UPORABA METOD STROJNEGA UČENJA ZA KLASIFIKACIJO NALOG PO PRIORITETAH V IT PROJEKTIH

Tatyana Unuchak, Mirjana Kljajić Borštnar, Yauhen Unuchak
Fakulteta za organizacijske vede, Univerza v Mariboru, Kidričeva cesta 55a,4000 Kranj
tatyana.unuchak@student.um.si, mirjana.kljajic@um.si, yauhen.unuchak@student.um.si

Izvleček

Določanje prioritet in razvrščanje nalog še vedno predstavlja izziv pri učinkovitem vodenju projektov. Obstaja veliko klasičnih pristopov za določanje prioritet. Vendar so te tehnike delovno intenzivne, subjektivne in neprilagodljive. V prispevku obravnavamo pristope za samodejno določanje prioritet nalog v IT projektih, ki temeljijo na strojnem učenju. Raziskujemo, kako lahko z uporabo metod strojnega učenja pomagamo projektnim vodjem pri učinkovitejšem razvrščanju nalog v IT projektih. V ta namen smo na množici več kot 1000000 zapisov projektnih nalog razvili klasifikacijski model za samodejno določanje prioritet. Problem, ki smo ga obravnavali, je večrazredni, pri tem je večina primerov, označenih z najvišjo prioriteto, kar predstavlja izziv pri modeliranju kot tudi pri učinkovitosti upravljanja IT projektov. Preskusili smo različne algoritme ter različne pristope, s ciljem izboljšanja rezultatov klasifikacije. Pokazali smo, da je naloge smiselno razvrstiti v manjše skupine prioritet, kar prispeva k večji natančnosti klasifikacijskega modela in preglednosti prioritet nalog, slednje pa lahko olajša upravljanje IT projektov.

Ključne besede: vodenje IT projektov, strojno učenje, določanje prioritet nalog, večrazredna klasifikacija, neuravnoteženost podatkov.

Using machine learning methods to classify tasks by priority in IT projects

Abstract

Prioritizing and classifying tasks remain a challenge in effective project management. There are many classic approaches to prioritization. However, all these techniques are labour intensive, subjective and lack flexibility. In this paper we investigate machine learning based approaches for automatic prioritization of tasks in IT projects. We explore how machine learning methods can be used to help project managers prioritize tasks in IT projects more efficiently. We developed a classification model for automatically determining priorities based on a dataset of over 1,000,000 project task records. The problem we addressed is multi-class, with the majority of cases labelled with the highest priority, which presents a challenge both in modelling and in the efficiency of IT project management. To address these challenges, we explored strategies to enhance classification performance, including reducing the number of priority classes. Our findings demonstrate that simplifying the priority structure improves the model's accuracy and contributes to more efficient task management in IT projects.

Keywords: *IT project management, machine learning, task prioritization, multiclass classification, data imbalance.*

1 UVOD

Napredki na področju umetne inteligence in strojnega učenja hitro spreminjajo številne panoge po vsem svetu [1]. Njihov največji vpliv na procese upravljanja je opazen v IT sektorju [2]. V zadnjih letih se je močno povečalo zanimanje za uporabo umetne inteligence pri optimizaciji različnih vidikov vodenja projektov. Nedavna raziskava [2] je pokazala, da se različne metode umetne inteligence, kot sta razvrščanje in napovedovanje, dosledno vključujejo v splošne procese vodenja projektov. Algoritmi strojnega učenja lahko samodejno kategorizirajo elektronska sporočila, posodabljajo statuse projektov in ustvarjajo poročila – naloge, ki bi sicer zahtevale dragoceni čas in vire [3], [4]. Ta avtomatizacija omogoča vodjem projektov, da se osredotočijo na strateške vidike svojega dela.

Posebej zanimiv vidik projektnega vodenja je povezan z določanjem prioritet in razvrščanjem nalog. Sodobni projekti namreč ustvarjajo ogromno količino podatkov, vključno z besedilnimi opisi nalog, časovnimi oznakami, metrikami uspešnosti itd. Ročna obdelava in analiza teh podatkov postaja vse bolj zamudna in neučinkovita. Z uporabo strojnega učenja pri razvrščanju nalog lahko bistveno izboljšamo vodenje projektov, saj to pomaga prepoznati kritične naloge, napovedati tveganja in optimizirati dodeljevanje virov. To je še posebej pomembno pri velikih in zapletenih projektih, kjer lahko napaka pri določanju prioritet nalog povzroči velike zamude in prekoračitve virov (denarnih, časovnih, človeških, tehničnih). Človeške napake in subjektivnost v procesu odločanja lahko negativno vplivajo na vodenje projektov. Uporaba strojnega učenja pomaga zmanjšati vpliv človeških napak ter zagotavlja bolj objektiven in temelječ na podatkih pristop k določanju prioritet nalog. Algoritmi strojnega učenja lahko samodejno obdelajo in analizirajo velike količine podatkov, kar znatno pospeši postopek odločanja. Algoritmi lahko upoštevajo različne dejavnike, kot so besedilni opis naloge v IT projektu, njena kompleksnost, časovni okvir, poslovna pomembnost itd. Avtomatizacija tega procesa omogoča tudi hitro prilagajanje spremembam, saj je mogoče modele sproti prilagajati posodobljenim podatkom.

Določanje prioritet nalog vključuje ocenjevanje in razvrščanje nalog glede na njihovo pomembnost, nujnost in vpliv. Pri vodenju projektov določanje prioritet nalog zagotavlja, da so prizadevanja usmerjena v naloge, ki so najpomembnejše za doseganje ciljev projekta, kar preprečuje, da bi se čas in viri zapravljali za manj pomembne naloge. Učinkovit organizator dela se zaveda, kako pomembno je določanje prioritet nalog za uspešno upravljanje časa.

Številni dokumenti, ki urejajo projektne dejavnosti (npr. PMBOK, BABOK, PRINCE2), vsebujejo opise nekaterih tehnik določanja prioritet [5], [6], [7]. Vendar so vse te tehnike pri izvajanju delovno intenzivne (zahtevajo posebne napore razvojne ekipe), subjektivne (ker prioritete določajo ljudje) in same po sebi niso prilagodljive (v primeru sprememb zahtev naročnika, pojave novih tehnologij itd.).

Cilj raziskave je razviti modele strojnega učenja za klasifikacijo nalog v IT projektu po prioritetah, kar bo omogočilo boljšo organizacijo delovnih nalog v procesu razvoja programske opreme. Klasifikacijski model bo pomagal samodejno določiti prioritete nalog, natančneje opredeliti najpomembnejše naloge in izboljšati dodeljevanje virov v IT projektih. Osnovno raziskovalno vprašanje je: »Ali lahko z uporabo metod strojnega učenja pomagamo projektnim vodjem pri bolj učinkovitem razvrščanju prioritet nalog v IT projektih?«.

V nadaljevanju članka najprej predstavimo pregled sorodnih del, povezanih z razvrščanjem prioritet, opišemo metodologijo raziskave, faze predhodne obdelave podatkov, uporabljene algoritme razvrščanja, metrike vrednotenja modelov razvrščanja in opisujemo postopek modeliranja. Nadaljujemo z rezultati modeliranja, vključno z analizo kakovosti dobljene razvrstitve. Na koncu zaključimo z oceno pridobljenih rezultatov, omejitvami in implikacijami za prakso.

2 SORODNA DELA

Mednarodni inštitut za poslovno analizo (angl. International Institute of Business Analysis) je koncept določanja prioritet oblikoval kot »določitev relativne pomembnosti niza elementov za določitev vrstnega reda, v katerem jih bomo obravnavali«[6]. Posebno pozornost je treba nameniti temu, kaj je predmet prioritizacije (ali čemu dajemo prioriteto). Številni avtorji obravnavajo problem »določanja prioritet zahtev« (angl. »requirements prioritization«) [8], [9], [10]. Vendar se današnji razvijalci ne soočajo le s prioritetnim razvrščanjem zahtev, temveč tudi s prioritetnim razvrščanjem epov (angl. Epic), uporabniških zgodb (angl. User Story), nalog (angl. Task), hroščev (angl. Bug) [11]. Raznolikost vrst projektnih nalog omogoča prilagodljivo vodenje razvojnega procesa, vendar hkrati prinaša izzive pri določanju prioritet in razvrščanju nalog. Ob upoštevanju teh vidikov lahko rečemo, da splošne pristope za določanje prioritet nalog lahko uporabimo tudi v IT projektih. Obstoječe pristope za določanje prioritet razvrstimo na dve skupini: klasični pristopi in pristopi, ki temeljijo na strojnem učenju.

2.1. Klasični pristopi

Obstaja več pristopov k določanju prioritet nalog, ki vključujejo naslednje metode: MoScoW (angl. Must have, Should have, Could have, Won't have this time), preprosto rangiranje, razvrščanje z mehurčki, drevo binarnega iskanja (angl. Binary Search Tree), analitični hierarhični proces (angl. Analytic Hierarchy Process, AHP), pristop stroškov in vrednosti (angl. Cost-Value Approach, CVA), metoda $100 (angl. Hundred Dollar Method), kumulativno glasovanje, igre načrtovanja (angl. Planning Game, PG), numerično razvrščanje (angl. Numerical Assignment, NA), teorija-W (Win-Win Theory), Wiegerjev pristop [8], [9], [10], [12]. Glede na izbrano metodo je treba uporabiti ordinalno, nominalno ali intervalno lestvico [12].

Med njimi najpogosteje uporabljene metode so AHP, igre načrtovanja, CVA, razvrščanje z mehurčki, MoSCoW [8]. Te metode omogočajo vodjem projektov in razvijalcem sprejemanje utemeljenih odločitev o tem, katere naloge ali zahteve imajo prednost pri izvedbi. Vendar pa imajo te metode resne omejitve, zlasti ko gre za stopnjevanost pri velikem številu zahtev. V pogojih sodobnih projektov programske opreme, kjer se število nalog in sprememb lahko meri v stotinah in celo tisočih, postanejo ročne metode določanja prioritet izjemno zamudne in neučinkovite [8], [12]. Postopek določanja prioritet mora biti enostaven za osvojitev, preprost za uporabo, neviden za zainteresirane strani, da si pridobi njihovo zanimanje in zaupanje, razumljiv za nestrokovnjake, zanesljiv in učinkovit [12].

Metode AHP, BST, numerično razvrščanje in MoSCoW, še vedno priljubljene za določanje prioritet nalog, vendar jih je treba prilagoditi bolj zapletenim in dinamičnim projektnim okoljem v realnem svetu [10]. Raziskovalci se zato obračajo k bolj prilagodljivim in razširljivim metodam, ki upoštevajo parametre, kot so odvisnost nalog, kritičnost in časovni razpored. Pri tem AHP je ena od najbolj natančnih metod, ki se uporabljajo v IT industriji, vendar obstajajo tudi njene omejitve pri razširljivosti in potrebe po iskanju novih pristopov [9].

Sodobni trendi določanja prioritet nalog se vse bolj obračajo k metodam strojnega učenja, ki lahko avtomatizirajo postopek analize nalog, ugotavljanja povezav in optimizacije vrstnega reda njihovega izvajanja [9].

2.2. Pristopi strojnega učenja

Sodobni pristopi k določanju prioritet nalog uporabljajo možnosti strojnega učenja za natančnejše in učinkovitejše upravljanje nalog, zlasti pri kompleksnih projektih, ki zahtevajo veliko podatkov. Metode strojnega učenja avtomatizirajo postopek določanja prioritet na podlagi podatkov o nalogi, kot so opis naloge, trajanje izvedbe, kritičnost in drugi dejavniki. Algoritmi, kot so odločitvena drevesa, naključni gozdovi, gradientno povečanje z drevesi odločanja in nevronske mreže, se lahko učijo iz preteklih podatkov in to znanje uporabijo za

določanje prioritet nalog v prihodnosti. Leta 2004 sta Cubranic in Murphy [13] izvedla eno od prvih raziskav, katerih cilj je bil določiti prioritete napak z uporabo metod strojnega učenja. Pokazali so, da lahko uporaba kategorizacije besedila z nadzorovanim Bayesovim učenjem izboljša natančnost dodeljevanja nalog razvijalcem. V študiji so uporabili velike nabore podatkov iz odprtokodnih projektov, ki so postali dejanski standardi za testiranje modelov strojnega učenja.

Možnosti uporabe strojnega učenja za izboljšanje natančnosti in hitrosti določanja prednostnih nalog v IT projektih so bile potrjene v številnih študijah. Kombinacija naivnega Bayesovega klasifikatorja z optimizacijo pričakovanj izboljša natančnost klasifikacije na 48 % [14].

Klasifikacija kot raziskovalno področje strojnega učenja danes zajema različne vidike in vrste nalog. Opredeljene so štiri glavne kategorije klasifikacijskih nalog v strojnem učenju: binarna, večrazredna, klasifikacija z več oznakami in neuravnotežena klasifikacija [15]. Najbolj priljubljeni algoritmi, ki se uporabljajo za reševanje takih nalog, so logistična regresija, k-najbližjih sosedov, odločitvena drevesa, metoda podpornih vektorjev, naivni Bayesov klasifikator, naključni gozdovi in gradientno povečanje z drevesi odločanja. Primeri različnih klasifikacij vključujejo odkrivanje neželene elektronske pošte, odkrivanje napak v proizvodnih procesih in napovedovanje pretvorbe, prepoznavanje obrazov, razvrščanje rastlinskih vrst in kategoriziranje novic itd. [15].

Pri delu z velikimi količinami podatkov pri razvoju programske opreme se pogosto pojavi problem neuravnoteženosti podatkov, ki otežuje postopek razvrščanja [15], [16], [17]. Uspešnost klasifikacije je zelo odvisna od stopnje neuravnoteženosti razredov [17]. Obstoječe rešitve za odpravo neuravnoteženosti razvite tako na ravni podatkov kot na ravni algoritmov. Na ravni podatkov je rešitev v uravnoteženju porazdelitve razredov s ponovnim vzorčenjem podatkovnega prostora, medtem ko se na ravni algoritmov rešitev osredotoča na prilagoditev obstoječih algoritmov za usposabljanje klasifikatorjev, s čimer se okrepi učenje manjšinskih razredov [16].

Tako pregled literature kaže, da se poleg tradicionalnih metod določanja prednosti nalog pogosto uporabljajo tudi sodobnejši pristopi, ki temeljijo na strojnem učenju in analizi podatkov. To je utemeljeno s potrebo po izboljšanju učinkovitosti in natančnosti projektnega vodenja v kontekstu nenehno naraščajoče količine podatkov in spremenljivosti zahtev.

3 METODOLOGIJA

V raziskavi smo sledili procesu CRISP-DM (angl. Cross-Industry Standard Process for Data Mining) [18]. Metodologija zagotavlja strukturiran pristop k izvajanju projektov podatkovne analize in strojnega učenja ter je sestavljena iz naslednjih šestih korakov: razumevanje (poslovnega) problema (angl. Business Understanding), razumevanje podatkov (angl. Data Understanding), priprava podatkov (angl. Data Preparation), modeliranje (angl. Modeling), vrednotenje modela (angl. Evaluation), izvajanje in spremljanje (angl. Deployment and Monitoring). Za podatkovno rudarjenje in obdelavo podatkov smo uporabili programski jezik Python in knjižnice ljson, pandas, seaborn, imblearn, matplotlib, sklearn.

3.1. Razumevanje (poslovnega) problema

V tej raziskavi je glavni poslovni problem potreba po avtomatizaciji postopka določanja prioritet nalog v IT projektih na podlagi njihovega besedilnega opisa. V sodobnih sistemih za upravljanje vodenje projektov, kot je Jira, je veliko nalog z različnimi prednostnimi nalogami. Vendar pa zaradi velike količine podatkov in zapletenosti ročnega razvrščanja obstaja tveganje, da naloge z visoko prioriteto ne bodo pravočasno obdelane, kar bo negativno vplivalo na učinkovitost ekipe. Ker je ocena prioritete naloge subjektivna, se pogosto zgodi, da je večina

nalog razvrščena v najvišjo prioriteto, kar ne prispeva k izboljšanju učinkovitosti izvajanja projektov.

Glavna ideja je raziskati smeri za izboljšanje rezultatov razvrščanja nalog v IT projektih na podlagi algoritmov strojnega učenja. Dobljeni rezultati naj bi postali osnova za izboljšanje procesa upravljanja nalog.

3.2. Razumevanje podatkov

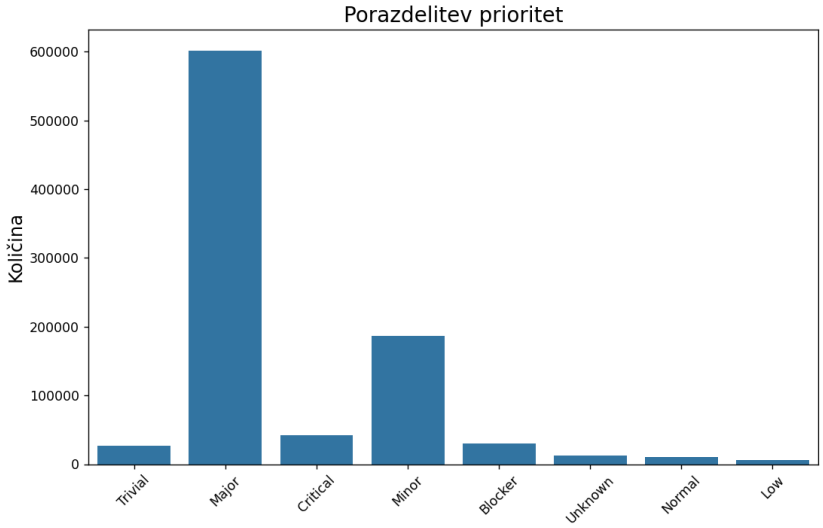
Podatke smo pridobili iz javno dostopnega vira Zenodo https://zenodo.org/records/5901804) [19]. Iz arhiva mongodump-JiraRepos.archive smo povzeli podatke in jih pretvorili v obliko JSON-datoteke, ki je vsebovala naslednje atribute (značilke) nalog v angleščini: prioriteta (angl. Priority), opis stanja (angl. Status Description), naziv stanja (angl. Status Name), kategorija stanja (angl. Status Category), ime ustvarjalca (angl. Creator Name), časovni pas ustvarjalca (angl. Creator Time Zone), ime poročevalca (angl. Reporter Name), skupni napredek (angl. Aggregate Progress), skupni seštevek (Aggregate Total), skupni odstotek (angl. Aggregate Percent), opis vrste zadeve (angl. Issue Type Description), ime vrste zadeve (angl. Issue Type Name), porabljeni čas (angl. Time Spent), ključ projekta (angl. Project Key), vrsta projekta (angl. Project Type), datum ustvarjanja (angl. Created Date), datum posodobitve (angl. Updated Date), opis naloge (angl. Task Description), povzetek (angl. Summary). Iz datoteke JSON smo izbrali polja "Priority", "Status Name", "Creator Name", "Time Spent", "Task Description" in pretvorili v obliko CSV. Izbrana polja omogočajo ključne informacije za razumevanje prioritet nalog, stanja njihove izvedbe, vpletenih oseb, porabljenega časa ter vsebine nalog. Za modeliranje smo izbrali značilke: prioriteta in opis naloge. Omejitev na dve ključni značilke zmanjša kompleksnost modela, kar je koristno pri začetnih analizah in primerno za končne uporabnike, saj omogoča jasne vpoglede v nujnost nalog in njihov kontekst. Obe značilki predstavljata besedilni podatkovni tip. Prioriteta je imela naslednje vrednosti: »Major«, »Minor«, »Critical«, »Blocker«, »Trivial«, »Unknown«, »Normal«, »Low«. Izvirna podatkovna datoteka je vsebovala 1014926 zapisov (primerov) o nalogah iz 645. IT projektov. Uporabili smo različne IT projekte, da bi lahko klasifikator učili na različnih naborih podatkov. Po odstranitvi zapisov z manjkajočimi vrednostmi nam je ostalo 916030 zapisov, enega od katerih prikazuje slika 1.

```
priority,status,creator_name,timespent,description
Major,Open,Ryan0751,,"In a production environment with some connectivity problems it was found the ZooKeeper
server was using over 1000 threads with name ""SyncThread"" (that were never being freed).
Looking through the server logs indicates that these nodes were experiencing connection timeouts to the leader.
A test environment (described below in the ""environment"" field of this ticket) showed that these connection
timeouts are what seem to be leaking these threads."
```

Slika 1: **Primer opisa naloge iz IT projekta s prioriteto »Major«**

Prioritete za vsako nalogo so določili avtorji, ki so te naloge ustvarili v Jiri. Za nadaljnjo obdelavo besedila je bila potrebna predhodna priprava podatkov, vključno s tokenizacijo, odstranjevanjem stop besed in lematizacijo.

Klasifikacijo smo izvedli na projektnih nalogah, ki so bile razvrščene v osmih razredih prioritet: »Trivial«, »Major«, »Critical«, »Minor«, »Blocker«, »Unknown«, »Normal«, »Low« (slika 2).



Slika 2: **Porazdelitev podatkov po prioritetah.**

Prioriteta »Major« je večinska (približno 80% vseh primerov sodi v ta razred), kar kaže na neuravnoteženost podatkov in jo je potrebno upoštevati pri nadaljnjem modeliranju.

3.3. Priprava podatkov

Priprava podatkov je pomemben korak v procesu razvoja modela strojnega učenja. Da bi se model učinkovito naučil in podal pravilne rezultate, je treba podatke očistiti in pretvoriti v ustrezno obliko. Sledili smo naslednjim korakom:

1. Nalaganje in začetna obdelava podatkov.
   Glavna naloga v tem koraku je bila naložiti podatke in se prepričati o njihovi pravilnosti.
2. Odstranitev zapisov z manjkajočimi podatki.
3. Odstranitev zapisov, ki vsebujejo dele programske kode, opredeljene s kombinacijami ključnih besed »if«, »for«, »class« ali dele SQL poizvedb, opredeljenih s kombinacijami ključnih besed »select«, »from«, »where« ali kombinacijami simbolov »:«, » : «, značilnih za podatke v obliki JSON.
4. Odstranitev kratkih zapisov, ki ne vsebujejo več kot ene besede, ob predpostavki, da kratki opisi nalog ne vsebujejo dovolj informacij za analizo.
5. Oblikovanje podatkov.
   Zaradi lažjega nadaljnjega dela smo podatke uredili v enotno obliko. Besedilo smo pretvorili v male črke, kar je omogočilo odpravo razlik med besedami, zapisanimi z različnimi črkami.
6. Tokenizacija in odstranitev stop besed.
   Tokenizacija je postopek razdelitve besedila na posamezne besede (»žetone«) [20]. Po tokenizaciji smo iz besedila odstranili stop besede (npr. prislovi in vezniki), ki za model niso nosile pomembnih informacij. V ta namen smo uporabili Pythonovo knjižnico »scikit-learn«. Vgrajeni seznam stop besed te knjižnice vključuje 318 besed, kot so »the«, »and«, »is«, »in«, »on«, »next«, »go«, »where«, »being« in druge.
7. Lematizacija.
   Lematizacija je postopek vračanja besed v njihovo osnovno obliko. Pri tem se odstranijo le pregibne končnice in vrne slovarsko obliko besede, ki je znana kot lema [20]. Za predobdelavo besedila smo uporabili lematizator WordNetLemmatizer iz knjižnice nltk (Pythonova knjižnica), ki je bese pretvoril v njihovo osnovno obliko. Na primer, besedi »following« in »thrown« sta bili pretvorjeni v obliko »follow« in »throw«. S tem se zmanjša število edinstvenih besed v besedilu in izboljša kakovost modela.
8. Vektorizacija besedila.

Vektorizacija besedila je postopek pretvorbe besedila v številčno obliko. V našem primeru smo uporabili tehniko TF-IDF (angl. Term Frequency-Inverse Document Frequency), ki nam omogoča, da besede predstavimo kot številčne vektorje glede na njihovo pomembnost v dokumentu [21]. V raziskavi smo za vektorizacijo besedila uporabili razred TfidfVectorizer iz scikit-learn (knjižnica Python).

9. Uravnoteženje podatkov.
   V prvotnih podatkih smo opazili precejšnje neravnovesje razredov – naloge s prioriteto »Major« so predstavljale približno 80% vseh nalog. Neuravnoteženi podatki predstavljajo izziv pri natančnosti razločevanja med posameznimi razredi, saj se model nauči prepoznati večinski razred, ne pa tudi manjšinskih razredov. Za rešitev te težave smo uporabili postopek uravnoteženja razredov z enakomernim vzorčenjem (angl. undersampling). Pod uravnoteženjem razumemo spreminjanje sestave vzorcev v učnem naboru podatkov z zmanjšanjem števila primerov v večinski skupini, tako da ustrezajo velikosti manjšinske skupine. Pri tem smo iz večinske skupine naključno izbrali primere, dokler nismo dosegli enake velikosti kot manjšinska skupina, in nato združili podatke v enoten učni nabor.

Po čiščenju in pripravi podatkov nam je ostalo 51536 zapisov o nalogah v IT projektih. Po odstranitvi stop besed in lematizaciji je bila velikost slovarja 84328 unikatnih besed. Med najpogostejšimi besedami so bile: »I« (frekvenca 40711-krat), »code« (pojavnost 28132-krat), »use« (pojavnosti 27479-krat), »error« (pojavnost 24775-krat), »info« (pojavnost 23067-krat).

3.4. Modeliranje

Modeliranje je faza procesa CRISP-DM, v kateri z izbranimi metodami strojnega učenja razvijamo, ocenjujemo in optimiziramo klasifikacijski model. V naši raziskavi smo uporabili različne algoritme strojnega učenja za razvoj klasifikacijskih modelov za določanje prioritet nalog v IT projektih. Model strojnega učenja je funkcija, ki preslika nize vhodnih podatkov (neodvisne spremenljivke, ki jih predstavljajo opisi nalog) v izhodne podatke (odvisna spremenljivka Y oziroma klasifikacijski razred). Cilj modela je pravilno uvrstiti vrednost odvisne spremenljivke na podlagi vrednosti neodvisnih spremenljivk.

V našem modelu je spremenljivka x_i opis naloge v IT projektu, potem je X množica, ki vsebuje opise vseh nalog v IT projektih. Spremenljivka y_k je prioriteta naloge, ki lahko pripada enemu od razredov množice $Y = \{\text{»Trivial«}, \text{»Major«}, \text{»Critical«}, \text{»Minor«}, \text{»Blocker«}, \text{»Unknown«}, \text{»Normal«}, \text{»Low«}\}$.

Podatke smo razdelili na učno množico U, ki je definirana kot množica, na kateri se algoritem uči in je definirana, kot prikazuje enačba (1), in testno množico T, ki je definirana kot množica, na kateri se algoritem preizkusi. Učna množica je definirana z uporabo desetkratnega prečnega preverjanja (angl. cross-validation), kjer model ob vsakem koraku uporablja 9/10 podatkov za učenje, preostalo 1/10 pa za preizkušanje [22].

$$U = \{(x_1, y_k), (x_2, y_k), \dots, (x_l, y_k)\}\,, \qquad (1)$$

kjer je $y_k \in Y, k \in [1, m]$, m je število razredov prioritet, $x_i \in X$, $i \in [1, n]$, n je število vseh opisov nalog v IT projektih, $l < n$, l je velikost učne množice.

Cilj algoritmov klasifikacije je, da ob dani učni množici *U* najdejo naslednjo funkcijo (2):

$$h: x_i \rightarrow y_k. \qquad (2)$$

Zgradili smo več modelov, pri čemer smo spreminjali število razredov izhodne spremenljivke: osnovno z 8. razredi prioritet in različice, kjer smo obstoječih osem razredov združili v dva razreda ($Y = \{\text{»High Priority«}, \text{»Low Priority«}\}$.), tri razrede ($Y =$

{» High Priority«, »Medium Priority«, »Low Priority«}) in štiri razrede ($Y =$ {» High Priority«, »Critical Priority«, »Medium Priority«, »Low Priority«}).

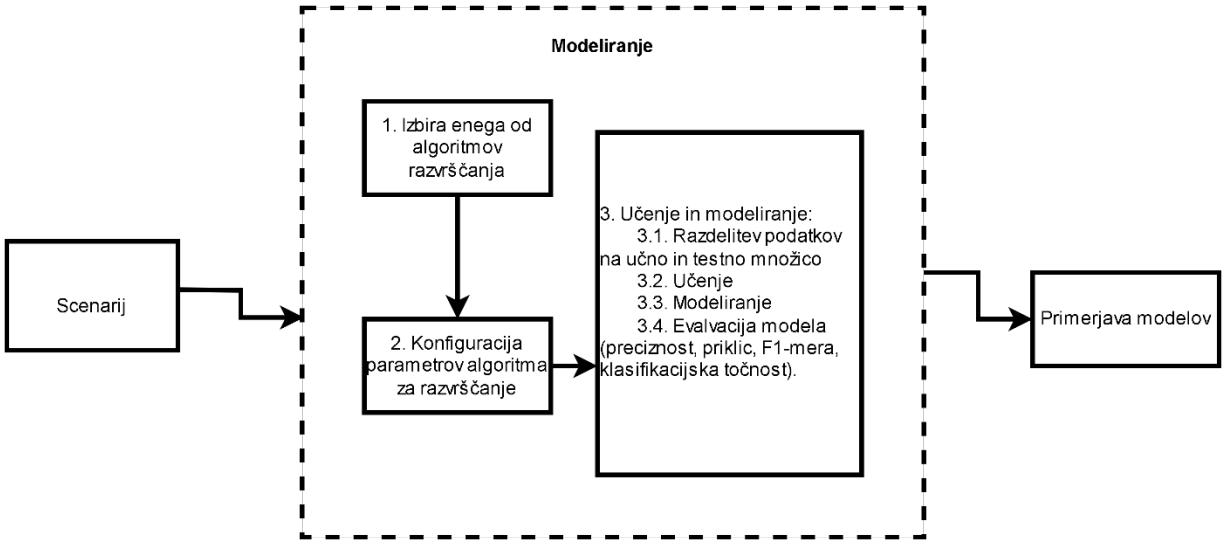
Vse postopke za modeliranje smo opisali z naslednjimi scenariji, ki so prikazani v tabeli 1.

Tabela 1: Scenariji modeliranja.

Naziv scenarija	Količina razredov	Opis	Število primerov
Scenarij 1	8	Vključuje prioritete »Trivial«, »Major«, »Critical«, »Minor«, »Blocker«, »Unknown«, »Normal«, »Low«. Podatki niso uravnoteženi	26508/601876 /42136/186604 /30258/12103 /10103/6442
Scenarij 2	8	Vključuje iste prioritete, kot pri scenariju 1, vendar podatki so uravnoteženi (»Trivial«, »Major«, »Critical«, »Minor«, »Blocker«, »Unknown«, »Normal«, »Low«)	6442 v vsakem razredu
Scenarij 3	8	Vključuje iste prioritete, kot pri scenariju 2, vendar podatki so uravnoteženi in programska koda je odstranjena(»Trivial«, »Major«, »Critical«, »Minor«, »Blocker«, »Unknown«, »Normal«, »Low«)	1430 v vsakem razredu
Scenarij 4	2	Dva razreda prioritet: 1) »High Priority«, v katerega smo vključili naloge s prioritetami »Blocker«, »Critical« in »Major«, 2) »Low Priority«, v katerega smo vključili naloge s prioritetami »Minor«, »Trivial«, »Normal«, »Unknown«, »Low«.	241760 v vsakem razredu
Scenarij 5	3	Tri razreda: naloge s prioriteto »Major«, »Blocker«, »Critical« smo uvrstili v razred 1)»High Priority«. V razred 2)»Low Priority« smo uvrstili naloge s prioriteto »Unknown«, »Low« in »Minor«. Naloge s »Trivial« in »Normal« prioritete smo uvrstili v razred 3)»Medium Priority«.	23771 v vsakem razredu
Scenarij 6	4	Štiri razreda. Skupina 1)»High Priority« vključuje naloge s prioriteto »Major«. Skupina 2)»Critical Priority« vključuje naloge s prioriteto »Blocker« in »Critical«. Skupina 3)»Medium Priority« vključuje naloge s prioriteto »Minor« in »Trivial«. Skupina 4)»Low Priority« združuje naloge označene z »Normal«, »Unknown« in »Low«.	28648 v vsakem razredu

Iz tabeli 1 je razvidno, da je bilo pri scenarijih od 1 do 3 število razredov izvirno (8 razredov). Pri scenariju 2 smo preverili, kako vpliva na rezultate klasifikacije uravnoteženost podatkov. Za uravnoteženost smo uporabili postopek uravnoteženja razredov z enakomernim vzorčenjem. Pri scenariju 3 smo preverili, kako vplivata na rezultate klasifikacije uravnoteženost podatkov in odstranitev programske kode. Pri scenarijih od 4 do 6 smo preverili, ali se rezultati klasifikacije izboljšajo, če podatke razvrstimo v manjše število razredov.

Shematsko je proces modeliranja prikazan na sliki 3.



Slika 3: **Postopek modeliranja**

Slika 3 prikazuje postopek modeliranja na podlagi različnih scenarijev. Najprej izberemo enega izmed izbranih algoritmov klasifikacije, nato prilagodimo njegove parametre. Podatke razdelimo na učne in testne množice, kar omogoča začetek postopka učenja modela. Sledi modeliranje, v katerem izračunamo metrike modela (preciznost, priklic in F1-mera). Postopek modeliranja izvedemo za vse scenarije z izbranimi algoritmi.

Pri modeliranju smo uporabili naslednje klasifikatorje: naključni gozd (angl. Random forest), metoda podpornih vektorjev (angl. support vector machine, SVM), logistična regresija (angl. Logistic regression), gradientno povečanje z drevesi odločanja (angl. Gradient Boosting with Decision Trees) in k-najbližjih sosedov (angl. k-Nearest Neighbors – kNN).

Razlogi za izbiro teh algoritmov izhajajo iz izkušenj preteklih raziskav. Algoritem naključni gozd dosledno zagotavlja visoko natančnost in robustnost, zlasti pri obravnavi velikih in zapletenih podatkovnih nizov [23]. Metoda podpornih vektorjev se je izkazala za učinkovito pri nalogah, ki zahtevajo visoko natančnost klasifikacije, zlasti, kadar je podatke težko ločiti [24]. Gradientno povečanje z drevesi odločanja sodi med ansambelske metode, ki s postopnim zmanjševanjem napake oziroma optimizacijo iz šibkih klasifikatorjev gradi močne klasifikatorje. V praksi kaže odlične rezultate pri nalogah, ki zahtevajo natančno klasifikacijo, in lahko znatno izboljša učinkovitost v primerjavi z modeli z eno samo optimizacijo, kot so odločitvena drevesa [25]. Logistična regresija se zaradi svoje preprostosti in učinkovitosti pogosto uporablja v različnih aplikacijah strojnega učenja, kjer je potrebno zanesljivo napovedovanje kategorij [26]. Algoritem k-najbližjih sosedov smo izbrali zaradi preprostosti izvajanja [27].

3.5. Vrednotenje modelov

Za vrednotenje pridobljenih rezultatov modeliranja smo uporabili naslednje metrike: klasifikacijska točnost (angl. Classification Accuracy), preciznost (angl. Precision), priklic (angl. Recall), F1-mera (angl. F1-score). Metrike, ki jih upoštevamo, temeljijo na naslednjih rezultatih: pravilno klasificirane kot pozitivne (angl. true positive – TP), pravilno klasificirane kot negativne (angl. true negative – TN), napačno klasificirane kot pozitivne (angl. false positive – FP) in napačno klasificirane kot negativne (angl. false negative – FN). V tabeli 2 prikazujemo osnovne enačbe za izračun metrik za vrednotenje modelov za binarno in večrazredno klasifikacijo. Uporabili smo enačbe za večrazredno klasifikacijo [16], [28], [29] (stolpec 3 tabele 2).

Tabela 2: Enačbe za izračun mer točnosti klasifikacije.

Metrika	**Izračun binarne klasifikacije**	**Izračun večrazredne klasifikacije, k-število razredov**
1	2	3
Klasifikacijska točnost (ACCUR)	$\text{ACCUR} = (TP + TN)/(TP + TN + FN + FP)$	$\text{Povprečno ACCUR} = \sum_{i=1}^{K} \text{ACCUR}_i/K$
Preciznost (PREC)	$\text{PREC} = TP/(TP + FP)$	$\text{Povprečno PREC} = \sum_{i=1}^{K} \text{PREC}_i/K$
Priklic (REC)	$\text{REC} = TP/(TP + FN)$	$\text{Povprečno REC} = \sum_{i=1}^{K} \text{REC}_i/K$
F1-mera (F1)	$\text{F1} = 2 * \text{PREC} * \text{REC}/(\text{PREC} + \text{REC})$	$\text{Povprečno F1} = \sum_{i=1}^{K} \text{F1}_i/K$

Za ocenjevanje kakovosti modelov smo uporabili tudi krivuljo delovne karakteristike sprejemnika (angl. Receiver Operating Characteristic –ROC krivulja), ploščino pod ROC krivuljo (angl. Area Under ROC Curve – AUC ROC), krivuljo preciznost-priklic (angl. Precision-Recall curve) in ploščino pod krivuljo preciznost-priklic (angl. Area Under the Precision-Recall Curve – AUC PR) [30], [31].

Krivulja ROC prikazuje razmerje med pogostostjo pravilno klasificiranih pozitivnih rezultatov (TP, na osi Y) in pogostostjo napačno klasificiranih pozitivnih rezultatov (FP, na osi X) ob spreminjanju praga razlikovanja [28], [32]. To nam pomaga razumeti, kako dobro naš model ločuje med pozitivnimi in negativnimi primeri, saj lahko spremljamo razmerje med pravilno prepoznanimi pozitivnimi primeri (TP) in napačno prepoznanimi negativnimi primeri (FP) pri različnih nastavitvah pragov. V osnovi je namenjena analizi binarne klasifikacije. Posebej je treba omeniti pristope za izračun ROC krivulj za večrazredno klasifikacijo in neuravnotežene podatke. Obstajata dva pristopa za izračun ROC krivulje za klasifikacijo več razredov: pristop »eden proti vsem« (angl. one versus all – OvA) in pristop »eden proti enemu« (angl. one versus one – OvO) [30], [33]. Pristop OvA temelji na učenju niza neodvisnih binarnih klasifikatorjev, v katerih je en razred pozitiven, vsi drugi pa negativni. Po izračunu ROC krivulje za vsak binarni klasifikator se krivulje povprečijo, da dobimo končno ROC krivuljo [30]. Pri pristopu OvO se za vsako možno kombinacijo kategorij nauči binarni klasifikator. Če je na primer treba razlikovati tri kategorije, se razvijejo trije ločeni binarni klasifikatorji (razred 1 proti razredu 2, razred 1 proti razredu 3 in razred 2 proti razredu 3). Ko je izračunana ROC krivulja za vsak binarni klasifikator, se krivulje združijo in dobimo končno krivuljo ROC [34]. V našem primeru so enačbe za izračun metrik binarnih klasifikatorjev in metrik več razredov podane v tabeli 2.

Na neuravnoteženih podatkih pa se še bolje, kot ROC krivulja izkaže krivulja preciznost-priklic, ki prikazuje razmerje med deležem pravilno klasificiranih pozitivnih primerov med vsemi pozitivnimi primeri, torej pravilno pozitivno klasificiranih in nepravilno negativno klasificiranih) [28] saj se stopnja lažno pozitivnih rezultatov pri zelo neuravnoteženih podatkih zmanjša zaradi velikega števila resničnih negativnih rezultatov [35], [36]. Za oceno kakovosti naučenih modelov smo uporabili tako krivuljo preciznost-priklic (za modele naučene na neuravnoteženih podatkih), medtem ko smo za modele, naučene na uravnoteženih podatkih uporabili ROC krivuljo.

4 REZULTATI

V tem poglavju predstavljamo rezultate modeliranja in oceno njihove uspešnosti z uporabo ustreznih metrik.

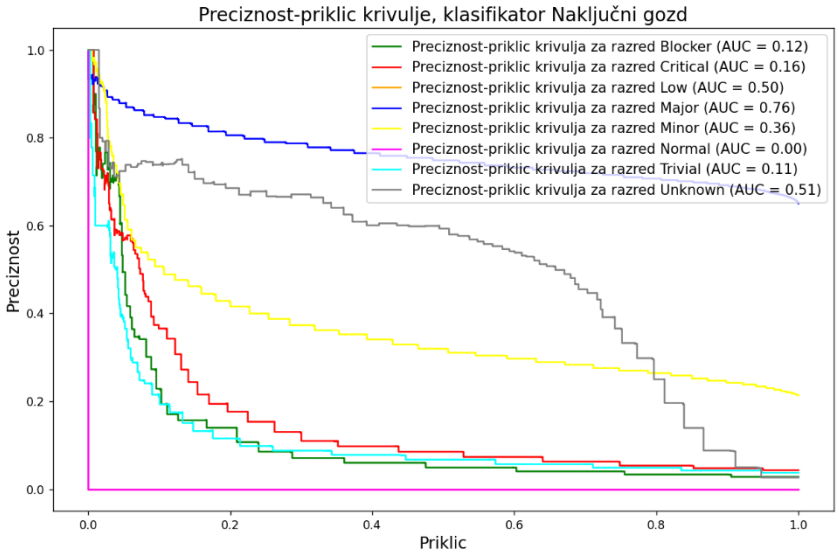
Modeliranje smo izvedli na predhodno pripravljenih podatkih o nalogah iz različnih IT projektov. Primerjali smo rezultate modeliranja za scenariji 1 – 6. Uporabili smo algoritme naključni gozd, metoda podpornih vektorjev, logistična regresija, gradientno povečanje z drevesi odločanja, k-najbližjih sosedov. Kriterije za preverjanje kakovosti naučenih modelov za različne scenarije prikazujemo v tabeli 3.

Tabela 3: Glavni dobljeni rezultati modeliranja (algoritem naključni gozd).

Metrike	**Scenarij 1**	**Scenarij 2**	**Scenarij 3**	**Scenarij 4**	**Scenarij 5**	**Scenarij 6**
1	**2**	**3**	**4**	**5**	**6**	**7**
Preciznost	0,20	0,41	0,36	0,74	0,57	0,53
Priklic	0,15	0,41	0,37	0,75	0,57	0,54
F1-mera	0,13	0,41	0,36	0,74	0,57	0,53
Klasifikacijska točnost	0,64	0,41	0,37	0,75	0,69	0,61
AUC ROC	-	0,81	0,75	0,82	0,77	0,80
AUC PR	0,32	0,43	0,37	0,80	0,64	0,56

V nadaljevanju rezultate za posamezne scenarije podrobneje opišemo.

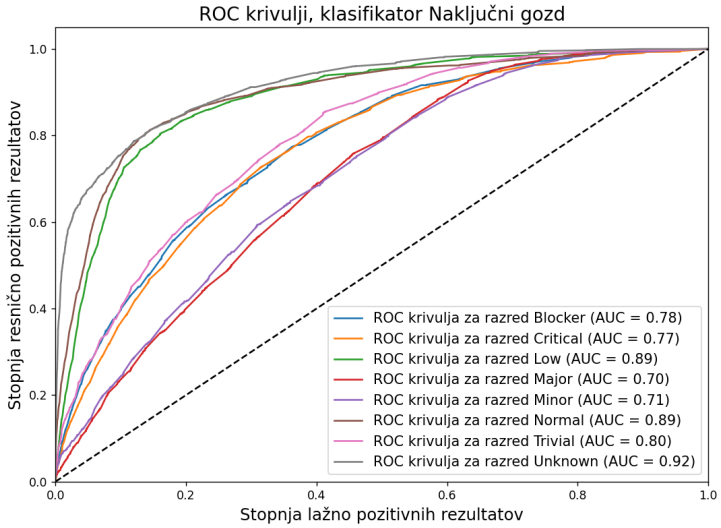
Prvi scenarij (scenarij 1) je predvideval modeliranje brez predhodnega postopka uravnoteženja podatkov. Rezultate testiranja modela prikazujemo s krivuljo preciznost-priklic (slika 4).



Slika 4: **Krivulje Preciznost-Priklic za scenarij 1 (algoritem naključni gozd).**

Iz slike 4 je razvidno, da ima razred »Major« (0,76) najvišjo vrednost AUC PR. Razreda »Unknown« (0,51) in »Low« (0,50) imata zmerni vrednosti AUC PR. Preostali razredi imajo nizke vrednosti AUC PR. Iz tabele 3 (stolpec 2) je razvidno, da so metrike priklic, preciznost in F1-mera zelo nizke.

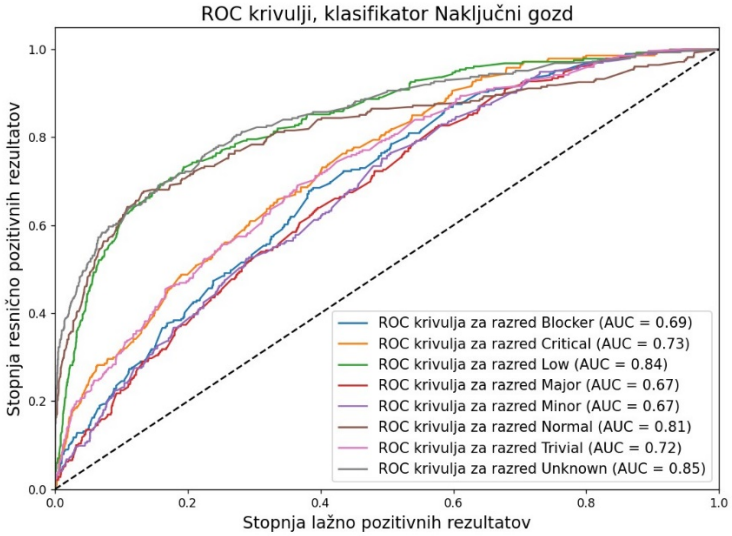
V scenariju 2 smo zaradi uravnoteženja podatkov za oceno rezultatov uporabili ROC-krivulje. Rezultati za scenarij 2 so prikazani na sliki 5.



Slika 5: **Krivulje ROC za scenarij 2 (algoritem naključni gozd).**

S slike 5 je razvidno, da model najbolje od ostalih razredov loči razrede »Unknown«, »Low« in »Normal« z vrednostmi AUC ROC 0,92 in 0,89. Za razred »Trivial« model kaže rezultate z AUC ROC okoli 0,80. Vrednosti AUC ROC za razrede »Blocker«, »Critical«, »Major« se gibljejo med 0,70 in 0,78.. Povprečne vrednosti metrik preciznosti, priklica in F1-mere za vse razrede so približno 0,41 (tabela 3, stolpec 3).

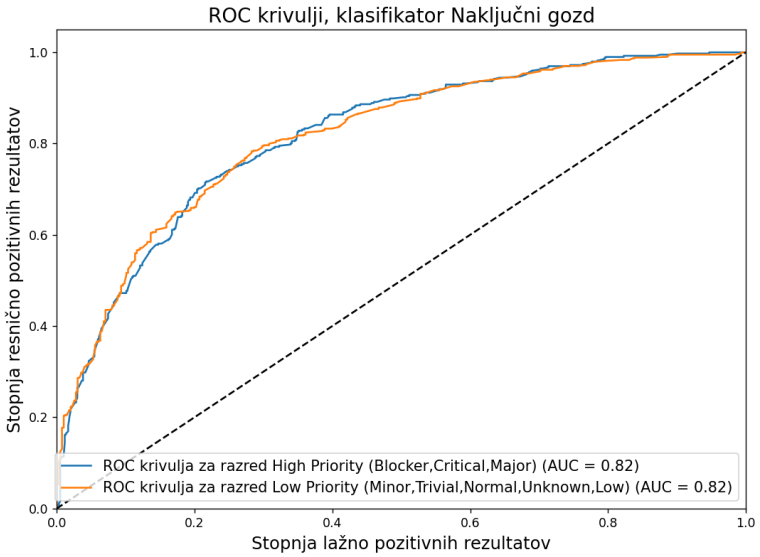
Pri scenariju 3 (odstranitev programske kode) so bili rezultati klasifikacije slabši (tabela 3 stolpec 4). Preciznost je bila 0,36, priklic 0,37, F1-mera 0,36. ROC krivulji za scenarij 3 so prikazani na sliki 6.



Slika 6: **Krivulje ROC za scenarij 3 (algoritem naključni gozd).**

S slike 6 je razvidno, da razred »Unknown« dosega najvišjo vrednost AUC ROC 0,85. Sledijo razredi »Critical« (AUC ROC= 0,73) in »Low« (AUC ROC = 0,84). Razredi »Blocker«, »Major«, in »Normal« imajo srednje vrednosti AUC ROC, ki se gibljejo med 0,67 in 0,81. Najslabše rezultate kaže razred »Trivial«, kjer vrednost AUC ROC znaša približno 0,75.

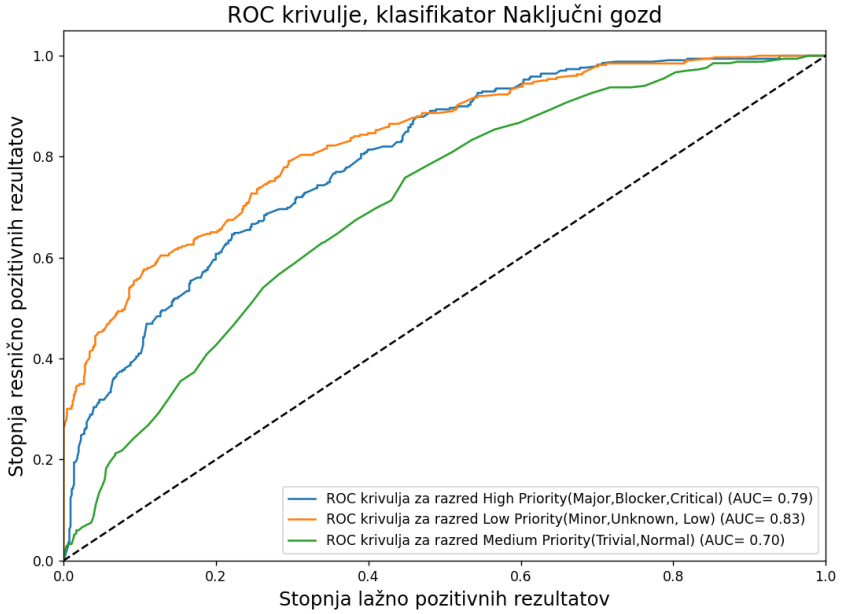
Dobljene vrednosti za scenarij 4 so prikazane na sliki 7 ter v tabeli 3 (stolpec 5).



Slika 7: Krivulji ROC za scenarij 4 (algoritem naključni gozd).

Na sliki 7 je za skupino razredov s prioriteto »Low Priority« in prioriteto »High Priority« AUC ROC 0,82. Tudi vrednosti metrik so precej visoke in znašajo 0,74 (0,75) (tabela 3 stolpec 5).

Na sliki 8 so prikazane vrednosti ploščine pod ROC krivuljo, pridobljene po razdelitvi podatkov v tri razrede (scenarij 5).

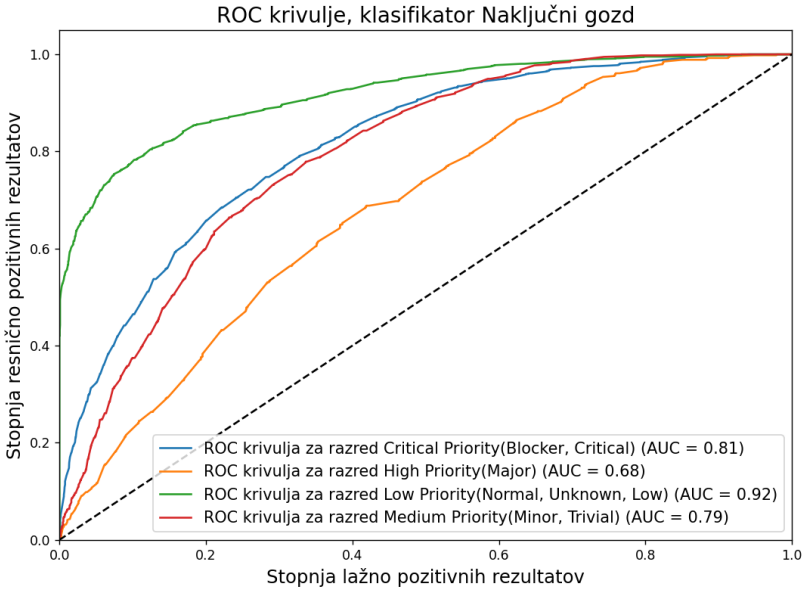


Slika 8: **Krivulje ROC za scenarij 5 (algoritem naključni gozd).**

Slika 8 prikazuje, da imata dve skupini (»High Priority« in »Low Priority«) vrednosti AUC ROC 0,79 in 0,83. Skupina razredov s prioriteto »Medium Priority« ima vrednost AUC ROC 0,70. Ta rezultat je v primerjavi z drugima dvema skupinama razredov izrazito nižji. Model je dosegel klasifikacijsko točnost 0,69 (tabela 3 stolpec 6).

Slika 9 prikazuje krivulje ROC za scenarij 6 (štiri skupine razredov). S slike 9 je razvidno, da model dosega najboljše rezultate za razrede s prioriteto »Low Priority«, kjer AUC ROC doseže

vrednost 0,92. Model prav tako uspešno klasificira razreda »Critical Priority« in »Medium Priority«, pri čemer vrednost AUC ROC znaša 0,81 oziroma 0,79. Vendar ima model pri razredu »High Priority« precejšnje težave, kar dokazuje nizka vrednost AUC ROC, ki znaša 0,68.



Slika 9: **Krivulje ROC za scenarij 6 (algoritem naključni gozd).**

Povprečne vrednosti metrik so približno 0,53 (tabela 3 stolpec 7), skupna točnost modela pa je 0,61.

5 DISKUSIJA

Glavni cilj naše naloge je bil razviti modele strojnega učenja za klasifikacijo nalog v IT projektih po prioritetah, kar bi olajšalo načrtovanje in izvedbo razvojnih procesov programske opreme.

V kontekstu naše raziskave in rezultatov analize podatkov lahko oblikujemo naslednje ugotovitve, ki jih povzemamo v tabeli 4.

Tabela 4: Pridobljeni rezultati.

Pristopi modeliranja	**Rezultat**
Uravnoteženje podatkov (scenarij 2)	izboljšalo
Odstranitev zapisov s programsko kodo (scenarij 3)	poslabšalo
Uporaba različnih algoritmov za razvrščanje (vsi scenariji)	brez vpliva, razen k-najbližjih sosedov (dobljeni rezultati za ta algoritem so bili vedno slabši)
Razdelitev v dva razreda (scenarij 4)	izboljšalo
Razdelitev v tri razrede (scenarij 5)	izboljšalo
Razdelitev v štiri razrede (scenarij 6)	izboljšalo

Rezultati modeliranja kažejo na pomembne ugotovitve glede uspešnosti modelov v različnih scenarijih. Pri scenariju 1 (neuravnoteženi podatki, osem razredov) smo pričakovano dosegli

najslabše rezultate kakovosti klasifikacijskega modela pri vseh uporabljenih algoritmih. Dobili smo nizke vrednosti metrik, kot so preciznost 0,20, priklic 0,15, F1-mera 0,13, pri tem je AUC PR 0,32. To poudarja potrebo po izboljšanju modela, zlasti z uporabo metod za uravnoteženje podatkov, da bi izboljšali uspešnost za razrede z nizko površino pod krivuljo preciznost-priklic.

Pri scenariju 2 (uravnoteženi podatki, osem razredov) smo z uporabo metode uravnoteženja podatkov (SMOTE) dosegli opazno izboljšanje kakovosti klasifikacijskega modela. Vrednosti metrik so se znatno povečale: preciznost se je zvišala z 0,20 na 0,41, priklic prav tako na 0,41, F1-mera na 0,41, medtem ko je AUC ROC dosegel 0,81. Povečanje teh metrik je rezultat boljše obravnave manj zastopanih razredov, kar omogoča bolj uravnoteženo delovanje modela. Ti rezultati potrjujejo, da ima uravnoteženje podatkov ključno vlogo pri izboljšanju učinkovitosti modela, še posebej pri klasifikaciji razredov z manjšo zastopanostjo.

Pri scenariju 3 (odstranitev zapisov s programsko kodo, osem razredov) smo opazili poslabšanje rezultatov modela. Preciznost se je znižala z 0,41 na 0,36, priklic je padel z 0,41 na 0,37, F1-mera se je znižala z 0,41 na 0,36, klasifikacijska točnost je zmanjšala z 0,41 na 0,37, je AUC ROC padla z 0,81 na 0,75. Odstranitev zapisov, ki so vsebovali programsko kodo, je negativno vplivala na zmogljivost modela, saj ti podatki pogosto vsebujejo ključne informacije, ki so pomembne za razvrščanje nalog. Ta scenarij poudarja potrebo po skrbni analizi podatkov pred njihovo odstranitvijo, da bi se izognili izgubi pomembnih informacij. Dobljeni rezultati so boljši v primerjavi z rezultati pri scenariju 1.

Pri scenariju 4 smo osem razredov preslikali v nova dva razreda (»High Priority« in »Low Priority«), kar je prineslo najstabilnejše in najnatančnejše rezultate, kar je tudi znano iz teorije in preteklih raziskav [37]. V primerjavi s scenarijem 3 se je preciznost povečala z 0,36 na 0,74, priklic je narasel z 0,37 na 0,75, F1-mera se je izboljšala z 0,36 na 0,74, klasifikacijska točnost pa se je prav tako dvignila z 0,37 na 0,75. AUC ROC je pokazal rahlo izboljšanje z 0,75 na 0,82. Vrednost AUC ROC je bila v tem scenariju najvišja, klasifikacija pa zanesljivejša. Poenostavljena klasifikacijska shema je omogočila bolj jasno razlikovanje med nalogami in izboljšala predvidljivost modela, kar je še posebej pomembno pri nalogah, kjer lahko klasifikacijske napake povzročijo resne posledice.

Pri scenariju 5 (oblikovanje treh razredov: »High Priority«, »Medium Priority« in »Low Priority«) smo opazili izboljšanje v primerjavi s scenariji 1–3. Ta razdelitev omogoča bolj natančno razlikovanje nalog glede na prioritete. Kljub izboljšavam so rezultati v primerjavi s scenarijem 4 nekoliko slabši. Preciznost se je zmanjšala iz 0,74 na 0,57, priklic se je znižal iz 0,75 na 0,57, F1-mera se je poslabšala iz 0,74 na 0,57, medtem ko je klasifikacijska točnost padla iz 0,75 na 0,69. AUC ROC je upadel iz 0,82 na 0,77. Vrednosti metrik in klasifikacijska točnost kažejo, da model deluje zadovoljivo.

Pri scenariju 6 (razdelitev v štiri razrede) so rezultati presegli rezultate iz scenarijev 1–3. Model je bolje razvrstil različne nivoje prioritete, pri čemer so vrednosti preciznosti, priklica in F1-mere ostale na zadovoljivi ravni. Kljub temu je bila uspešnost nekoliko slabša v primerjavi s scenarijem 5 (z razdelitvijo v tri razrede). Preciznost se je zmanjšala iz 0,57 na 0,53, priklic je padel iz 0,57 na 0,54, prav tako se je F1-mera znižala iz 0,57 na 0,53. Klasifikacijska točnost se je zmanjšala iz 0,69 na 0,61. Pri AUC ROC opazimo rahlo izboljšanje z 0,77 na 0,80, medtem ko je AUC PR padla z 0,64 na 0,56. To je še posebej pomembno pri nalogah, kjer so napake pri razvrščanju lahko kritične. V podobnih primerih je lahko koristno poenostaviti razvrščanje in se osredotočiti na ključne kategorije, da bi dosegli optimalno točnost.

Naša raziskava je pokazala, da je zmanjšanje števila razredov prioritet nalog v IT projektih dobra rešitev za izboljšanje rezultatov razvrščanja nalog po prioritetah. Uravnoteženje podatkov in zmanjšanje števila prednostnih razredov nalog sta prispevala k boljšim rezultatom. Ta pristop resnično poenostavi postopek in daje dobre rezultate. Hkrati se zavedamo, da je

njegova uporaba v resničnih projektih lahko omejena zaradi raznolikosti in zapletenosti nalog. Uporaba različnih algoritmov za razvrščanje ni bistveno vplivala na rezultate. Algoritem k-najbližjih sosedov je v primerjavi z drugimi algoritmi razvrščanja pokazal bistveno slabše rezultate. Razlago za to vidimo v kompleksnosti analiziranih podatkov. Izbira algoritma za razvrščanje igra vlogo, vendar je ta vpliv v primerjavi z drugimi dejavniki manj izrazit.

Pomemben dejavnik, ki vpliva na kakovost razvrščanja, ostaja način opisovanja nalog. V večini primerov sam opis naloge IT projekta ni popoln, ni jasen in celo ni zaporeden, v samih opisih je težko zaslediti kakršno koli strukturiranost. Vse to lahko prispeva k slabši kakovosti naučenih klasifikatorjev. Posledično obstaja potreba po standardizaciji opisov, saj lahko jasno in strukturirano besedilo bistveno izboljša natančnost odkrivanja vzorcev in razvrščanja.

6 ZAKLJUČEK

V prispevku smo obravnavali uporabo metod strojnega učenja za samodejno določanje prioritet nalog v IT projektih. V ta namen smo po metodologiji CRISP-DM razvili klasifikacijske modele z uporabo algoritmov naključni gozd, metoda podpornih vektorjev, logistična regresija, gradientno povečanje z drevesi odločanja in k-najbližjih sosedov. Raziskavo smo izvedli na podatkovni množici, ki vsebuje številne zapise projektnih nalog za različne IT projekte iz javno dostopnega vira Zenodo v obliki JSON-datoteke. Pri tem smo naslovili dva problema: neuravnoteženost podatkov in večrazredno klasifikacijo. Oblikovali smo osem scenarijev ter preučili, kako le-ti vplivajo na kakovost klasifikacijskih modelov. Pokazali smo, da uporaba algoritmov strojnega učenja omogoča razvoj klasifikacijskega modela, kar lahko prispeva k učinkovitejšemu vodenju projektov, še posebej v kontekstu velikih količin podatkov in zapletenih struktur nalog.

Raziskava je pokazala, da uravnoteženje podatkov pri učenju modelov strojnega učenja vpliva na doseganje večje natančnosti razvrščanja, kar je še posebej opazno pri velikem številu razredov. Prav tako smo pokazali vpliv števila razredov na uspešnost klasifikacijskega modela, kar tudi sicer predstavlja velik izziv pri transparentnem določanju prioritet nalog in izvajanju projektov. V praksi je tako potrebno poiskati ravnovesje med številom razredov prioritet, ki projektnemu timu še zagotavljajo učinkovito vodenje in izvajanje projektov ter natančnostjo klasifikacijskega modela, s katerim lahko podpremo projektne vodje pri določanju prioritet nalog.

Uporaba strojnega učenja za določanje prioritet nalog predstavlja obetavno področje, ki lahko bistveno izboljša učinkovitost vodenja IT projektov. Rezultati raziskave so pomemben prispevek k praksi avtomatiziranega določanja prioritet in omogočajo boljše upravljanje z nalogami v IT projektih. Prihodnje raziskave se lahko usmerijo v nadaljnji razvoj algoritmov in njihovo integracijo v obstoječe sisteme za vodenje projektov, kar bo omogočilo široko uporabo v poslovnem okolju ter izboljšanje učinkovitosti in uspešnosti projektnega vodenja.

ZAHVALA

Raziskava je finančno podprla Javna agencija za znanstvenoraziskovalno in inovacijsko dejavnost Republike Slovenije v okviru raziskovalnega programa P5-0018.

Literatura

[1] W. Wang in K. Siau, „Artificial Intelligence, Machine Learning, Automation, Robotics, Future of Work and Future of Humanity: A Review and Research Agenda“, *J. Database Manag.*, let. 30, str. 61–79, maj 2022, doi: 10.4018/978-1-6684-6291-1.ch076.

[2] M. Nenni, F. De Felice, C. De Luca, in A. Forcina, „How artificial intelligence will transform project management in the age of digitization: a systematic literature review“, *Manag. Rev. Q.*, apr. 2024, doi: 10.1007/s11301-024-00418-z.
[3] H. Dinendra, C. Rajapakse, in P. Asanka, „Personalized Classification of Non-Spam Emails Using Machine Learning Techniques“, sep. 2022, str. 171–177. doi: 10.1109/SCSE56529.2022.9905110.
[4] R. K. Karn, V. E. Jesi, in S. M. Aslam, „Spam Email Detection Using Machine Learning Integrated In Cloud“, *2023 Int. Conf. Netw. Commun. ICNWC*, Pridobljeno: 28. september 2024. [Na spletu]. Dostopno na: https://www.academia.edu/102418331/SPAM_EMAIL_DETECTION_USING_MACHINE_LEARNING_INTEGRATED_IN_CLOUD
[5] AXELOS Limited, Ur., *Managing successful projects with PRINCE2*, 6th edition. London Norwich: TSO, 2017.
[6] IIBA, „BABOK® Guide Glossary | IIBA®“. Pridobljeno: 12. julij 2024. [Na spletu]. Dostopno na: https://www.iiba.org/knowledgehub/glossary/
[7] PMI, *The standard for project management and a guide to the project management body of knowledge (PMBOK guide)*., Newtown Square, Pennsylvania 19073-3299 USA., julij 2021. [Na spletu]. Dostopno na: https://ibimone.com/PMBOK%207th%20Edition%20(iBIMOne.com).pdf
[8] P. Achimugu, A. Selamat, R. Ibrahim, in M. N. Mahrin, „A systematic literature review of software requirements prioritization research“, *Inf. Softw. Technol.*, let. 56, št. 6, str. 568–585, jun. 2014, doi: 10.1016/j.infsof.2014.02.001.
[9] F. A. Bukhsh, Z. A. Bukhsh, in M. Daneva, „A systematic literature review on requirement prioritization techniques and their empirical evaluation“, *Comput. Stand. Interfaces*, let. 69, str. 103389, mar. 2020, doi: 10.1016/j.csi.2019.103389.
[10] R. Qaddoura, A. Abu srhan, M. Haj Qasem, in A. Hudaib, „Requirements Prioritization Techniques Review and Analysis“, okt. 2017, str. 258–263. doi: 10.1109/ICTCS.2017.55.
[11] Y. Bugayenko *idr.*, „Prioritizing tasks in software development: A systematic literature review“, *PLOS ONE*, let. 18, št. 4, str. e0283838, 2023, doi: 10.1371/journal.pone.0283838.
[12] M. Narendhar in D. K. Anuradha, „Different Approaches of Software Requirement Prioritization“.
[13] D. Cubranic in G. C. Murphy, „Automatic bug triage using text categorization“, SEKE 2004: Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering., 2004, str. 92-97. Pridobljeno: 23. september 2024. [Na spletu]. Dostopno na: https://www.cs.ubc.ca/labs/spl/papers/2004/seke04-bugzilla.pdf
[14] J. Xuan, J. He, Z. Ren, J. Yan, in Z. Luo, „Automatic Bug Triage using Semi-Supervised Text Classification.“, jan. 2010, str. 209–214.
[15] J. Brownlee, *Imbalanced Classification with Python: Better Metrics, Balance Skewed Classes, Cost-Sensitive Learning*. Independently published, 2021.
[16] Yanmin Sun, A. Wong, in M. S. KAMEL, „Classification of imbalanced data: a review“, *Int. J. Pattern Recognit. Artif. Intell.*, let. 23, nov. 2011, doi: 10.1142/S0218001409007326.
[17] F. Thabtah, S. Hammoud, F. Kamalov, in A. Gonsalves, „Data imbalance in classification: Experimental evaluation“, *Inf. Sci.*, let. 513, str. 429–441, mar. 2020, doi: 10.1016/j.ins.2019.11.004.
[18] C. Shearer, „The CRISP-DM Model: The New Blueprint for Data Mining“, let. 5, št. 4, str. 13–22, 2000.
[19] L. Montgomery, C. Lüders, in Prof. Dr. W. Maalej, „The Public Jira Dataset“. Zenodo, 25. januar 2022. doi: 10.5281/zenodo.5901804.
[20] C. Manning, P. Raghavan, in H. Schuetze, *Introduction to Information Retrieval*. Cambridge, England: Cambridge University Press, 2009.
[21] M. Arora, V. Mittal, in P. Aggarwal, „Enactment of tf-idf and word2vec on Text Categorization“, v *Proceedings of 3rd International Conference on Computing Informatics and Networks*, A. Abraham, O. Castillo, in D. Virmani, Ur., Singapore: Springer, 2021, str. 199–209. doi: 10.1007/978-981-15-9712-1_17.
[22] D. Berrar, „Cross-Validation“, 2018. doi: 10.1016/B978-0-12-809633-8.20349-X.

[23] L. Breiman, „Random Forests“, *Mach. Learn.*, let. 45, št. 1, str. 5–32, okt. 2001, doi: 10.1023/A:1010933404324.
[24] C. Cortes in V. Vapnik, „Support-vector networks“, *Mach. Learn.*, let. 20, št. 3, str. 273–297, sep. 1995, doi: 10.1007/BF00994018.
[25] J. Friedman, „Greedy Function Approximation: A Gradient Boosting Machine“, *Ann. Stat.*, let. 29, nov. 2000, doi: 10.1214/aos/1013203451.
[26] M. Collins, R. E. Schapire, in Y. Singer, „Logistic Regression, AdaBoost and Bregman Distances“, *Mach. Learn.*, št. 48(1/2/3), jan. 2002.
[27] T. Cover in P. Hart, „Nearest neighbor pattern classification“, *IEEE Trans. Inf. Theory*, let. 13, št. 1, str. 21–27, jan. 1967, doi: 10.1109/TIT.1967.1053964.
[28] C. C. Aggarwal, *Data Mining: The Textbook*. New York, USA: Springer, 2015.
[29] A. Hevapathige, „Binary and Multi-Class Classification Using Supervised Machine Learning Algorithms and Ensemble Model“, sep. 2021.
[30] M. Majnik in Z. Bosnic, „ROC analysis of classifiers in machine learning: A survey“, *Intell. Data Anal.*, let. 17, str. 531–558, maj 2013, doi: 10.3233/IDA-130592.
[31] J. Czakon, „F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose?“, neptune.ai. Pridobljeno: 14. julij 2024. [Na spletu]. Dostopno na: https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc
[32] A. P. Jawalkar *idr.*, „Early prediction of heart disease with data analysis using supervised learning with stochastic gradient boosting“, *J. Eng. Appl. Sci.*, let. 70, št. 1, str. 122, okt. 2023, doi: 10.1186/s44147-023-00280-y.
[33] A. Alnuaimi in T. Albaldawi, „An overview of machine learning classification techniques“, *BIO Web Conf.*, let. 97, str. 00133, apr. 2024, doi: 10.1051/bioconf/20249700133.
[34] K. Marshall, „How to Use the AUC ROC Curve for the Multi-class Model?“, Deepchecks. Pridobljeno: 14. julij 2024. [Na spletu]. Dostopno na: https://deepchecks.com/question/how-to-use-the-auc-roc-curve-for-the-multi-class-model/
[35] T. Saito in M. Rehmsmeier, „The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets“, *PLoS ONE*, let. 10, št. 3, str. e0118432, mar. 2015, doi: 10.1371/journal.pone.0118432.
[36] J. E. de la Calle, „How and Why I Switched from the ROC Curve to the Precision-Recall Curve to Analyze My Imbalanced…“, Medium. Pridobljeno: 13. julij 2024. [Na spletu]. Dostopno na: https://juandelacalle.medium.com/how-and-why-i-switched-from-the-roc-curve-to-the-precision-recall-curve-to-analyze-my-imbalanced-6171da91c6b8
[37] C. Thrampoulidis, S. Oymak, in M. Soltanolkotabi, „Theoretical Insights Into Multiclass Classification: A High-dimensional Asymptotic View“, v *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020, str. 8907–8920. Pridobljeno: 5. december 2024. [Na spletu]. Dostopno na: https://proceedings.nips.cc/paper_files/paper/2020/hash/6547884cea64550284728eb26b0947ef-Abstract.html

Tatyana Unuchak je magistrica organizatorka informatičarka in strokovnjakinja na področju razvoja spletnih in mobilnih rešitev. Raziskovalni interesi so povezani s strojnim učenjem in z izboljšanjem procesov projektnega vodenja.

Mirjana Kljajić Borštnar je redna profesorica za področje informacijskih sistemov na Fakulteti za organizacijske vede Univerze v Mariboru. Njeno raziskovalno delo je usmerjeno v sisteme za podporo odločanju, odkrivanje znanja v podatkih in strojno učenje ter organizacijsko učenje. Je glavna urednica revije Uporabna informatika, podpredsednica Slovenskega društva INFORMATIKA, sovodja programskih odborov mednarodnega simpozija operacijskih raziskav in Blejske e-konference ter članica izvršnega odbora AI4Slovenia.

Yauhen Unuchak je magister organizator informatik in strokovnjak na področju avtomatiziranega testiranja in testiranja zmogljivosti programske opreme. Raziskovalni interesi so povezani z

avtomatiziranim testiranjem, sistemi menedžmenta kakovosti, korporativnimi informacijskimi sistemi, velepodatki in napovedno analitiko. Eden od avtorjev knjige »IT-Startup: 10 nasvetov za začetnike«.