogao automatizirano određivati, najmanji mogući, minimalan broj značajki u niždimenzionalnom prostoru koji zadovoljava zadani prag performansi. Ovaj

pristup omogućio bi dinamičku prilagodbu modela, gdje bi se balans između predikcijske preciznosti i brzine izvršavanja optimizirao sukladno zahtjevima specifičnih zadataka klasifikacije, regresije i vremenskih serija.

7. ZAKLJUČAK

U ovom poglavlju sažimaju se zaključci o navedenom cilju i svrsi provedenog istraživanja ovog rada te rezultatima razvijenih metoda za pretprocesiranje podataka i razvijenih metoda redukcije dimenzionalnosti kroz prilagođenu verziju CRISP-DM metodološkog okvira i doprinosima doktorskog rada. Na kraju, prikazane su implikacije za praksu.

Provedeno istraživanje usmjereno je na razvoj i evaluaciju automatiziranog analitičkog informacijskog sustava temeljenog na strojnome učenju – *MoziaisMl* – koji se temelji na prilagođenoj verziji CRISP-DM metodološkog okvira. Glavni cilj bio je razviti algoritam latencije koji omogućava optimizaciju inferencijske vremenske učinkovitosti modela bez narušavanja predikcijske točnosti, pri čemu se istraživanjem doktorskog rada nastojao povećati stupanj automatizacije uz istovremeno smanjenje ljudske intervencije u fazama pripreme, modeliranja i optimizacije modela strojnog učenja. Sustav *MoziaisMl* uključuje razvijene metode za automatizirano pretprocesiranje podataka, kao i razvijene metode za redukciju dimenzionalnosti – *Glasanja* i *Ruksaka*.

Ciljevi rada realizirani su kroz razvoj algoritma latencije za produkcijsku primjenu unutar prilagođene verzije CRISP-DM metodološkog okvira. Algoritam latencije s metodama i slijednim procesnim koracima analizira i reducira dimenzionalnost ulaznog skupa podataka kako bi se dobio skup podataka temeljem kojeg metode strojnog učenja rade brže uz neznatne gubitke u točnosti ili greški predviđanja, čime su realizirani ciljevi rada.

Prilagođena verzija CRISP-DM metodološkog okvira je razvijena i unutar nje je razvijen algoritam latencije kao automatizirani cjevovod *MoziaisMl* sustava. Sustav *MoziaisMl* sastoji se od sljedećih, razvijenih, metoda za pretprocesiranje podataka: metode za utvrđivanje *NA, N/A, NaN, NULL* i praznih vrijednosti, metode za obradu *NA, N/A, NaN, NULL* i praznih vrijednosti kroz kreiranje novih stupaca koji označavaju da u nekim stupcima i redovima u skupu podataka ne postoje vrijednosti, metode za detekciju dupliciranih vrijednosti i metode za razdvajanje tekstualnih vrijednosti kroz kreiranje novih stupaca koji sadrže razdvojene tekstualne vrijednosti.

Osim razvijenih metoda za pretprocesiranje podataka, algoritam latencije odnosno sustav *MoziaisMl* sadrži vlastito razvijene metode za redukciju dimenzionalnosti:

Glasanja i *Ruksaka*. Razvijene metode *Glasanja* i *Ruksaka* reduciraju ulazne skupove podataka u nižedimenzionalne prostore s manje značajki pri tom eliminiraju redundantne informacije kako bi postigle niže inferencijsko vrijeme izvršavanja odnosno brzinu, uz istovremeno očuvanje relevantnih značajki za predikciju preciznosti.

Učinak razvijenih metoda *Glasanja* i *Ruksaka* kroz analizu podataka u poglavlju rezultati, potvrđen je funkcijom cilja. Najvažniji, u funkciji cilja – prvi parametar po rangu – je bilo inferencijsko vrijeme izvršavanja, a drugi parametar – utjecaj primjene algoritma latencije na točnost metrika predviđanja. U funkciji cilja, manja vrijednost funkcije F(T,I) označavala je bolji balans između brzine i točnosti. Funkcija cilja imala je parametar α za prilagođavanje težine između prioriteta brzine (α bliže 1) i preciznosti (α bliže 0). Obzirom kako je bio prvi po rangu, kriterij, inferencijsko vrijeme izvršavanja odnosno brzina, parametar alpha (α) je postavljen na 0,7 ($\alpha = 0.7$), s čime se ističe prvi rang.

Vrijednosti funkcije cilja, nad zadatcima binarne klasifikacije i regresije, za vlastito razvijenu metodu *Glasanja* su bile - 0,418 gdje se postigao umjereno zadovoljavajući balans između brzine i preciznosti, ali s tendencijom prema kompromisu koji može rezultirati gubitkom točnosti zbog većeg utjecaja inferencijskog vremena izvršavanja te 0,046 što je ukazivalo na izuzetno povoljan balans između brzine i niske RMSE greške kod regresijskog zadatka, gdje razvijena metoda *Glasanja*, ne samo postigla iznimno nisko inferencijsko vrijeme izvršavanja odnosno brzinu, već istovremeno je očuvala nisku razinu RMSE greške.

Vrijednosti funkcije cilja, nad zadatcima binarne klasifikacije i regresije, za vlastito razvijenu metodu *Ruksaka* su bile 0,145, što ukazuje na vrlo povoljan balans između dvaju kriterija – inferencijskog vremena izvršavanja i točnosti. Relativno niska vrijednost funkcije cilja sugerira da je razvijena metoda *Ruksaka* optimizirana tako da postiže iznimno nisko inferencijsko vrijeme izvršavanja, uz istovremeni umjereni kompromis u održavanju točnosti predikcija unutar prihvatljivih granica te 0,152, što ukazuje na visoko učinkovito izvršavanje razvijene metode *Ruksaka*, gdje razvijena metoda *Ruksaka* ostvaruje minimalnu moguću RMSE grešku, bez povećanja RMSE greške uzrokovane bržim inferencijskim vremenom izvršavanja predikcije. Relativno

niska vrijednost funkcije cilja naglašava na vrlo povoljan balans između brzine i RMSE greške.

Na zadnjem višeklasifikacijskom zadatku, vrijednost funkcije cilja za razvijene metode *Glasanja* i *Ruksaka* je bila 0,036, što ukazuje na izuzetno povoljan balans između brzine i preciznosti, pri čemu su se razvijene metode *Glasanja* i *Ruksaka* dokazale da uspješno minimiziraju inferencijsko vrijeme izvršavanja, dok istovremeno osiguravaju maksimalnu predikcijsku točnost. Rezultat funkcije cilja, sugerira vrlo visoku razinu robusnosti razvijenih metoda *Glasanja* i *Ruksaka*, gdje su oba kriterija – brzina i točnost - ostvarena u gotovo idealnim uvjetima.

Usporedna analiza učinkovitosti razvijenih metoda *Glasanja* i *Ruksaka* s popularnim tehnikama odnosno metodama redukcije dimenzionalnosti kod analize generalizacijskih sposobnosti ukazuje kako kod razvijenih metoda *Glasanja* i *Ruksaka* postoji povoljan balans između brzine i preciznosti jer metode PCA, t-SNE, UMAP i autoenkoderske neuronske mreže su imale zadovoljavajuću brzinu, ali uz veliki gubitak točnosti zbog većeg utjecaja nižeg inferencijskog vremena izvršavanja, a u binarnom klasifikacijskom zadatku su autoenkoderske neuronske mreže imale višu stopu točnosti od razvijenih metoda *Glasanja* i *Ruksaka*, uz istovremeno značajan gubitak u brzini odnosno autoenkoderske neuronske mreže su imale sporije inferencijsko vrijeme izvršavanja predikcije.

Za ispitivanje hipoteza primijenjen je parni t-test, što je opravdan izbor za usporedbu metrika (točnosti i inferencijskog vremena izvođenja) prije i poslije primjene razvijenih metoda *Glasanja* i *Ruksaka* na istim podatcima. Ispitivanje H1 pokazalo je deskriptivna poboljšanja u brzini izvođenja, ali bez statistički značajnog efekta prema zadanoj razini značajnosti ($p=0.05$), zbog čega H1 nije potvrđena parnim t-testom, iako može se smatrati potvrđenom u okviru postavljenog praga smanjenja inferencijskog vremena (-15%), unatoč statističkoj nepouzdanosti parnog t-testa na danom uzorku tri skupa podataka, obzirom kako je inferencijsko vrijeme bilo niže za -19,78%, -28,68%, -147,61%, -110,27% i -157,97%. Slijedom statističke nepouzdanosti parnog t-testa, korištena je bootstrap metoda za provjeru H1 hipoteze. Bootstrap metoda potvrdila je statistički značajnu razliku u srednjoj relativnoj promjeni inferencijskog vremena

izvršavanja te potkrijepljena je 95% intervalom povjerenja koji je u cijelosti ispod postavljenog praga H1 hipoteze (-15%).

S druge strane, ispitivanje H2 pokazalo je da razlike u točnosti predikcija između razvijenih metoda i bazne metode/modela nisu statistički značajne, a dobivene p-vrijednosti ukazuju na to da nema sustavne degradacije predikcijske učinkovitosti nakon primjene obje razvijene metode - *Glasanja* i *Ruksaka*. Stoga, rezultati parnog t-testa potvrđuju H2, odnosno upućuju na to da proširenje CRISP-DM metodologije algoritmom latencije za optimizaciju izvedbe ne narušava metrike predikcijske preciznosti ni u klasifikacijskim, a ni u regresijskim zadatcima.

Nastavno, dobiveni rezultati podupiru tvrdnju da prilagođena verzija CRISP-DM metodologije može značajno unaprijediti inferencijsku vremensku učinkovitost, bez negativnog utjecaja na predikcijske performanse. Stoga, zaključuje se kako su postavljene hipoteze potvrđene. Također, najveći gubitak točnosti je bio -0,71% odnosno povećanje greške za +1,68%. Ostali rezultati su povećali točnost za 12,54% i 0,06% te smanjili grešku za 9,08%. Potvrda postavljenih hipoteza i dobiveni rezultati osiguravaju metodološku valjanost, robusnost te generalizacijsku sposobnost predloženog prilagođenog okvira u primjeni u *MoziaisMl* sustavu u produkcijskim okružjima.

Zaključno, eksperimentalni rezultati podupiru tvrdnju da prilagođeni CRISP-DM metodološki okvir omogućava vremensku optimizaciju uz zadržavanje predikcijske preciznosti.

Istraživanje donosi znanstveni doprinos u području *AutoML* sustava kroz modularno unapređenje postojeće CRISP-DM metodologije, uključujući razvoj algoritma latencije, kao i dviju metoda za redukciju dimenzionalnosti - *Glasanja* i *Ruksaka*. Sustav *MoziaisMl* integrira automatizirane mehanizme za detekciju problema u podatcima, a kroz metode *Glasanja* i *Ruksaka* nudi balans između interpretabilnosti modela, računalne učinkovitosti i preciznosti predikcije. Uspješna primjena metoda potvrđena je eksperimentalnim validacijama koje ukazuju na njihovu stabilnost, robusnost i mogućnost generalizacije na različite tipove klasifikacijskih i regresijskih zadataka.

Rezultati istraživanja imaju izravne implikacije za praksu, posebno u kontekstu implementacije analitičkih informacijskih sustava u poslovna produkcijska okružja gdje su prioriteti niska latencija, minimalna ljudska intervencija, visok stupanj automatizacije

te gdje marginalna odstupanja od jedan ili dva posto razlike u točnosti predviđanja nema preveliki utjecaj, ako se predviđanje može ostvariti u značajnije kraćem inferencijskom vremenu.

Prilagođeni CRISP-DM metodološki okvir i *MoziaisMl* sustav omogućuju brzu i pouzdanu automatizaciju modela strojnog učenja u domenama gdje zadržavanje stabilne predikcijske točnosti i niska latencija predstavljaju važne mjere kriterija uspješnosti.

LITERATURA

[1] Zekić-Sušac, M., Mitrović, S., i Has, A., "Machine learning based system for managing energy efficiency of public sector as an approach towards smart cities," (in English), *Int J Inf Manage,* Article vol. 58, 2021, Art no. 102074, doi: 10.1016/j.ijinfomgt.2020.102074.

[2] Mitrović, S. i Zekić-Sušac, M., "A systematic literature review of machine learning algorithms in modeling buildings energy efficiency," in *WEENTECH Proceedings in Energy*, Edinburgh, United Kingdom, A. S. Agarwal, Renu, Ed., 25 January 2018 2017, vol. 5: World Energy and Environment Technology Ltd., 2018, pp. 191-196.

[3] Zekić-Sušac, M., Has, A., i Mitrović, S., "Recursive Partitioning in Predicting Energy Consumption of Public Buildings," in *Central European Conference on Information and Intelligent Systems : CECIIS 2018 : 29th International Conference Proceedings*, Varaždin: University of Zagreb Faculty of Organization and Informatics, V. K. Strahonja, Valentina, Ed., 2018, pp. 179-186.

[4] Tonkovic, Z., Mitrovic, S., i Zekic-Susac, M., "BUSINESS INTELLIGENCE SYSTEM FOR MANAGING NATURAL GAS CONSUMPTION OF PUBLIC BUILDINGS," in *36th International Scientific Conference on Economic and Social Development (ESD) - Building Resilient Society*, Zagreb, CROATIA, 2018

Dec 14-15 2018, in International Scientific Conference on Economic and Social Development, 2018, pp. 769-778.

[5] Baratchi, M. *et al.*, "Automated machine learning: past, present and future," (in English), *Artif Intell Rev,* Article vol. 57, no. 5, 2024, Art no. 122, doi: 10.1007/s10462-024-10726-1.

[6] Pidó, S., Pinoli, P., Crovari, P., Ieva, F., Garzotto, F., i Ceri, S., "Ask Your Data - Supporting Data Science Processes by Combining AutoML and Conversational Interfaces," (in English), *IEEE Access,* Article vol. 11, pp. 45972-45988, 2023, doi: 10.1109/ACCESS.2023.3272503.

[7] Rutqvist, D., Kleyko, D., i Blomstedt, F., "An automated machine learning approach for smart waste management systems," (in English), *IEEE Trans. Ind. Inf.,* Article vol. 16, no. 1, pp. 384-392, 2020, Art no. 8709695, doi: 10.1109/TII.2019.2915572.

[8] Schmitt, M., "Automated machine learning: AI-driven decision making in business analytics," (in English), *Intell. Syst. Applications.,* Article vol. 18, 2023, Art no. 200188, doi: 10.1016/j.iswa.2023.200188.

[9] Yayah, F. C., Ghauth, K. I., i Ting, C. Y., "The automated machine learning classification approach on telco trouble ticket dataset," (in English), *J. Eng. Sci. Technol.,* Article vol. 16, no. 5, pp. 4263-4282, 2021.

[10] Mitrović, S. i Vrček, N., "Automated machine learning methods for efficient prediction of carbon dioxide emissions in building sector," in *SINERGY*

International Conference on Smart Energy Management technologies, Belgrade, Serbia, September 26-29 2023.

[11] Shearer, C., "The CRISP-DM Model: The New Blueprint for Data Mining," *Journal of Data Warehousing,* vol. 5, no. 4, 2000.

[12] Mitrović, S. i Vrček, N., "Automation of classification tasks with imbalanced datasets using an adapted CRISP-DM methodology," in *Hinweis Third International Conference on Computer Science, Cyber Security and Information Technology (CCIT)*, June 28-29 2025.

[13] 5G.hr. "LATENCIJA." 5g.hr. https://www.5g.hr/pojam/latencija/ (pristupljeno 18.3.2025., 2025).

[14] Bartal, Y., Fandina, N., i Neiman, O., "Dimensionality reduction: theoretical perspective on practical measures," in *Neural Information Processing Systems*, 2019.

[15] Geron, A., *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2019.

[16] Van Der Maaten, L., Postma, E., i Van den Herik, J., "Dimensionality reduction: a comparative review," *J Mach Learn Res,* vol. 10, pp. 66--71, 2009.

[17] van der Maaten, L. i Hinton, G., "Viualizing data using t-SNE," *Journal of Machine Learning Research,* vol. 9, pp. 2579-2605, 11/01 2008.

[18] Healy, J. i McInnes, L., "Uniform manifold approximation and projection," *Nature Reviews Methods Primers,* vol. 4, no. 1, p. 82, 2024/11/21 2024, doi: 10.1038/s43586-024-00363-x.

[19] Goodfellow, I., Bengio, Y., i Courville, A., *Deep Learning*. MIT Press, 2016.

[20] Mitrović, S. i Vrček, N., "Methodology for the comparative analysis of PCA and Autoencoders in dimensionality reduction: impact on classification accuracy and computational efficiency," in *36th Central European Conference on Information and Intelligent Systems*, Varazdin, Croatia, September 17-19 2025.

[21] Chapman, P., *CRISP-DM 1.0: Step-by-step Data Mining Guide*. SPSS, 2000.

[22] Costa, R., *The CRISP-ML Methodology: A Step-by-Step Approach to Real-World Machine Learning Projects*. https://www.amazon.com/CRISP-ML-Methodology-Step-Step-Real-World/dp/B0BR9DQMW5: Independently published, 2022, p. 75.

[23] Lin, M. "Awesome-AutoML-Papers." Github. https://github.com/hibayesian/awesome-automl-papers (pristupljeno 3.4.2025., 2025).

[24] LeCun, Y., "Yann LeCun on a vision to make AI systems learn and reason like animals and humans," vol. 2022, ed. META AI: Meta, 2022.

[25] developers, s.-l. "RandomForestClassifier." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html (pristupljeno 6.6.2024., 2024).

[26] developers, s.-l. "RandomForestRegressor." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html (pristupljeno 6.6.2024., 2024).

[27] developers, x. "XGBoost." xgboost developers. https://xgboost.readthedocs.io/en/stable/ (pristupljeno 6.6.2024., 2024).

[28] developers, s.-l. "mutual_info_classif." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html (pristupljeno 6.6.2024., 2024).

[29] developers, s.-l. "mutual_info_regression." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html (pristupljeno 6.6.2024., 2024).

[30] developers, s.-l. "RFE." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html (pristupljeno 6.6.2024., 2024).

[31] developers, s.-l. "GridSearchCV." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html (pristupljeno 6.6.2024., 2024).

[32] developers, s.-l. "StratifiedKFold." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedKFold.html (pristupljeno 6.6.2024., 2024).

[33] developers, s.-l. "KFold." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html (pristupljeno 6.6.2024., 2024).

[34] Aria, M. a. C., Corrado, "bibliometrix : An R-tool for comprehensive science mapping analysis," (in "en"), *J. Informetr.,* vol. 11, no. 4, pp. 959--975, nov 2017, doi: 10.1016/j.joi.2017.08.007.

[35] Alabi, G., "Bradford's law and its application," *International Library Review,* vol. 11, no. 1, pp. 151-158, 1979/01/01/ 1979, doi: https://doi.org/10.1016/0020-7837(79)90044-X.

[36] Zhou, P. i Leydesdorff, L., "Fractional counting of citations in research evaluation: An option for cross- and interdisciplinary assessments," 12/01 2010.

[37] Friedman, A., "The Power of Lotka's Law Through the Eyes of R," *The Romanian Statistical Review,* vol. 2, 02/15 2015.

[38] Pedregosa, F. *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research,* vol. 12, pp. 2825--2830, 2011.

[39] Li, Y., Shen, Y., Zhang, W., Zhang, C., i Cui, B., "VolcanoML: speeding up end-to-end AutoML via scalable search space decomposition," (in English), *VLDB J.,* Article vol. 32, no. 2, pp. 389-413, 2023, doi: 10.1007/s00778-022-00752-2.

[40] Gosiewska, A., Kozak, A., i Biecek, P., "Simpler is better: Lifting interpretability-performance trade-off via automated feature engineering," (in English), *Decis Support Syst,* Article vol. 150, 2021, Art no. 113556, doi: 10.1016/j.dss.2021.113556.

[41] Yakovlev, A. *et al.*, "Oracle AutoML: A Fast and Predictive AutoML Pipeline," (in English), *Proc. VLDB Endow.,* Article vol. 13, no. 12, pp. 3166-3180, 2020, doi: 10.14778/3415478.3415542.

[42] Fister, I., Jr., Farthofer, L. A., Pecnik, L., Fister, I., i Holzinger, A., "NiaAML: AutoML for classification and regression pipelines," (in English), *SoftwareX,* Article vol. 29, p. 5, Feb 2025, Art no. 101974, doi: 10.1016/j.softx.2024.101974.

[43] Eldeeb, H. i Elshawi, R., "Empowering Machine Learning with Scalable Feature Engineering and Interpretable AutoML," (in English), *IEEE. Trans. Artif. Intell.,* Article pp. 1-16, 2024, doi: 10.1109/TAI.2024.3400752.

[44] Kedziora, D. J., Musial, K., i Gabrys, B., "AutonoML: Towards an Integrated Framework for Autonomous Machine Learning," (in English), *Found. Trends Mach. Learn.,* Article vol. 17, no. 4, pp. 590-766, 2024, doi: 10.1561/2200000093.

[45] Wu, Q., Wang, C., Langford, J., Mineiro, P., i Rossi, M., "ChaCha for Online AutoML," presented at the Proceedings of the 38th International Conference on Machine Learning, Proceedings of Machine Learning Research, 2021.

[46] Cohen, E. "Reducing Dimensionality from Dimensionality Reduction Techniques." Medium. https://medium.com/towards-data-science/reducing-dimensionality-from-dimensionality-reduction-techniques-f658aec24dfe (pristupljeno 2.7.2024., 2024).

[47] Jolliffe, I. T., *Principal Component Analysis*. Springer, 2002.

[48] Sachinsoni. "Mastering t-SNE(t-distributed stochastic neighbor embedding)." Medium. https://medium.com/@sachinsoni600517/mastering-t-sne-t-distributed-stochastic-neighbor-embedding-0e365ee898ea (pristupljeno 15.7.2024, 2024).

[49] Tyagi, K., Rane, C., i Manry, M., "Chapter 1 - Supervised learning," in *Artificial Intelligence and Machine Learning for EDGE Computing*, R. Pandey, S. K. Khatri, N. k. Singh, i P. Verma Eds.: Academic Press, 2022, pp. 3-22.

[50] Chandra, S. B. "Introduction to Autoencoders [Theory and Implementation]." Medium. https://medium.com/@c17hawke/introduction-to-autoencoders-theory-and-implementation-72fed7cf2f70 (pristupljeno 2.7.2024., 2024).

[51] Zhu, Y., Jiang, W., i Alonso, G., "Efficient Tabular Data Preprocessing of ML Pipelines," 2024.

[52] Pisinger, D. i Toth, P., "Knapsack Problems," in *Handbook of Combinatorial Optimization: Volume1–3*, D.-Z. Du i P. M. Pardalos Eds. Boston, MA: Springer US, 1998, pp. 299-428.

[53] Programming, A. f. C. "Knapsack Problem." cp-algorithms. https://cp-algorithms.com/dynamic_programming/knapsack.html (pristupljeno 30.11.2024, 2024).

[54] Bratkovsky, E., "A simple explanation of CRISP-ML for beginners," vol. 2024, ed. Kaggle: Kaggle, 2024.

[55] Kaggle. "Kaggle: Your Machine Learning and Data Science Community." Kaggle.com. https://www.kaggle.com/ (pristupljeno 1.7.2022., 2022).

[56] Kaggle. "Titanic - Machine Learning from Disaster." Kaggle. https://www.kaggle.com/competitions/titanic/overview (pristupljeno 10.8.2024., 2024).

[57] Chris, D. S. W. "Kaggle Titanic – Data Cleaning and Preprocessing." Data Science With Chris. https://datasciencewithchris.com/kaggle-titanic-data-cleaning-and-preprocessing/#:~:text=,number%20without%20the%20last%20digit (pristupljeno 10.8.2024., 2024).

[58] developers, s.-l. "LabelEncoder." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html (pristupljeno 20.6.2024., 2024).

[59] developers, s.-l. "OneHotEncoder." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html (pristupljeno 20.6.2024., 2024).

[60] developers, s.-l. "QuantileTransformer." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html (pristupljeno 20.6.2024., 2024).

[61] Krleža, L. z. M. "varijanca." Hrvatska enciklopedija. https://www.enciklopedija.hr/clanak/varijanca (pristupljeno 18.3.2025., 2025).

[62] developers, s.-l. "Cross-validation: evaluating estimator performance." scikit learn. https://scikit-learn.org/stable/modules/cross_validation.html (pristupljeno 6.6.2024., 2024).

[63] Breiman, L., Friedman, J., Stone, C. J., i Olshen, R. A., *Classification and Regression Trees*. Taylor & Francis, 1984.

[64] developers, s.-l. "Decision Trees." scikit learn. https://scikit-learn.org/stable/modules/tree.html (pristupljeno 6.6.2024., 2024).

[65] Hastie, T., Tibshirani, R., i Friedman, J., *The elements of statistical learning: data mining, inference and prediction*, 2 ed. Springer, 2009.

[66] Quinlan, J. R., *C4.5: Programs for Machine Learning*. Elsevier Science, 1993.

[67] developers, s.-l. "GaussianNB." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html (pristupljeno 6.6.2024., 2024).

[68] developers, s.-l. "Naive Bayes." scikit learn. https://scikit-learn.org/stable/modules/naive_bayes.html (pristupljeno 6.6.2024., 2024).

[69] developers, s.-l. "Support Vector Machines." scikit learn. https://scikit-learn.org/stable/modules/svm.html (pristupljeno 6.6.2024., 2024).

[70] developers, s.-l. "GradientBoostingClassifier." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html (pristupljeno 6.6.2024., 2024).

[71] developers, s.-l. "KNeighborsClassifier." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html (pristupljeno 6.6.2024., 2024).

[72] developers, s.-l. "MLPClassifier." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html (pristupljeno 6.6.2024., 2024).

[73] developers, s.-l. "LinearRegression." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html (pristupljeno 6.6.2024., 2024).

[74] developers, s.-l. "Lasso." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html (pristupljeno 6.6.2024., 2024).

[75] developers, s.-l. "Ridge." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html (pristupljeno 6.6.2024., 2024).

[76] developers, s.-l. "ElasticNet." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html (pristupljeno 6.6.2024., 2024).

[77] developers, s.-l. "SVR." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html (pristupljeno 6.6.2024., 2024).

[78] developers, s.-l. "KNeighborsRegressor." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html (pristupljeno 6.6.2024., 2024).

[79] developers, s.-l. "MLPRegressor." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html (pristupljeno 6.6.2024., 2024).

[80] developers, s.-l. "DecisionTreeRegressor." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html (pristupljeno 6.6.2024., 2024).

[81] developers, s.-l. "GradientBoostingRegressor." scikit learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html (pristupljeno 6.6.2024., 2024).

[82] Foundation, P. S. "pickle." Python Software Foundation. https://docs.python.org/3/library/pickle.html (pristupljeno 10.6.2024., 2024).

[83] International, E. "ECMA-404 The JSON data interchange syntax." Ecma International. https://ecma-international.org/publications-and-standards/standards/ecma-404/ (pristupljeno 20.6.2024., 2024).

[84] Montoya, A. i DataCanary. "House Prices - Advanced Regression Techniques." Kaggle. https://kaggle.com/competitions/house-prices-advanced-regression-techniques (pristupljeno 10.6.2024., 2024).

[85] Sukul, A., State), A. I., Zinnel, L., i xiangm. "Lab 9 Classifiers: Which Vaccine Was It?" Kaggle. https://www.kaggle.com/competitions/which-vaccine-class-was-it/overview (pristupljeno 10.8.2024., 2024).

[86] Montoya, A. i DataCanary. "House Prices - Advanced Regression Techniques - expanded test dataset 50459 rows." Kaggle. https://www.kaggle.com/datasets/sasamitrovicos/house-prices-advanced-regression-techniques (pristupljeno 10.1.2025., 2025).

[87] Sukul, A., State), A. I., Zinnel, L., i xiangm. "Lab 9 Classifiers: Which Vaccine Was It? - Expanded 50k test data." Kaggle. https://www.kaggle.com/datasets/sasamitrovicos/multi-class-classifiers-which-vaccine-was-it/data (pristupljeno 20.2.2025., 2025).

[88] Mukhiya, S. K. i Ahmed, U., *Hands-On Exploratory Data Analysis with Python: Perform EDA Techniques to Understand, Summarize, and Investigate Your Data*. Packt Publishing, Limited, 2020.

[89] Ross, A. i Willson, V. L., "Paired Samples T-Test," in *Basic and Advanced Statistical Tests: Writing Results Sections and Creating Tables and Figures*. Rotterdam: SensePublishers, 2017, pp. 17-19.

[90] González-Manteiga, W. i Crujeiras, R. M., "Bootstrap Methods," in *International Encyclopedia of Statistical Science*, M. Lovric Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2025, pp. 327-334.

[91] Mokhtar, S., Yusof, Z., i Sapiri, H., "Confidence Intervals by Bootstrapping Approach: A Significance Review," *Malaysian Journal of Fundamental and Applied Sciences,* vol. 19, pp. 30-42, 02/25 2023, doi: 10.11113/mjfas.v19n1.2660.

[92] community, T. S. "scipy.stats.ttest_rel." The SciPy community. https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.ttest_rel.html (pristupljeno 15.10.2025., 2025).

PRILOG 1. Python kod

```
# -*- coding: utf-8 -*-
"""
@author: Sasa
"""
import os
import graphviz
import itertools
from IPython.display import display
import matplotlib.pyplot as plt # graphing with insane defaults
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import plotly.express as px
import seaborn as sns # graphing with sane defaults
import scipy.stats as stats
from sklearn import linear_model
from sklearn import preprocessing # Preprocess data (e.g. scale numerical data to 0-1
from sklearn import tree
from sklearn.base import BaseEstimator, clone
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
import umap as umap # UMAP za PC1 - paziti na verziju, Mac M2 je OK
from sklearn.discriminant_analysis import QuadraticDiscriminantAnalysis
from sklearn.exceptions import ConvergenceWarning
from sklearn.ensemble import RandomForestRegressor, ExtraTreesRegressor, GradientBoostingRegressor, AdaBoostRegressor, HistGradientBoostingRegressor, IsolationForest, AdaBoostClassifier, RandomForestClassifier, GradientBoostingClassifier, HistGradientBoostingClassifier
from sklearn.feature_selection import SelectFromModel, SelectKBest, chi2, f_classif, f_regression, r_regression, mutual_info_classif, mutual_info_regression, SequentialFeatureSelector, VarianceThreshold, RFE
from sklearn.gaussian_process import GaussianProcessClassifier
import sklearn.metrics as metrics
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
```

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report, PrecisionRecallDisplay, roc_auc_score
from sklearn.model_selection import train_test_split, KFold, GridSearchCV, StratifiedKFold, cross_val_score
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score, accuracy_score
from sklearn.utils._testing import ignore_warnings
from termcolor import colored, cprint
import typing # Apply common types to objects
import warnings
from yellowbrick.classifier.rocauc import roc_auc
from yellowbrick.classifier import precision_recall_curve
from yellowbrick.features import PCA as yellowPCA, Manifold
from yellowbrick.regressor import ResidualsPlot, PredictionError
from sklearn.preprocessing import QuantileTransformer
from mlxtend.feature_selection import ExhaustiveFeatureSelector as EFS, SequentialFeatureSelector as SFS
from lightgbm import LGBMClassifier
import lightgbm as lgb
import matplotlib.gridspec as gridspec
from feature_engine.selection import SelectByShuffling, RecursiveFeatureElimination, DropConstantFeatures, DropDuplicateFeatures
from sklearn.pipeline import Pipeline
# adapted from: http://www.codiply.com/blog/hyperparameter-grid-search-across-multiple-models-in-scikit-learn/


import joblib
from datetime import datetime
from xgboost import XGBClassifier, XGBRegressor
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.linear_model import Lasso, Ridge, ElasticNet
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
```

```
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
import json
import time
import pyodbc
import xlsxwriter
import re
from scipy.stats import ttest_rel, bootstrap
import tensorflow # radi na svim enviromentima
from tensorflow.keras.models import Model  # radi na svim enviromentima
from tensorflow.keras.layers import Input, Dense  # radi na svim enviromentima
warnings.simplefilter(action='ignore', category=FutureWarning)


FeatureEngineering = typing.TypeVar('MoziaisMlSystem.FeatureEngineering')
FeatureSelector = typing.TypeVar('MoziaisMlSystem.FeatureSelector')
DimensionalityReductionTechniques = typing.TypeVar('MoziaisMlSystem.DimensionalityReductionTechniques')



class MoziaisMlSystem:
    class FoiPhd:
        def About():
            about = 'PhD theme :: HR :: Metodološki okvir za izradu analitičkog informacijskog sustava temeljenog na metodama strojnog učenja ' + os.linesep + 'PhD theme ::  EN :: Methodological framework for developing analytical information system based on machine learning methods'+ os.linesep + 'Student :: Sasa Mitrovic :: Mentor :: Neven Vrcek :: Valentina Janev' + os.linesep + 'Development Status :: Final - 1.0.0'+ os.linesep + 'Intended Audience :: Science/Research' + os.linesep + os.linesep + 'Operating System :: Windows :: macOS' + os.linesep +  'Programming Language :: Python :: 3.9.7'
            print(about)
        def percentage_difference_calculator(self, v1 : float, v2 : float):
            """
            Vraća postotnu razliku između dva broja.
            -------
            broj1-v1 : float
```

```
        broj2-v2 : float
        """
        percentage = ((v1-v2)/((v1+v2)/2))*100
        print(percentage, " %")
    def convert_seconds_to_milliseconds(self, seconds):
        milliseconds = seconds * 1000
        return milliseconds
    def objective_function(self, execution_time, accuracy_impact, alpha=0.7,
                           min_execution_time=100, max_execution_time=1000,
                           min_accuracy_impact=0.01, max_accuracy_impact=0.5):
        """
        Funkcija cilja koja balansira između vremena izvršavanja i utjecaja latencije na preciznost.

        :param execution_time: Vrijeme izvršavanja algoritma glasacki sustav ili naprtnjaca  (manje je bolje)
        :param accuracy_impact: Utjecaj latencije  algoritma glasacki sustav ili naprtnjaca na točnost predikcije (niže je bolje)
        :param alpha: Težinski faktor za balansiranje između brzine i preciznosti (0 ≤ alpha ≤ 1)
        :param min_execution_time: Najmanje vrijeme izvršavanja u skupu podataka
        :param max_execution_time: Najveće vrijeme izvršavanja u skupu podataka
        :param min_accuracy_impact: Najmanji utjecaj latencije na preciznost
        :param max_accuracy_impact: Najveći utjecaj latencije na preciznost
        :return: Optimizacijska vrijednost (niže je bolje)
        """
        #execution_time = self.convert_seconds_to_milliseconds(execution_time)
        #min_execution_time = self.convert_seconds_to_milliseconds(min_execution_time)
        #max_execution_time = self.convert_seconds_to_milliseconds(max_execution_time)
        # Normalizacija podataka
        exec_time_norm = (execution_time - min_execution_time) / (max_execution_time - min_execution_time)
        acc_impact_norm = (accuracy_impact - min_accuracy_impact) / (max_accuracy_impact - min_accuracy_impact)
```

```
        # Funkcija cilja
        objective_value = alpha * exec_time_norm + (1 - alpha) * acc_impact_norm

        return objective_value

# Definicija funkcije cilja s parametriziranim alpha 0.5 - 0.9
def alpha_5_9(self):
    # Skup podataka za sve tri domene
    datasets = {
        "Titanic - Metoda Glasanja": {
            "exec_time": 47.18,
            "acc_impact": 0.337320574,
            "min_exec_time": 38.43,
            "max_exec_time": 66.12,
            "min_acc_impact": 0.394736842,
            "max_acc_impact": 0.311004785,
        },
        "Titanic - Metoda Ruksaka": {
            "exec_time": 43.10,
            "acc_impact": 0.385167464,
            "min_exec_time": 43.10,
            "max_exec_time": 67.00,
            "min_acc_impact": 0.449760766,
            "max_acc_impact": 0.315789474,
        },
        "House - Metoda Glasanja": {
            "exec_time": 26.02,
            "acc_impact": 42249.71,
            "min_exec_time": 18.68,
            "max_exec_time": 172.68,
            "min_acc_impact": 41544.31,
            "max_acc_impact": 59289.90,
        },
```

```
    "House - Metoda Ruksaka": {
        "exec_time": 49.94,
        "acc_impact": 37933.15,
        "min_exec_time": 27.93,
        "max_exec_time": 172.68,
        "min_acc_impact": 37933.15,
        "max_acc_impact": 55919.27,
    },
    "Vaccine - Metoda Glasanja i Metoda Ruksaka": {
        "exec_time": 143.29,
        "acc_impact": 0.376321839,
        "min_exec_time": 84.85,
        "max_exec_time": 1220.54,
        "min_acc_impact": 0.376321839,
        "max_acc_impact": 0.248275862,
    }
}
# Raspon alpha vrijednosti
alphas = np.linspace(0.5, 0.9, 100)
# Izračun vrijednosti funkcije cilja za svaki dataset
results = {}
for name, params in datasets.items():
    values = []
    for alpha in alphas:
        value = self.objective_function(
            params["exec_time"], params["acc_impact"], alpha,
            params["min_exec_time"], params["max_exec_time"],
            params["min_acc_impact"], params["max_acc_impact"]
        )
        values.append(value)
    results[name] = values
# Grafički prikaz
for name, values in results.items():
    plt.plot(alphas, values, label=name)
```

```
    plt.xlabel("Vrijednost parametra α")
    plt.ylabel("Vrijednost funkcije cilja")
    plt.title("Osjetljivost funkcije cilja na varijacije parametra α")
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

def paired_t_test(self, model1_scores, model2_scores):
    """
    Izvodi parni t-test između dvaju skupova rezultata modela.

    Parameters
    ----------
    model1_scores : list or np.array
        Rezultati prvog modela.
    model2_scores : list or np.array
        Rezultati drugog modela.

    Returns
    -------
    t_stat : float
        T-statistika testa.
    p_value : float
        P-vrijednost testa.
    """
    t_stat, p_value = stats.ttest_rel(model1_scores, model2_scores)
    return t_stat, p_value

def paired_t_test_all_datasets_results(self):
    """
    Izvodi parni t-test za sve kombinacije modela unutar svakog skupa podataka.
```

```
    Returns
    -------
    results_summary : dict
       Rječnik sa sažetkom rezultata t-testa skupno za sve skupove podataka.
    """
    # Vrijeme izvršavanja: bez redukcije vs. Glasački vs. Naprtnjača
    execution_baseline = [57.53, 172.68, 1220.54]
    execution_voting = [47.18, 26.02, 143.29]
    execution_knapsack = [43.10, 49.94, 143.29]

    # Preciznost (Titanic, Vaccine) ili RMSE (House Prices): niže je bolje samo za RMSE
    accuracy_baseline = [0.339712919, 41544.31, 0.376091954]
    accuracy_voting = [0.337320574, 42249.71, 0.376321839]
    accuracy_knapsack = [0.385167464, 37933.15, 0.376321839]

    # H1: Paired t-test za vrijeme izvršavanja (alpha = 0.7)
    t_stat_voting_h1, p_value_voting_h1 = ttest_rel(execution_baseline, execution_voting)
    t_stat_knapsack_h1, p_value_knapsack_h1 = ttest_rel(execution_baseline, execution_knapsack)

    # H2: Paired t-test za preciznost / RMSE (ovisno o metrika tipa)
    t_stat_voting_h2, p_value_voting_h2 = ttest_rel(accuracy_baseline, accuracy_voting)
    t_stat_knapsack_h2, p_value_knapsack_h2 = ttest_rel(accuracy_baseline, accuracy_knapsack)

    # Prikaz rezultata
    results = {
       "H1 - Voting vs Baseline": (t_stat_voting_h1, p_value_voting_h1),
       "H1 - Knapsack vs Baseline": (t_stat_knapsack_h1, p_value_knapsack_h1),
       "H2 - Voting vs Baseline": (t_stat_voting_h2, p_value_voting_h2),
       "H2 - Knapsack vs Baseline": (t_stat_knapsack_h2, p_value_knapsack_h2)
    }
```

```
# Grafički prikaz razlika u vremenu izvršavanja
labels = ["Titanic", "House", "Vaccine"]
x = np.arange(len(labels))
width = 0.25

fig, ax = plt.subplots()
ax.bar(x - width, execution_baseline, width, label='Bez redukcije')
ax.bar(x, execution_voting, width, label='Metoda Glasanja')
ax.bar(x + width, execution_knapsack, width, label='Metoda Ruksaka')

ax.set_ylabel('Vrijeme izvršavanja (ms)')
ax.set_title('Usporedba vremena izvršavanja po metodi i skupu podataka')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

# Grafički prikaz razlika u točnosti / RMSE
# Grafikon 1: Titanic i Vaccine..točnost je 0.3 do 0.4
fig, ax1 = plt.subplots()
ax1.bar([0 - width, 2 - width], [accuracy_baseline[0], accuracy_baseline[2]], width, label='Bez redukcije')
ax1.bar([0, 2], [accuracy_voting[0], accuracy_voting[2]], width, label='Metoda Glasanja')
ax1.bar([0 + width, 2 + width], [accuracy_knapsack[0], accuracy_knapsack[2]], width, label='Metoda Ruksaka')

ax1.set_ylabel('Točnost predikcije')
ax1.set_title('Usporedba točnosti (Titanic i Vaccine)')
ax1.set_xticks([0, 2])
ax1.set_xticklabels(["Titanic", "Vaccine"])
ax1.legend()
```

```
        plt.grid(True)
        plt.tight_layout()
        plt.show()

        # Grafikon 2: House Prices...RMSE doseže 40000
        fig, ax2 = plt.subplots()
        ax2.bar([1 - width], [accuracy_baseline[1]], width, label='Bez redukcije')
        ax2.bar([1], [accuracy_voting[1]], width, label='Metoda Glasanja')
        ax2.bar([1 + width], [accuracy_knapsack[1]], width, label='Metoda Ruksaka')

        ax2.set_ylabel('RMSE')
        ax2.set_title('Usporedba RMSE (House Prices)')
        ax2.set_xticks([1])
        ax2.set_xticklabels(["House"])
        ax2.legend()
        plt.grid(True)
        plt.tight_layout()
        plt.show()

        return results


    def test_hypothesis_H1_execution_time(self, improved_times= [47.18, 43.10, 26.02, 49.94, 143.29], original_times= [57.53, 57.53, 172.68, 172.68, 1220.54], alpha=0.7, threshold=-0.15, n_resamples=10000):
        """
        Testira hipotezu H1 o smanjenju vremena izvršavanja za više od 15% pomoću postotnih razlika i bootstrap CI.

        :param improved_times: Lista vremena izvršavanja za optimizirane metode (Glasanje, Ruksak)
        :param original_times: Lista vremena izvršavanja za originalni model bez optimizacije
        :param alpha: Razina značajnosti (default 0.7 za 95% CI)
        :param threshold: Prag postotnog poboljšanja (-15%)
        :param n_resamples: Broj bootstrap uzoraka
```

```
"""
#improved_times = [47.18, 43.10, 26.02, 49.94, 143.29]     # glasanje/ruksak
#original_times = [57.53, 57.53, 172.68, 172.68, 1220.54]  # bez redukcije

# Izračun relativnih postotnih promjena za svaki par
pct_changes = [(new - old) / old for new, old in zip(improved_times, original_times)]

# Bootstrap CI
data = (np.array(pct_changes),)
ci = bootstrap(data, np.mean, confidence_level=1 - alpha, n_resamples=n_resamples, method='percentile')
mean_change = np.mean(pct_changes)
ci_low, ci_high = ci.confidence_interval

# Ispis rezultata
print(f"Srednja postotna promjena vremena izvršavanja: {mean_change*100:.2f}%")
print(f"95% CI: [{ci_low*100:.2f}%, {ci_high*100:.2f}%]")
if ci_low < threshold:
    print("H1 je potvrđena: donja granica CI prelazi prag od -15%.")
else:
    print("H1 nije potvrđena: donja granica CI ne prelazi prag od -15%.")

# Vizualizacija
sns.histplot(pct_changes, kde=True)
plt.axvline(x=mean_change, color='blue', linestyle='--', label='Srednja promjena')
plt.axvline(x=ci_low, color='green', linestyle='--', label='95% CI - donja granica')
plt.axvline(x=ci_high, color='red', linestyle='--', label='95% CI - gornja granica')
plt.axvline(x=threshold, color='black', linestyle=':', label='Prag H1 (-15%)')
plt.title("Distribucija postotnih promjena vremena izvršavanja")
plt.xlabel("Relativna promjena (%)")
plt.ylabel("Broj slučajeva (učestalost)")
plt.legend()
plt.tight_layout()
```

```
        plt.show()



class FeatureEngineering:
    def get_dataset_from_database(self, servername : str, database : str, sql_query : str):
        """
        SQL upit za dohvat podataka iz SQL Server baze podataka.
        """
        DB = {'servername': servername,
            'database': database}# create the connection
        conn = pyodbc.connect('DRIVER={SQL Server};SERVER=' + DB['servername'] + ';DATABASE=' + DB['database'] + ';Trusted_Connection=yes')
        df = pd.read_sql_query(sql_query,conn)
        return df
    def preprocessing_data_pipeline(self, df: pd.DataFrame, test_df: pd.DataFrame, target_col: str = "", encodecolumns=12, mlc=False):
        """
        Primjenjuje niz transformacija na ulazne DataFrameove (trening i test skup) kako bi se podaci pripremili
        za analizu u strojnom učenju. Uključene transformacije su:
        - Detekcija nedostajućih vrijednosti i popunjavanje tih vrijednosti (s kreiranjem pomoćnih stupaca)
        - Detekcija dupliciranih redaka (s ispisom)
        - Rastavljanje vrijednosti u tekstualnim (object) stupcima prema razmaku (ili drugim separatorima)
        - Uklanjanje identifikacijskih (ID) stupaca
        - Uklanjanje stupaca koji nisu zajednički između trening i test skupa (osim ciljnog stupca)
        - Enkodiranje kategorijskih varijabli (ovisno o postavci mlc)
        - Kvantilna transformacija značajki (osim ciljnog stupca)

        Parametri
        ----------
        df : pd.DataFrame
            Trening skup podataka koji se transformira.
        test_df : pd.DataFrame
```

Test skup podataka koji se transformira.

target_col : str, optional

Naziv ciljnog stupca koji se ne transformira ili uklanja, by default "".

encodecolumns : int, optional

Prag broja jedinstvenih vrijednosti za izbor metode enkodiranja, by default 12.

mlc : bool, optional

Ako je True, koristi se posebna metoda za enkodiranje kategorijskih stupaca, by default False.

Vraća

(pd.DataFrame, pd.DataFrame)

Transformirani trening i test skup podataka.

"""

```
# 1. Detekcija nedostajućih vrijednosti u trening i test skupovima
missing_columns = self.detect_missing(df, "Train podaci")
test_missing_columns = self.detect_missing(test_df, "Test podaci")

# 2. Popunjavanje nedostajućih vrijednosti s vrijednošću 0, uz kreiranje pomoćnih stupaca
df = self.fill_missing_create_columns(df, missing_columns, 0)
test_df = self.fill_missing_create_columns(test_df, test_missing_columns, 0)

# 3. Detekcija dupliciranih redaka (samo se ispisuju, ne mijenjaju podatke)
self.detect_duplicates(df)

# 4. Rastavljanje vrijednosti u object stupcima prema razmaku
df = self.break_up_by_string(df, ' ')
test_df = self.break_up_by_string(test_df, ' ')

# 5. Uklanjanje ID stupaca iz oba skupa (stupci u kojima je svaki red jedinstven)
id_cols = self.detect_id_columns(df.copy())
df = df.drop(columns=id_cols)
test_df = test_df.drop(columns=id_cols)
```

```
        # 6. Uklanjanje stupaca koji nisu zajednički između trening i test skupa (osim ciljnog stupca)
        self.drop_unshared_columns(df, test_df, [target_col])

        # 7. Enkodiranje kategorijskih stupaca
        if mlc:
            # Ako je mlc True, koristi se metoda za enkodiranje kategorijskih stupaca koja vraća i enkodere
            df, encoders = self.encode_categorical_columns(df)
            test_df, encoders = self.encode_categorical_columns(test_df)
        else:
            # Inače, enkodiraju se stupci tipa object prema zadanom pragu (encodecolumns)
            df, test_df = self.encode_columns(df, df.select_dtypes(include=['object']).columns, test_df, encodecolumns)

        # 8. Ponovno uklanjanje stupaca koji nisu zajednički (nakon eventualnih promjena tijekom enkodiranja)
        self.drop_unshared_columns(df, test_df, [target_col])

        # 9. Kvantilna transformacija značajki (sve osim ciljnog stupca) radi normalizacije raspodjele
        df = self.quantile_transform_column_wise(df, target_col)
        test_df = self.quantile_transform_column_wise(test_df, target_col)

        return df, test_df
    def detect_missing(self, df_temp: pd.DataFrame, name: str = '', silent: bool = False, plot: bool = True):
        """
        Detektira nedostajuće vrijednosti u proslijeđenom DataFrameu.
        Metoda pronalazi vrijednosti koje su:
        - Pandas-ova null (NaN, None, itd.)
        - Stringovi koji odgovaraju: "NA", "N/A", "NaN", "NULL" ili prazno

        Parameters
```

```
    ----------
    df_temp : pd.DataFrame
        DataFrame u kojem se detektiraju nedostajuće vrijednosti.
    name : str, optional
        Naziv DataFramea radi opisnijeg ispisa, by default ''
    silent : bool, optional
        Ako je True, ispis se ne prikazuje, by default False
    plot : bool, optional
        Ako je True, prikazuje se bar plot broja nedostajućih vrijednosti po stupcima, by default True

    Returns
    -------
    List[str]
        Lista imena stupaca u kojima su pronađene nedostajuće vrijednosti.
    """
    # Definicija skupova string vrijednosti koje se smatraju "missing"
    missing_markers = {"NA", "N/A", "NAN", "NULL", ""}

    # Kreiraj masku: True za one elemente koji su ili pd.isnull() ili (ako su string i odgovaraju nekoj od vrijednosti)
    mask = df_temp.applymap(lambda x: pd.isnull(x) or (isinstance(x, str) and x.strip().upper() in missing_markers))

    # Ukupan broj nedostajućih vrijednosti
    count_missing = mask.sum().sum()

    # Stupci u kojima ima nedostajućih vrijednosti
    columns_with_missing = list(mask.columns[mask.any()])

    # Ispis rezultata, ako silent nije True
    if not silent:
        if name:
            print(f'Otkrivam nedostajuće vrijednosti u {name}')
        print(f'Nedostajuće vrijednosti u podacima: {count_missing}')
```

```
            if count_missing > 0:
                print('******')
                for col in columns_with_missing:
                    col_missing = mask[col].sum()
                    print(f'Nedostajuće vrijednosti u {col}: {col_missing}')
                print('******')
        print('')

        # Ako je zatražen plot i ima nedostajućih vrijednosti, prikaži bar plot
        if plot and count_missing > 0:
            # Računamo broj missing vrijednosti po stupcima (samo za one stupce gdje ih ima)
            missing_counts = mask.sum()[mask.sum() > 0]
            sns.barplot(x=missing_counts.index, y=missing_counts.values)
            plt.show()

        return columns_with_missing
    def fill_missing_create_columns(self, df_temp: pd.DataFrame, columns: typing.List[str], value: float = 0) -> pd.DataFrame:
        """
        Popunjava nedostajuće vrijednosti u odabranim stupcima i stvara nove stupce koji označavaju
        redove u kojima su se pojavile nedostajuće vrijednosti. Nedostajuće vrijednosti se prepoznaju
        kao:
        - Pandas-ove null vrijednosti (NaN, None, itd.)
        - String vrijednosti: "NA", "N/A", "NaN", "NULL" i prazni string

        Parameters
        ----------
        df_temp : pd.DataFrame
            DataFrame koji se modificira
        columns : typing.List[str]
            Popis stupaca u DataFrameu koji se obrađuju
        value : float, optional
```

```
        Vrijednost kojom će se zamijeniti nedostajuće vrijednosti, by default 0

        Returns
        -------
        pd.DataFrame
            Modificirani DataFrame s popunjenim nedostajućim vrijednostima i novim stupcima koji označavaju
            redove u kojima su se pojavile nedostajuće vrijednosti.
        """
        # Definiramo skup string vrijednosti koje se smatraju nedostajućima
        missing_markers = {"NA", "N/A", "NAN", "NULL", ""}

        for col in columns:
            # Kreiraj masku koja označava nedostajuće vrijednosti:
            # True ako je vrijednost pandas-ov null ili (ako je string i odgovara nekoj od missing_markers)
            mask = df_temp[col].isnull() | df_temp[col].astype(str).str.strip().str.upper().isin(missing_markers)

            # Stvori novi stupac koji označava redove u kojima je vrijednost bila nedostajuća (1 = nedostajuća, 0 = nije)
            df_temp[col + "_was_missing"] = mask.astype(int)

            # Zamijeni nedostajuće vrijednosti s proslijeđenom vrijednošću
            df_temp[col] = df_temp[col].mask(mask, other=value)

        return df_temp
    def detect_duplicates(self, df_temp: pd.DataFrame, silent: bool = False):
        """
        Detektira duplicirane vrijednosti u podacima i ispisuje stupce s jedinstvenim vrijednostima (ID stupce).

        Parameters
        ----------
        df_temp : pd.DataFrame
```

```
        DataFrame u kojem se traže duplicirane vrijednosti.
    silent : bool
        Ako je True, ispis nije prikazan.

    Returns
    -------
    tuple
        Tuple koji sadrži:
        - count_dupes: broj dupliciranih redaka u filtriranom DataFrameu
        - id_cols: lista stupaca koji sadrže samo jedinstvene vrijednosti (ID stupci)
    """
    # Koristi vectoriziranu metodu nunique za dobivanje broja jedinstvenih vrijednosti po stupcu
    nunique = df_temp.nunique()
    # Stupci s dupliciranim vrijednostima (nije identičan broj jedinstvenih vrijednosti i duljini DataFramea)
    cols_to_use = nunique[nunique < len(df_temp)].index.tolist()
    # Stupci s isključivo jedinstvenim vrijednostima
    id_cols = nunique[nunique == len(df_temp)].index.tolist()

    # Filtriraj DataFrame tako da ostanu samo stupci u kojima se mogu pojaviti duplikati
    df_filtered = df_temp[cols_to_use].copy()
    count_dupes = df_filtered.duplicated().sum()

    if not silent:
        print('Duplicirane vrijednosti u podacima:', count_dupes)
        print('Kad se izfiltriraju ID stupci:', id_cols)

    return count_dupes, id_cols
def break_up_by_string(self, df_temp: pd.DataFrame, splitting_string: str) -> pd.DataFrame:
    """
    Rastavlja vrijednosti u object stupcima DataFramea prema zadanom separatoru i dodaje nove stupce s dobivenim dijelovima.
```

```
    Ako je splitting_string "CL", rastavlja tekst prema početnom velikom slovu.

    Parameters
    ----------
    df_temp : pd.DataFrame
        DataFrame u kojem se obavlja rastavljanje.
    splitting_string : str
        String po kojem se vrši rastavljanje. Ako je "CL", rastavlja se prema početnom velikom slovu.

    Returns
    -------
    pd.DataFrame
        Modificirani DataFrame s dodatnim stupcima koji sadrže rastavljene vrijednosti.
    """
    # Radimo kopiju ulaznog DataFramea
    df_result = df_temp.copy()
    # Odaberi stupce tipa object
    obj_cols = df_result.select_dtypes(include=[object]).columns

    for col in obj_cols:
        # Ako radimo na temelju "Capital Letters" režima
        if splitting_string == "CL":
            # Koristimo vectoriziranu metodu za pronalazak svih pojavljivanja obrasca:
            # svaki token započinje velikim slovom i slijedi niz znakova koji nisu velika slova.
            split_series = df_result[col].str.findall(r'([A-Z][^A-Z]*)')
            # Preskoči stupac ako nema nijednog pronađenog tokena
            if split_series.dropna().empty:
                continue
            # Proširi listu u stupac pomoću apply(pd.Series)
            df_splits = split_series.apply(pd.Series)
            # Očisti eventualne vodeće/završne razmake u tokenima
            df_splits = df_splits.applymap(lambda x: x.strip() if isinstance(x, str) else x)
```

```
            # Kreiraj nove nazive stupaca: npr. originalni naziv + "_CL" + redni broj
            rename_dict = {i: f"{col}_CL{i}" for i in df_splits.columns}
        else:
            # Ako je zadani separator neki drugi string
            # Provjeri ima li u stupcu barem jedno pojavljivanje separatora
            if not df_result[col].str.contains(splitting_string, na=False).any():
                continue
            # Rastavi stupac koristeći zadani separator (vectorizirano)
            df_splits = df_result[col].str.split(splitting_string, expand=True)
            rename_dict = {}
            for i in df_splits.columns:
                # Ako separator nije prazni razmak, uključujemo ga u naziv stupca
                if splitting_string != " ":
                    rename_dict[i] = f"{col}{splitting_string}{i}"
                else:
                    rename_dict[i] = f"{col}{i}"
            # Preimenuj nove stupce
            df_splits.rename(columns=rename_dict, inplace=True)
            # Popuni eventualne NaN vrijednosti (npr. ako je broj dijelova različit) s 0
            df_splits = df_splits.fillna(0)
            # Spoji originalni DataFrame s novim stupcima
            df_result = pd.concat([df_result, df_splits], axis=1)

    return df_result
def detect_id_columns(self, df_temp: pd.DataFrame) -> typing.List[str]:
    """
    Detektira stupce koji su ID stupci, odnosno one u kojima postoji jedinstvena vrijednost za svaki red.

    Parameters
    ----------
    df_temp : pd.DataFrame
        DataFrame u kojem se traže ID stupci.
```

```
        Returns
        -------
        typing.List[str]
            Lista ID stupaca koji su detektirani.
        """
        # Dobivanje broja jedinstvenih vrijednosti za svaki stupac
        nunique = df_temp.nunique()
        # Odabir stupaca gdje je broj jedinstvenih vrijednosti jednak broju redaka
        id_cols = nunique[nunique == len(df_temp)].index.tolist()
        return id_cols
    def drop_unshared_columns(self, df_temp: pd.DataFrame, df_temp_2: pd.DataFrame, exclude_columns: typing.List[str]):
        """
        Uklanja stupce koji nisu zajednički između dva DataFramea, osim onih navedenih u exclude_columns.
        Brisanje se vrši in-place.

        Parameters
        ----------
        df_temp : pd.DataFrame
            Prvi DataFrame za provjeru zajedničkih stupaca.
        df_temp_2 : pd.DataFrame
            Drugi DataFrame za provjeru zajedničkih stupaca.
        exclude_columns : typing.List[str]
            Lista stupaca koji se ne uklanjaju.

        Returns
        -------
        None
        """
        # Izračunaj stupce koje treba ukloniti iz df_temp_2: oni koji nisu u df_temp, a nisu ni u exclude_columns
        drop_cols_df2 = set(df_temp_2.columns) - set(df_temp.columns) - set(exclude_columns)
        if drop_cols_df2:
```

```
            df_temp_2.drop(columns=list(drop_cols_df2), inplace=True)


        # Izračunaj stupce koje treba ukloniti iz df_temp: oni koji nisu u df_temp_2, a nisu ni u exclude_columns
        drop_cols_df1 = set(df_temp.columns) - set(df_temp_2.columns) - set(exclude_columns)
        if drop_cols_df1:
            df_temp.drop(columns=list(drop_cols_df1), inplace=True)
    def encode_categorical_columns(self, df: pd.DataFrame):
        """
        Automatski pretvara sve stupce tipa 'object' u brojčane vrijednosti koristeći Label Encoding.

        :param df: pandas DataFrame
        :return: DataFrame s konvertiranim kategorijama
        """
        df_encoded = df.copy()
        label_encoders = {}

        for column in df_encoded.select_dtypes(include=['object']).columns:
            le = preprocessing.LabelEncoder()
            df_encoded[column] = le.fit_transform(df_encoded[column])
            label_encoders[column] = le  # Sprema encoder ako treba kasnije dekodirati

        return df_encoded, label_encoders
    def encode_columns(self, df: pd.DataFrame, columns: typing.List[str], test_df: pd.DataFrame = None, cutoff: int = 12, specific_col: str = 'Vaccine'):
        """
        Enkodira stupce na temelju broja jedinstvenih vrijednosti. Ako broj jedinstvenih vrijednosti
        (uz dodatak 'None' za nepoznate vrijednosti) je manji od cutoff (i nije točno 3), koristi se one-hot encoding;
        inače se primjenjuje label encoding. Za test podatke, nepoznate vrijednosti se zamjenjuju s 'None'.

        Parameters
```

```
    ----------
    df : pd.DataFrame
        DataFrame u kojem se enkodiraju stupci.
    columns : typing.List[str]
        Popis stupaca koji se enkodiraju.
    test_df : pd.DataFrame, optional
        Test DataFrame koji se enkodira prema klasama iz trening podataka, by default None.
    cutoff : int, optional
        Prag broja klasa za odabir između label encodinga i one-hot encodinga, by default 12.
    specific_col : str, optional
        Specifični stupac za koji se ne primjenjuje one-hot encoding u test setu, by default 'Vaccine'.

    Returns
    -------
    (pd.DataFrame, pd.DataFrame)
        Enkodirani DataFrame i, ako je zadan, i test DataFrame.
    """
    for col in columns:
        # Priprema stupca: radimo s tekstualnim reprezentacijama
        col_series = df[col].astype(str)
        # Dohvati jedinstvene vrijednosti, sortiraj i dodaj 'None' za nepoznate vrijednosti
        classes_to_encode = sorted(col_series.unique().tolist())
        if 'None' not in classes_to_encode:
            classes_to_encode.append('None')
        le = preprocessing.LabelEncoder()
        le.fit(classes_to_encode)

        # Ako je broj klasa manji od cutoff i nije točno 3, koristi se one-hot encoding
        if len(le.classes_) < cutoff and len(le.classes_) != 3:
            df = pd.get_dummies(df, columns=[col])
            if test_df is not None and col != specific_col:
```

```
                test_df = pd.get_dummies(test_df, columns=[col])
            else:
                # Label encoding za trening podatke
                df[col] = le.transform(col_series)
                if test_df is not None:
                    # Enkodiranje test podataka: zamijeni vrijednosti koje nisu u trening setu
s 'None'
                    test_col = test_df[col].astype(str)
                    valid_values = set(col_series.unique())
                    test_col = test_col.where(test_col.isin(valid_values), other='None')
                    test_df[col] = le.transform(test_col)
        return df, test_df
    def quantile_transform_column_wise(self, df_temp: pd.DataFrame, target_col: str =
"", output_distribution: str = "uniform") -> pd.DataFrame:
        """
        Transformira vrijednosti u DataFrameu pomoću kvantilne distribucije (npr.
uniform)
        za sve stupce osim target_col.

        Parameters
        ----------
        df_temp : pd.DataFrame
            DataFrame koji se transformira.
        target_col : str, optional
            Naziv ciljnog stupca koji se ne transformira, by default "".
        output_distribution : str, optional
            Izlazna distribucija za transformaciju (npr. "uniform"), by default "uniform".

        Returns
        -------
        pd.DataFrame
            Transformirani DataFrame.
        """
        df_temp = df_temp.copy()
        # Odredi broj kvantila kao minimum između 1000 i broja redaka u df_temp
```

```
        n_samples = min(1000, len(df_temp))
        # Odaberi sve stupce osim target_col
        cols_to_transform = [col for col in df_temp.columns if col != target_col]

        if cols_to_transform:
            qt = preprocessing.QuantileTransformer(random_state=1,
n_quantiles=n_samples, output_distribution=output_distribution)
            # Vectorizirano transformiranje svih odabranih stupaca odjednom
            df_temp[cols_to_transform] = qt.fit_transform(df_temp[cols_to_transform])

        return df_temp
    class EstimatorSelection:
        def __init__(self, models, params):
            self.models = models
            self.params = params
            self.keys = models.keys()
            self.grid_searches = {}
            self.best_scores = {}
            self.best_estimator = {}
            self.best_params = {}
            self.datetime = datetime.now().strftime("%d_%m_%Y_%H_%M_%S")

        def write_to_excel(self, df : pd.DataFrame, filename : str, sheetname : str):
            """
            Zapisuje dataframe u excel datoteku s danim imenom i nazivom lista.
            Ako datoteka već postoji, prepisuje je.
            Ako je sheetname već prisutan, prepisuje ga.
            Ako datoteka ne postoji, stvara novu.
            Parameters
            ----------
            df : pd.DataFrame
                Dataframe koji se zapisuje excel
            filename : str
                Naziv datoteke u koji se zapisuje
```

```
    sheetname : str
        Radni list u koji se zapisuje
    """
    writer = pd.ExcelWriter(filename, engine='xlsxwriter')
    df.to_excel(writer, sheet_name=sheetname, index=False)
    writer.save()

def grid_search_cv_fit(self, X, y, **grid_kwargs):
    results_list = []
    for key in self.keys:
        print('Pokretanje GridSearchCV-a za %s.' % key)
        start_time = time.time()
        model = self.models[key]
        params = self.params[key]
        grid_search = GridSearchCV(model, params, **grid_kwargs)
        grid_search.fit(X, y)
        execution_time = time.time() - start_time
        self.grid_searches[key] = grid_search
        self.best_estimator[key] = grid_search.best_estimator_
        self.best_scores[key] = grid_search.best_score_
        self.best_params[key] = grid_search.best_params_

        # Dodajemo sve rezultate u listu za DataFrame
        result_entry = {
            'Model': key,
            'Najbolji rezultat': grid_search.best_score_,
            'Najbolji parametri': grid_search.best_params_,
            'Vrijeme izvršavanja (s)': round(execution_time, 2),
            'Najbolji estimator': str(grid_search.best_estimator_),
            'Broj kombinacija': len(grid_search.cv_results_['params']),
            'Status': 'Završeno'
        }

        results_list.append(result_entry)
```

```
            print(" ---- START ---- Rezultati iz Grid Search CV-a %s." % key )
            print("\n Najbolji procjenitelj među SVIM pretraženim:\n",grid_search.best_estimator_)
            print("\n Najbolji rezultat među SVIM pretraženim parametrima:\n",grid_search.best_score_)
            print("\n Najbolji parametri među SVIM pretraženim parametrima:\n",grid_search.best_params_)
            print(" ---- KRAJ ---- Rezultati iz Grid Search CV-a %s." % key )
        print('Done.')
        results_df = pd.DataFrame(results_list)
        results_df = results_df.sort_values(
            by=['Najbolji rezultat', 'Vrijeme izvršavanja (s)'],
            ascending=[False, True]
        ).reset_index(drop=True)

        results_df = results_df[[
            'Model',
            'Najbolji rezultat',
            'Vrijeme izvršavanja (s)',
            'Najbolji estimator',
            'Najbolji parametri',
            'Broj kombinacija',
            'Status'
        ]]
        print("\nSažetak svih rezultata:")
        display(results_df)
        results_df.to_csv('grid_search_rezultati.csv', index=False)
        self.write_to_excel(results_df, self.datetime+"_score_summary.xlsx", "GridSearchCV")

    def fit(self, X, y, **grid_kwargs):
        for key in self.keys:
            print('Pokretanje GridSearchCV-a za %s.' % key)
            model = self.models[key]
```

```
            params = self.params[key]
            grid_search = GridSearchCV(model, params, **grid_kwargs)
            grid_search.fit(X, y)
            self.grid_searches[key] = grid_search
            self.best_scores[key] = grid_search.best_estimator_.score(X, y)
        print('Done.')
        joblib.dump(self.grid_searches, self.datetime+'_grid_searches.pkl')
        joblib.dump(self.best_scores, self.datetime+'_best_scores.pkl')
        json_object = json.dumps(self.best_scores, indent=4)
        with open(self.datetime+"_best_scores.json", "w") as outfile:
            outfile.write(json_object)

    def score_summary(self, sort_by='best_score'):
        frames = []
        for key in self.keys:
            frame = pd.DataFrame()
            frame = frame.filter(regex='^(?!.*param_).*$')
            frame['best_estimator'] = self.best_estimator[key]
            frame['best_score'] = self.best_scores[key]
            frame['best_params'] = self.best_params[key]
            frames.append(frame)
        df = pd.concat(frames)
        df = df.sort_values([sort_by], ascending=False)
        df = df.reset_index()
        #df = df.drop(['rank_test_score', 'index'], 1)
        self.write_to_excel(df, self.datetime+"_score_summary.xlsx", "GridSearchCV")
        columns = df.columns.tolist()
        df = df[columns]
        df.to_pickle(self.datetime+"_score_summary.pkl")
        df.to_csv(self.datetime+"_score_summary.csv",index=False)
        return df


    def score_summary_old(self, sort_by='mean_test_score'):
```

```
        frames = []
        for name, grid_search in self.grid_searches.items():
            frame = pd.DataFrame(grid_search.cv_results_)
            frame = frame.filter(regex='^(?!.*param_).*$')
            frame['estimator'] = len(frame)*[name]
            frame['mean_fit_time'] = mean_fit_time = grid_search.cv_results_['mean_fit_time']
            frame['mean_score_time'] = mean_score_time = grid_search.cv_results_['mean_score_time']
            frame['n_splits'] = n_splits = grid_search.n_splits_ #number of splits of training data
            frame['n_iter'] = n_iter = pd.DataFrame(grid_search.cv_results_).shape[0] #Iterations per split
            frame['elapsed_time'] = np.mean(mean_fit_time + mean_score_time) * n_splits * n_iter
            frames.append(frame)
        df = pd.concat(frames)

        df = df.sort_values([sort_by], ascending=False)
        df = df.reset_index()
        #df = df.drop(['rank_test_score', 'index'], 1)

        columns = df.columns.tolist()
        columns.remove('estimator')
        columns = ['estimator']+columns
        df = df[columns]
        df.to_pickle(self.datetime+"_score_summary.pkl")
        df.to_csv(self.datetime+"_score_summary.csv",index=False)
        return df
    def clf_models_and_params():
        """
        Usage:
            helper1 = EstimatorSelectionHelper(models1, params1)
            helper1.fit(X, y, scoring='f1', n_jobs=2)
            helper1.score_summary()
```

```
"""
modelsClf = {
'DecisionTreeClassifier': DecisionTreeClassifier(),
'GaussianNB': GaussianNB(), # Naive Bayes Classifier
'RandomForestClassifier': RandomForestClassifier(),
'SVC': SVC(),
'GradientBoostingClassifier': GradientBoostingClassifier(),
'XGBClassifier': XGBClassifier(),
'KNeighborsClassifier': KNeighborsClassifier(),
'MLPClassifier': MLPClassifier() # ANN
}

paramsClf = {
   'DecisionTreeClassifier': {
      'criterion': ['gini', 'entropy'],
      'max_depth': [None, 10, 20, 30],
      'max_features': [None, 'sqrt', 'log2'],
      'class_weight': [None, 'balanced']
   },
   'GaussianNB': {'priors': [None], 'var_smoothing': [0.00000001, 0.000000001,
0.0000000001]},
   'RandomForestClassifier': {
      'n_estimators': [10, 50, 100],
      'criterion': ['gini', 'entropy'],
      'max_depth': [None, 10, 20, 30],
      'max_features': [None, 'sqrt', 'log2'],
      'class_weight': [None, 'balanced']
   },
   'SVC':  {'kernel': ['linear', 'rbf'], 'C': [0.1, 1, 10], 'gamma': ['scale', 'auto', 0.1, 1,
10]},
   'GradientBoostingClassifier': {
      'loss': ['deviance', 'exponential'],
      'learning_rate': [0.01, 0.1, 0.2, 0.3],
```

```
            'n_estimators': [10, 50, 100],
            'criterion': ['friedman_mse', 'mse', 'mae'],
            'max_depth': [None, 10, 20, 30],
            'max_features': [None, 'sqrt', 'log2']
        },
        'XGBClassifier': {
            'criterion': ['gini', 'entropy'],
            'max_depth': [None, 10, 20, 30],
            'max_features': [None, 'sqrt', 'log2'],
            'class_weight': [None, 'balanced']
        },
        'KNeighborsClassifier': {
            'n_neighbors': [3, 5, 7, 9, 11],
            'weights': ['uniform', 'distance'],
            'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
            'p': [1, 2],
            'metric': ['euclidean', 'manhattan', 'minkowski']
        },
        'MLPClassifier': {
            'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 100)],
            'activation': ['logistic', 'tanh', 'relu'],
            'solver': ['adam'],
            'alpha': [0.0001, 0.05],
            'learning_rate': ['constant',  'adaptive'],
            'max_iter': [500]
        }
    }
    return modelsClf, paramsClf
def multi_clf_models_and_params():
    """
    Usage:
        helper1 = EstimatorSelectionHelper(models1, params1)
        helper1.fit(X, y, scoring='f1', n_jobs=2)
        helper1.score_summary()
```

```
"""
modelsClf = {
'DecisionTreeClassifier': DecisionTreeClassifier(),
'GaussianNB': GaussianNB(), # Naive Bayes Classifier
'RandomForestClassifier': RandomForestClassifier(),
'SVC': SVC(),
'XGBClassifier': XGBClassifier(),
'KNeighborsClassifier': KNeighborsClassifier(),
'MLPClassifier': MLPClassifier() # ANN
}

paramsClf = {
   'DecisionTreeClassifier': {
      'criterion': ['gini', 'entropy'],
      'max_depth': [None, 10, 20, 30],
      'max_features': [None, 'sqrt', 'log2'],
      'class_weight': [None, 'balanced']
   },
   'GaussianNB': {'priors': [None], 'var_smoothing': [0.00000001, 0.000000001,
0.0000000001]},
   'RandomForestClassifier': {
      'n_estimators': [10, 50, 100],
      'criterion': ['gini', 'entropy'],
      'max_depth': [None, 10, 20, 30],
      'max_features': [None, 'sqrt', 'log2'],
      'class_weight': [None, 'balanced']
   },
   'SVC':  {'kernel': ['linear', 'rbf'], 'C': [0.1, 1, 10], 'gamma': ['scale', 'auto', 0.1, 1,
10]},
   'XGBClassifier': {
      'criterion': ['gini', 'entropy'],
      'max_depth': [None, 10, 20, 30],
      'max_features': [None, 'sqrt', 'log2'],
```

```
            'class_weight': [None, 'balanced']
        },
        'KNeighborsClassifier': {
            'n_neighbors': [3, 5, 7, 9, 11],
            'weights': ['uniform', 'distance'],
            'algorithm': ['auto', 'ball_tree', 'kd_tree', 'brute'],
            'p': [1, 2],
            'metric': ['euclidean', 'manhattan', 'minkowski']
        },
        'MLPClassifier': {
            'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 100)],
            'activation': ['logistic', 'tanh', 'relu'],
            'solver': ['adam'],
            'alpha': [0.0001, 0.05],
            'learning_rate': ['constant',  'adaptive'],
            'max_iter': [500]
        }
    }
    return modelsClf, paramsClf


def reg_models_and_params():
    """
    Usage:
        helper1 = EstimatorSelectionHelper(models1, params1)
        helper1.fit(X, y, scoring='f1', n_jobs=2)
        helper1.score_summary()

    """
    modelsReg = {
    'LinearRegression': LinearRegression(),
    'Lasso': Lasso(),
    'Ridge': Ridge(),
    'ElasticNet': ElasticNet(),
    'SVR': SVR(),
```

```
        'KNeighborsRegressor': KNeighborsRegressor(),
        'XGBRegressor': XGBRegressor(),
        'MLPRegressor': MLPRegressor(),
        'DecisionTreeRegressor': DecisionTreeRegressor(),
        'RandomForestRegressor': RandomForestRegressor(),
        'GradientBoostingRegressor': GradientBoostingRegressor()
        }


        paramsReg = {
            'LinearRegression': {'fit_intercept': [True], 'copy_X': [True], 'n_jobs': [None], 'positive': [False] },
            'Lasso': {'alpha': [1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100, 1000, 10000] },
            'Ridge': {'alpha': [1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100, 1000, 10000] },
            'ElasticNet': {'alpha': [1e-5, 1e-4, 1e-3, 1e-2, 1e-1, 1, 10, 100, 1000, 10000], 'l1_ratio': np.linspace(0, 1, 11)},
            'SVR': { 'kernel': ['linear', 'rbf'], 'C': [0.1, 1, 10], 'gamma': ['scale', 'auto', 0.1, 1, 10]},
            'KNeighborsRegressor': {
               'n_neighbors': [3, 5, 7, 9, 11],
               'weights': ['uniform', 'distance'],
               'metric': ['euclidean', 'manhattan', 'minkowski']
            },
            'XGBRegressor': {
               'n_estimators': [100, 400, 800],
               'learning_rate': [0.05, 0.1, 0.20],
               'max_depth': [3, 6, 9],
               'min_child_weight': [1, 10, 100],
               'tree_method': ['gpu_hist']
            },
            'MLPRegressor': {
              'hidden_layer_sizes': [(50,), (100,), (50, 50), (100, 100)],
              'activation': ['logistic', 'tanh', 'relu'],
              'solver': ['adam'],
              'alpha': [0.0001, 0.05],
              'learning_rate': ['constant', 'adaptive'],
```

```
            'max_iter': [500]
        },
        'DecisionTreeRegressor': {
            'max_depth': [None, 10, 20, 30],
            'max_features': [None, 'auto', 'sqrt', 'log2'],
            'ccp_alpha': [0.0, 0.1, 0.2, 0.3]
        },
        'RandomForestRegressor': {
            'n_estimators': [10, 50, 100],
            'max_depth': [None, 10, 20, 30],
            'max_features': [None, 'auto', 'sqrt', 'log2'],
            'ccp_alpha': [0.0, 0.1, 0.2, 0.3]
        },
        'GradientBoostingRegressor': {
            'n_estimators': [10, 50, 100],
            'learning_rate': [0.001, 0.01, 0.1],
            'subsample': [0.5, 0.75, 1.0],
            'max_depth': [3, 10, 20],
            'max_features': [None, 'auto', 'sqrt', 'log2'],
            'alpha': [0.1, 0.5, 0.9]
        }
    }
    return modelsReg, paramsReg


class FeatureSelector:
    def __init__(self, top_percentage=20):
        self.top_percentage = top_percentage
        self.selected_features = []


    def rank_features_with_kfold(self, X, y, model, task_type='classification'):
        """
        Univerzalna metoda za rangiranje značajki koristeći KFold (StratifiedKFold za klasifikaciju)
        i zadani model.
```

```
        """
        if task_type == 'classification':
            kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
        else:
            kf = KFold(n_splits=5, shuffle=True, random_state=42)

        feature_importances = np.zeros(X.shape[1])

        for train_idx, test_idx in kf.split(X, y):
            X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
            y_train, y_test = y[train_idx], y[test_idx]
            model.fit(X_train, y_train)
            if hasattr(model, "feature_importances_"):
                feature_importances += model.feature_importances_
            elif hasattr(model, "coef_"):
                feature_importances += np.abs(model.coef_).ravel()

        feature_importances /= kf.get_n_splits()
        return pd.DataFrame({
            'feature': X.columns,
            'importance': feature_importances
        }).sort_values(by='importance', ascending=False)

    def rank_features_random_forest(self, X, y, task_type='classification'):
        if task_type == 'classification':
            model = RandomForestClassifier(random_state=42, n_jobs=-1)
        else:
            model = RandomForestRegressor(random_state=42, n_jobs=-1)
        return self.rank_features_with_kfold(X, y, model, task_type)

    def rank_features_xgboost(self, X, y, task_type='classification'):
        if task_type == 'classification':
            model = XGBClassifier(random_state=42, use_label_encoder=False,
eval_metric='logloss', n_jobs=-1)
```

```
        else:
            model = XGBRegressor(random_state=42, n_jobs=-1)
        return self.rank_features_with_kfold(X, y, model, task_type)

    #def rank_features_dt(self, X, y):
        # Ova metoda je trenutno definirana samo za klasifikaciju
        #return self.rank_features_with_kfold(X, y, DecisionTreeClassifier(max_depth=10), task_type='classification')

    def rank_features_mutual_info(self, X, y, task_type='classification'):
        """
        Rangiranje značajki koristeći Mutual Information s (Stratified)KFold.
        """
        if task_type == 'classification':
            kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
            mutual_info = np.zeros(X.shape[1])
            for train_idx, test_idx in kf.split(X, y):
                X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
                y_train, y_test = y[train_idx], y[test_idx]
                mutual_info += mutual_info_classif(X_train, y_train, random_state=42)
        else:
            kf = KFold(n_splits=5, shuffle=True, random_state=42)
            mutual_info = np.zeros(X.shape[1])
            for train_idx, test_idx in kf.split(X, y):
                X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
                y_train, y_test = y[train_idx], y[test_idx]
                mutual_info += mutual_info_regression(X_train, y_train, random_state=42)
        mutual_info /= kf.get_n_splits()
        return pd.DataFrame({
            'feature': X.columns,
            'importance': mutual_info
        }).sort_values(by='importance', ascending=False)

    def rank_features_rfe(self, X, y, task_type='classification'):
```

```
        """
        Rangiranje značajki koristeći Rekurzivnu Eliminaciju Značajki (RFE) s (Stratified)KFold.
        """
        if task_type == 'classification':
            kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
            estimator = LogisticRegression(random_state=42, max_iter=1000)
        else:
            kf = KFold(n_splits=5, shuffle=True, random_state=42)
            estimator = LinearRegression()

        feature_importances = np.zeros(X.shape[1])
        for train_idx, test_idx in kf.split(X, y):
            X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
            y_train, y_test = y[train_idx], y[test_idx]
            rfe = RFE(estimator, n_features_to_select=1)
            rfe.fit(X_train, y_train)
            # Negiramo rangiranje jer su manji brojevi bolji (1 je najbolji)
            feature_importances += -rfe.ranking_
        feature_importances /= kf.get_n_splits()
        return pd.DataFrame({
            'feature': X.columns,
            'importance': feature_importances
        }).sort_values(by='importance', ascending=False)

    def voting_selection(self, X, y, task_type='classification'):
        """
        Odabir značajki koristeći sustav glasovanja temeljen na najboljim značajkama iz svih metoda.
        :param X: DataFrame sa značajkama.
        :param y: Ciljana varijabla.
        :param task_type: 'classification' ili 'regression'.
        :return: Smanjeni skup podataka s odabranim značajkama.
        """
```

```
        # Rangiramo značajke koristeći sve metode prema tipu zadatka
        rf_rankings = self.rank_features_random_forest(X, y, task_type=task_type)
        xgb_rankings = self.rank_features_xgboost(X, y, task_type=task_type)
        mi_rankings = self.rank_features_mutual_info(X, y, task_type=task_type)
        rfe_rankings = self.rank_features_rfe(X, y, task_type=task_type)

        # Definiramo prag za odabir top postotka značajki
        n_top_features = int(len(rf_rankings) * (self.top_percentage / 100))

        # Inicijaliziramo broj glasova za svaku značajku
        votes = pd.Series(0, index=X.columns)

        # Dodajemo glasove za top značajke iz svake metode
        for rankings in [rf_rankings, xgb_rankings, mi_rankings, rfe_rankings]:
            top_features = rankings.head(n_top_features)['feature']
            votes[top_features] += 1

        # Odabir značajki s najviše glasova
        selected_features = votes.sort_values(ascending=False).head(n_top_features).index.tolist()
        self.selected_features = selected_features

        return X[self.selected_features]

    def knapsack_top_features(self, X, y, capacity=10, task_type='classification', model_choice='rf'):
        """
        Odabir značajki koristeći Knapsack algoritam na temelju njihove važnosti i varijance.
        :param X: DataFrame sa značajkama.
        :param y: Ciljana varijabla.
        :param capacity: Maksimalna dopuštena težina za knapsack.
        :param task_type: 'classification' ili 'regression'.
        :param model_choice: 'rf' (RandomForest) ili 'xgb' (XGBoost).
        :return: Smanjeni skup podataka s odabranim značajkama.
```

```
    """
    # Odabir KFold metode i konstruktora modela prema tipu zadatka
    if task_type == 'classification':
        kf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
        if model_choice == 'rf':
            model_constructor = RandomForestClassifier
        elif model_choice == 'xgb':
            model_constructor = XGBClassifier
        else:
            raise ValueError(f"Nepoznat izbor modela za klasifikaciju: {model_choice}")
    else:
        kf = KFold(n_splits=5, shuffle=True, random_state=42)
        if model_choice == 'rf':
            model_constructor = RandomForestRegressor
        elif model_choice == 'xgb':
            model_constructor = XGBRegressor
        else:
            raise ValueError(f"Nepoznat izbor modela za regresiju: {model_choice}")

    feature_importances = np.zeros(X.shape[1])

    # Iteracija kroz preklopove i agregacija značajki
    for train_idx, test_idx in kf.split(X, y):
        X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
        y_train, y_test = y[train_idx], y[test_idx]

        # Posebno postavljanje parametara za XGBClassifier
        if task_type == 'classification' and model_choice == 'xgb':
            model = model_constructor(random_state=42, use_label_encoder=False, eval_metric='logloss', n_jobs=-1)
        else:
            model = model_constructor(random_state=42, n_jobs=-1)
```

```
            model.fit(X_train, y_train)
            feature_importances += model.feature_importances_

        feature_importances /= kf.get_n_splits()

        # Kombinacija značajki, važnosti i varijance
        rankings = pd.DataFrame({
            'feature': X.columns,
            'importance': feature_importances,
            'variance': X.var()
        })

        # Normalizacija važnosti i varijance
        rankings['normalized_importance'] = rankings['importance'] / rankings['importance'].sum()
        rankings['normalized_variance'] = rankings['variance'] / rankings['variance'].sum()

        # Skaliranje varijance kako bi se uklopila u kapacitet
        rankings['scaled_variance'] = rankings['normalized_variance'] * capacity

        # Kreiramo listu stavki za Knapsack (vrijednost = normalized_importance, težina = scaled_variance)
        items = [
            (row['normalized_importance'], row['scaled_variance'], row['feature'])
            for _, row in rankings.iterrows()
        ]

        # Inicijalizacija DP tablice za Knapsack algoritam
        n = len(items)
        dp = [[0] * (capacity + 1) for _ in range(n + 1)]

        for i in range(1, n + 1):
            value, weight, _ = items[i - 1]
            for w in range(1, capacity + 1):
```

```
                if weight <= w:
                    dp[i][w] = max(dp[i - 1][w], dp[i - 1][int(w - weight)] + value)
                else:
                    dp[i][w] = dp[i - 1][w]

        # Backtracking za pronalaženje odabranih značajki
        selected_features = []
        w = capacity
        for i in range(n, 0, -1):
            if dp[i][w] != dp[i - 1][w]:
                _, weight, feature = items[i - 1]
                selected_features.append(feature)
                w -= int(weight)

        self.selected_features = selected_features
        return X[self.selected_features]
    class Evaluation:
        def __init__(self, df : pd.DataFrame, test_df : pd.DataFrame, target_var : str, estimator: BaseEstimator, task_type : str ='classification', x_feature_selector: np.ndarray = None):
            self.target_var = target_var
            self.task_type = task_type
            self.df = df
            self.test_df = test_df
            self.x_feature_selector = x_feature_selector
            self.estimator = estimator

        def evaluate(self):
            if self.task_type == 'classification':
                # Pretpostavljamo da su df i test_df već učitani
                X = None
                test_data = None

                y = self.df[self.target_var].values
```

```
        if self.x_feature_selector is not None:
            X = self.df[self.x_feature_selector].values
            test_data = self.test_df[self.x_feature_selector].values
        else:
            # Ako x_feature_selector nije postavljen, koristimo sve značajke osim ciljne varijable
            X = self.df.drop(self.target_var).values
            test_data = self.test_df.values

        # Inicijalizacija mjerenja vremena
        start_train = time.time()
        skf = StratifiedKFold(n_splits=5)
        lr = self.estimator
        scores = []
        all_predictions = []
        prediction_time = 0.0  # Dodajemo varijablu za vrijeme predikcije

        for train_index, test_index in skf.split(X, y):
            # Treniranje modela
            lr.fit(X[train_index], y[train_index])

            # Mjerenje vremena predikcije za ovaj fold
            start_pred = time.time()
            fold_pred = lr.predict(test_data)
            prediction_time += time.time() - start_pred

            all_predictions.append(fold_pred)
            scores.append(lr.score(X[test_index], y[test_index]))

        # Računanje finalne predikcije
        final_prediction = np.round(np.mean(all_predictions, axis=0)).mean()  # Prosjek svih predikcija

        finish_train = time.time()
```

```
            print("\nRezultat (Score):", np.mean(scores))
            print("Prosječna finalna predikcija:", final_prediction)
            print(f"Ukupno vrijeme treniranja: {finish_train - start_train} sekundi")
            print(f"Ukupno vrijeme predikcije: {prediction_time} sekundi")
        else:
            # Pretpostavljamo da su df i test_df već učitani
            X = None
            test_data = None

            y = self.df[self.target_var].values
            if self.x_feature_selector is not None:
                X = self.df[self.x_feature_selector].values
                test_data = self.test_df[self.x_feature_selector].values
            else:
                # Ako x_feature_selector nije postavljen, koristimo sve značajke osim ciljne
varijable
                X = self.df.drop(self.target_var).values
                test_data = self.test_df.values

            # Inicijalizacija mjerenja vremena
            start_train = time.time()
            kf = KFold(n_splits=5, shuffle=True, random_state=42)
            lr = self.estimator
            rmse_scores = []
            all_predictions = []
            prediction_time = 0.0  # Varijabla za vrijeme predikcije

            for train_index, test_index in kf.split(X, y):
                # Treniranje modela
                lr.fit(X[train_index], y[train_index])

                # Izračun predikcija na validacijskom skupu i RMSE
                y_val_pred = lr.predict(X[test_index])
```

```
            rmse = np.sqrt(mean_squared_error(y[test_index], y_val_pred))
            rmse_scores.append(rmse)

            # Mjerenje vremena predikcije za ovaj fold na vanjskom test skupu
            start_pred = time.time()
            fold_pred = lr.predict(test_data)
            prediction_time += time.time() - start_pred

            all_predictions.append(fold_pred)

        # Računanje finalne predikcije: prosjek svih predikcija na test_data
        final_prediction = np.round(np.mean(all_predictions, axis=0)).mean()

        finish_train = time.time()

        print("\nProsječni RMSE:", np.mean(rmse_scores))
        print("Prosječna finalna predikcija:", final_prediction)
        print(f"Ukupno vrijeme treniranja: {finish_train - start_train} sekundi")
        print(f"Ukupno vrijeme predikcije: {prediction_time} sekundi")
```

```
class DimensionalityReductionTechniques:
    '''
    Klasa za primjenu PCA, t-SNE i UMAP na skup podataka koristeći KFold ili StratifiedKFold.
    Primjenjuje se PCA, t-SNE i UMAP na skup podataka koristeći KFold ili StratifiedKFold.
    Služi za analizu generalizacijskih sposobnosti vlastitio razvijenih metoda iz komponente FeatureSelector usporedbom s PCA, t-SNE, UMAP i AutoEncoders.
    '''
    def __init__(self, n_components: int, target_col: str, df: pd.DataFrame, test_df: pd.DataFrame, estimator: BaseEstimator, task_type: str = 'classification'):
        self.n_components = n_components
        self.target_col = target_col
        self.df = df
        self.test_df = test_df
```

```
        self.estimator = estimator
        self.task_type = task_type


    def pca(self):
        """
        Primjena PCA na skup podataka koristeći StratifiedKFold za klasifikacijske zadatke
        te KFold za regresijske zadatke.
        """
        # Podaci
        y = self.df[self.target_col].values
        X = self.df.drop(columns=self.target_col).values
        test_data = self.test_df.values  # Pretpostavka da postoji test_df

        # Odabir odgovarajuće metode folda (CV)
        if self.task_type == 'classification':
            cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
        else:
            cv = KFold(n_splits=5, shuffle=True, random_state=42)

        model = self.estimator
        scores = []
        all_preds = []
        total_train_time = 0
        total_pred_time = 0

        # Iteracija kroz foldove
        for train_idx, val_idx in cv.split(X, y):
            start_fold = time.time()
            pca = PCA(n_components=self.n_components)
            X_train = X[train_idx]

            # Fit i transformacija
            X_train_pca = pca.fit_transform(X_train)
```

```
            test_data_pca = pca.transform(test_data)

            # Treniranje
            model.fit(X_train_pca, y[train_idx])
            train_time = time.time() - start_fold
            total_train_time += train_time

            # Evaluacija
            X_val_pca = pca.transform(X[val_idx])
            scores.append(model.score(X_val_pca, y[val_idx]))

            # Predikcija
            start_pred = time.time()
            preds = model.predict(test_data_pca)
            total_pred_time += time.time() - start_pred
            all_preds.append(preds)

        # Finalna predikcija (prosjek)
        final_pred = np.round(np.mean(all_preds, axis=0)).mean()

        print(f"PCA Rezultat: {np.mean(scores)}")
        print(f"Finalna predikcija: {final_pred}")
        print(f"Ukupno vrijeme treniranja: {total_train_time}s")
        print(f"Ukupno vrijeme predikcije: {total_pred_time}s")
        return final_pred, np.mean(scores), total_train_time, total_pred_time

    def tsne(self):
        """
        Primjena t-SNE na skup podataka koristeći StratifiedKFold za klasifikacijske zadatke
        te KFold za regresijske zadatke.
        """
        # Podaci
        y = self.df[self.target_col].values
```

```
        X = self.df.drop(columns=self.target_col).values
        test_data = self.test_df.values  # Pretpostavka da postoji test_df

        # Postavke
        # Odabir odgovarajuće metode folda (CV)
        if self.task_type == 'classification':
            cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
        else:
            cv = KFold(n_splits=5, shuffle=True, random_state=42)

        model = self.estimator
        all_preds = []
        total_train_time = 0
        total_pred_time = 0

        for train_idx, val_idx in cv.split(X, y):
            start_fold = time.time()
            X_train = X[train_idx]

            # Kombinacija podataka za t-SNE
            combined = np.vstack((X_train, test_data))
            tsne = TSNE(n_components=self.n_components, random_state=0,
method='exact')
            combined_tsne = tsne.fit_transform(combined)

            # Podjela podataka
            X_train_tsne = combined_tsne[:len(X_train)]
            test_tsne = combined_tsne[len(X_train):]

            # Treniranje
            model.fit(X_train_tsne, y[train_idx])
            total_train_time += time.time() - start_fold

            # Predikcija
```

```
            start_pred = time.time()
            preds = model.predict(test_tsne)
            total_pred_time += time.time() - start_pred
            all_preds.append(preds)

        # Finalna predikcija
        final_pred = np.round(np.mean(all_preds, axis=0)).mean()

        print(f"t-SNE Finalna predikcija: {final_pred}")
        print(f"Ukupno vrijeme treniranja: {total_train_time}s")
        print(f"Ukupno vrijeme predikcije: {total_pred_time}s")
        return final_pred, total_train_time, total_pred_time


    def umap(self):
        """
        Primjena UMAP na skup podataka koristeći StratifiedKFold za klasifikacijske zadatke
        te KFold za regresijske zadatke.
        """
        # Podaci
        y = self.df[self.target_col].values
        X = self.df.drop(columns=self.target_col).values
        test_data = self.test_df.values  # Pretpostavka da postoji test_df

        # Postavke
        # Odabir odgovarajuće metode folda (CV)
        if self.task_type == 'classification':
            cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
        else:
            cv = KFold(n_splits=5, shuffle=True, random_state=42)

        model = self.estimator
        scores = []
        all_preds = []
```

```
total_train_time = 0
total_pred_time = 0

for train_idx, val_idx in cv.split(X, y):
    # Inicijalizacija modela za svaki fold
    umap_model = umap.UMAP(
        n_components=self.n_components,
        n_neighbors=15,
        min_dist=0.2,
        metric='cosine',
        random_state=42
    )

    start_fold = time.time()

    # Podjela podataka
    X_train = X[train_idx]
    y_train = y[train_idx]

    # UMAP transformacija
    X_train_umap = umap_model.fit_transform(X_train)
    test_umap = umap_model.transform(test_data)

    # Treniranje modela
    model.fit(X_train_umap, y_train)
    train_time = time.time() - start_fold
    total_train_time += train_time

    # Evaluacija
    X_val_umap = umap_model.transform(X[val_idx])
    scores.append(model.score(X_val_umap, y[val_idx]))

    # Predikcija
    start_pred = time.time()
```

```
            preds = model.predict(test_umap)
            total_pred_time += time.time() - start_pred
            all_preds.append(preds)

        # Agregacija rezultata
        final_pred = np.round(np.mean(all_preds, axis=0)).mean()

        print(f"\nUMAP + Random Forest Rezultat: {np.mean(scores)}")
        print(f"Finalna predikcija: {final_pred}")
        print(f"Ukupno vrijeme treniranja: {total_train_time}s")
        print(f"Ukupno vrijeme predikcije: {total_pred_time}s")
        return final_pred, np.mean(scores), total_train_time, total_pred_time


    def autoencoder(self):
        """
        Primjena AutoEncoders na skup podataka koristeći StratifiedKFold za klasifikacijske zadatke
        te KFold za regresijske zadatke.
        """
        # Podaci
        y = self.df[self.target_col].values
        X = self.df.drop(columns=self.target_col).values
        test_data = self.test_df.values  # Pretpostavka da postoji test_df

        # Postavke
        # Odabir odgovarajuće metode folda (CV)
        if self.task_type == 'classification':
            cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
        else:
            cv = KFold(n_splits=5, shuffle=True, random_state=42)

        model = self.estimator
        scores = []
        all_preds = []
```

```
total_train_time = 0
total_pred_time = 0

# Autoencoder arhitektura
input_dim = X.shape[1]
encoding_dim = self.n_components  # Dimenzija kodiranja

for train_idx, val_idx in cv.split(X, y):
    start_fold = time.time()
    X_train = X[train_idx]

    # Skaliranje
    #scaler = StandardScaler()
    #X_train_scaled = scaler.fit_transform(X_train)
    #test_scaled = scaler.transform(test_data)
    X_train_scaled = X_train
    test_scaled = test_data




    # Autoencoder - tensorflow
    input_layer = Input(shape=(input_dim,))
    encoder = Dense(encoding_dim, activation='relu')(input_layer)
    decoder = Dense(input_dim, activation='sigmoid')(encoder)
    autoencoder = Model(input_layer, decoder)
    autoencoder.compile(optimizer='adam', loss='mse')

    # Treniranje autoencodera
    autoencoder.fit(X_train_scaled, X_train_scaled,
                    epochs=50,
                    batch_size=32,
                    verbose=0)

    # Autoencoder - tensorflow
```

```
    # Enkodiranje - transformacija
    encoder_model = Model(input_layer, encoder)
    X_train_enc = encoder_model.predict(X_train_scaled)
    test_enc = encoder_model.predict(test_scaled)

    # Treniranje modela
    model.fit(X_train_enc, y[train_idx])
    total_train_time += time.time() - start_fold

    # Evaluacija
    X_val_scaled = X[val_idx] #scaler.transform()
    X_val_enc = encoder_model.predict(X_val_scaled)
    scores.append(model.score(X_val_enc, y[val_idx]))

    # Predikcija
    start_pred = time.time()
    preds = model.predict(test_enc)
    total_pred_time += time.time() - start_pred
    all_preds.append(preds)

# Finalna predikcija
final_pred = np.round(np.mean(all_preds, axis=0)).mean()

print(f"Autoencoder Rezultat: {np.mean(scores)}")
print(f"Finalna predikcija: {final_pred}")
print(f"Ukupno vrijeme treniranja: {total_train_time}s")
print(f"Ukupno vrijeme predikcije: {total_pred_time}s")
return final_pred, np.mean(scores), total_train_time, total_pred_time
```

Životopis

Saša Mitrović od ožujka 2013. godine na Ekonomski fakultet u Osijeku izabran je u naslovno suradničko zvanje asistenta u području društvenih znanosti, znanstveno polje informacijske znanosti.

Radi kao asistent na kolegijima Informatike, Razvoj poslovnih aplikacija, te Razvoj distribuiranih i web aplikacija, Baza podataka i poslovnih procesa, Kvantitativnih metoda u poslovnom odlučivanju i Upravljanja odnosima s potrošačima , a od listopada 2018. godine je izabran u suradničko zvanje asistenta u punom radnom vremenu.

Saša Mitrović je i Microsoft Certified Professional za razvoj Windows aplikacija od 2012.

Njegove druge vještine uključuju aktivno znanje engleskog jezika u govoru i pismu, te napredno korištenje brojnih tehnologija kao što su C#, ASP.NET, Python, R, JavaScript, SQL Server, PostgreSQL i MySQL baze podataka.

Kao član organizacijskog odbora sudjelovao je i u organizaciji međunarodne znanstvene konferencije Interdisciplinary Management Research (od 2019. do danas). Bio je istraživač na znanstvenom projektu Hrvatske zaklade za znanost - Metodološki okvir za učinkovito upravljanje energijom s pomoću inteligentne podatkovne analitike, br. IP-2016-06-8350.

Popis radova:

1. **Mitrović, Saša** Razvoj CRISP-DM utemeljenog AutoML sustava za klasifikaciju i regresijsku analizu potencijala obnovljivih izvora energije u zemljama Europske unije // PLIN 2025 - Zbornik radova 16. međunarodnog skupa o prirodnom plinu, toplini i vodi / Raos, Pero; Kozak, Dražan; Raos, Marija et al. (ur.). Slavonski Brod: Sveučilište u Slavonskom Brodu, 2025. str. 76-87 (predavanje, cjeloviti rad (in extenso), znanstveni)
2. **Mitrović, Saša;** Neven Vrček Methodology for the comparative analysis of PCA and Autoencoders in dimensionality reduction: impact on classification accuracy and computational efficiency // Proceedings of the 36th Central European Conference on Information and Intelligent Systems // Varaždin: Faculty of Organization and Informatics, University of Zagreb, 2025 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)
3. **Mitrović, Saša**; Vrček, Neven Automation of classification tasks with imbalanced datasets using an adapted CRISP-DM methodology, Hinweis Third International Conference on Computer Science, Cyber Security and Information Technology (CCIT), June 28-29, 2025 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)
4. **Mitrović, Saša**; Vrček, Neven Automated machine learning methods for efficient prediction of carbon dioxide emissions in building sector, SINERGY International Conference on Smart Energy Management technologies, September 26-29, 2023 (predavanje, međunarodna recenzija, znanstveni)
5. **Mitrović, Saša** E-LEARNING DURING THE COVID-19 PANDEMIC // Proceedings of the 90th International Scientific Conference on Economic and Social Development – "Building Resilient Society: National and Corporate Security" / Kopal, Robert ; Samodol, Ante ; Buccella, Domenico (ur.). Zagreb, 2022. str. 157-167 (predavanje, recenziran, cjeloviti rad (in extenso), znanstveni)
6. Zekić-Sušac, Marijana; **Mitrović, Saša**; Has, Adela Machine learning based system for managing energy efficiency of public sector as an approach towards smart cities. // International journal of information management, In Press (2020), 102074, 12 doi:10.1016/j.ijinfomgt.2020.102074 (međunarodna recenzija, članak, znanstveni)

7. Dukić, Branimir; Dukić, Stojanka; **Mitrović, Saša** MODEL OF ADAPTATION OF THE OPERATION OF THE FACULTY OF ECONOMICS IN OSIJEK TO THE DIGITAL AGE. // 8th INTERNATIONAL SCIENTIFIC SYMPOSIUM ECONOMY OF EASTERN CROATIA – VISION AND GROWTH / Leko Šimić, Mirna ; Crnković, Boris (ur.). Osijek: Josip Juraj Strossmayer University of Osijek, Faculty of Economics in Osijek, Croatia, 2019. str. 908-924 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)
8. **Mitrović, Saša**; Dukić, Stojanka; Dukić, Branimir WEBSITE USABILITY EVALUATION MODEL: ECONOMICS FACULTIES IN THE REPUBLIC OF CROATIA. // INTERDISCIPLINARY MANAGEMENT RESEARCH XV/INTERDISZIPLINÄRE MANAGEMENTFORSCHUNG XV / Barković, D. ; Crnković, B. ; Zekić Sušac, M. ; Dernoscheg, K. H. ; Pap, N. ; Runzheimer, B. ; Wentzel, D. (ur.). Opatija: Josip Juraj Strossmayer University of Osijek, Faculty of Economics in Osijek, croatiaPostgraduate doctoral Study Program in ManagementHochschule Pforzheim University, germanycroatian Academy of Sciences and Arts, 2019. str. 1110-1125 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)
9. Tonković, Zlatko; **Mitrović, Saša**; Zekić-Sušac, Marijana Business intelligence system for managing natural gas consumption of public buildings. // International Scientific Conference on Economic and Social Development – "Building Resilient Society" / Veselica, Rozana ; Dukic, Gordana ; Hammes, Khalid (ur.). Zagreb: Varazdin Development and Entrepreneurship Agency, Varazdin, Croatia, 2018. str. 769-778 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)
10. Zekić-Sušac, Marijana; Has, Adela; **Mitrović, Saša** Recursive Partitioning in Predicting Energy Consumption of Public Buildings. // Proceedings of the 29th Central European Conference on Information and Intelligent Systems / Strahonja, Vjeran ; Kirinić, Valentina (ur.). Varaždin: Faculty of Organization and Informatics, University of Zagreb, 2018. str. 179-186 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)
11. **Mitrović, Saša**; Zekić Sušac, Marijana A systematic literature review of machine learning algorithms in modeling buildings energy efficiency. // WEENTECH

Proceedings in Energy / Agarwal, Avlokita ; Singh, Renu (ur.). Edinburgh, United Kingdom: World Energy and Environment Technology Ltd., 2017. str. 191-196 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)

12. Požega, Željko; Crnković, Boris; **Mitrović, Saša** Analysis of Rates of Return on Investment in Education // 2nd International Conference "Vallis Aurea" - focus on: Regional Development / Katalinić, Branko (ur.). Požega, 2010. str. 1191-1195 (predavanje, međunarodna recenzija, cjeloviti rad (in extenso), znanstveni)