Predviđanje pogrešaka u programskoj podršci pomoću strojnog učenja

Cvitković, Marko

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I
INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij Računarstvo

PREDVIĐANJE POGREŠAKA U PROGRAMSKOJ PODRŠCI POMOĆU STROJNOG UČENJA

Diplomski rad

Marko Cvitković

Osijek, 2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za ocjenu diplomskog rada na sveučilišnom diplomskom studiju

Ocjena diplomskog rada na sveučilišnom diplomskom studiju

Ime i prezime pristupnika:	Marko Cvitković
Studij, smjer:	Sveučilišni diplomski studij Računarstvo
Mat. br. pristupnika, god.	D1274R, 07.10.2022.
JMBAG:	0165081389
Mentor:	doc. dr. sc. Dražen Bajer
Sumentor:	dr. sc. Mario Dudjak
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	izv. prof. dr. sc. Zdravko Krpić
Član Povjerenstva 1:	dr. sc. Mario Dudjak
Član Povjerenstva 2:	doc. dr. sc. Bruno Zorić
Naslov diplomskog rada:	Predviđanje pogrešaka u programskoj podršci pomoću strojnog učenja
Znanstvena grana diplomskog rada:	**Umjetna inteligencija (zn. polje računarstvo)**
Zadatak diplomskog rada:	U radu je potrebno opisati problem predviđanja pogrešaka u programskoj podršci kao problem nadzirane klasifikacije. Uz to, potrebno je pretražiti literaturu kako bi se utvrdio uobičajeni tijek učenja iz podataka koji opisuju zadani problem, s naglaskom na postupke predobrade skupova podataka te algoritme za klasifikaciju.U praktičnom dijelu rada nužno je implementirati istaknute postupke predobrade te algoritme za klasifikaciju u programskom jeziku po volji. Također, potrebno je napraviti eksperimentalnu analizu na javno dostupnim skupovima podataka
Datum ocjene pismenog dijela diplomskog rada od strane mentora:	17.09.2024.
Ocjena pismenog dijela diplomskog rada od strane mentora:	Izvrstan (5)
Datum obrane diplomskog rada:	30.9.2024.
Ocjena usmenog dijela diplomskog rada (obrane):	Vrlo dobar (4)
Ukupna ocjena diplomskog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije diplomskog rada čime je pristupnik završio sveučilišni diplomski studij:	04.10.2024.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 04.10.2024.

Ime i prezime Pristupnika:	Marko Cvitković
Studij:	Sveučilišni diplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	D1274R, 07.10.2022.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Predviđanje pogrešaka u programskoj podršci pomoću strojnog učenja**

izrađen pod vodstvom mentora doc. dr. sc. Dražen Bajer

i sumentora dr. sc. Mario Dudjak

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

Sadržaj

1. UVOD

Predviđanje pogrešaka u programskoj podršci predstavlja jedan od ključnih izazova za organizacije koje razvijaju sustave. Uz sve veći rast složenosti programske podrške, postoji i povećana potreba za pouzdanim alatima koji mogu unaprijed identificirati potencijalne pogreške. Pravovremeno otkrivanje i ispravljanje pogrešaka doprinosi smanjenju troškova održavanja te poboljšava ukupnu kvalitetu i sigurnost programske podrške. Cilj ovog rada je istražiti primjenu strojnog učenja u zadatku predviđanja pogrešaka u programskoj podršci, s posebnim naglaskom na metode klasifikacije. Kroz analizu povijesnih podataka o pogreškama, strojno učenje omogućuje otkrivanje obrazaca koji pomažu u predviđanju budućih pogrešaka. Međutim, ovaj zadatak dodatno otežavaju izazovi poput neuravnoteženosti klasa te specifičnih karakteristika podataka koji se koriste u analizi. U okviru ovog rada obrađuju se ključni algoritmi strojnog učenja primjenjivi u predviđanju pogrešaka, s naglaskom na klasifikaciju, rukovanje neuravnoteženim podacima i odabir značajki. Posebna pažnja posvećena je tehnikama predobrade podataka koje doprinose povećanju točnosti i učinkovitosti modela. Korišteni skupovi podataka i metode iz literature također su detaljno analizirani kako bi se pružio uvid u trenutno stanje istraživanja.

Struktura rada obuhvaća nekoliko ključnih cjelina. Drugo poglavlje razmatra problem predviđanja pogrešaka u programskoj podršci, klasifikaciju kao zadatak nadziranog učenja te problem neuravnoteženih klasa i unutarnjih karakteristika podataka. Pregled literature daje širi kontekst korištenih skupova podataka, algoritama i tehnika predobrade. U trećem poglavlju prikazano je ostvareno programsko rješenje, dok se u četvrtom provodi eksperimentalna analiza koja uključuje ispitivanje učinka algoritama, odabira značajki i uzorkovanja. Rad završava zaključnim razmatranjima u zadnjem poglavlju.

2. PREDVIĐANJE POGREŠAKA U PROGRAMSKOJ PODRŠCI

U suvremenom razvoju softvera, kvaliteta i pouzdanost programske podrške od presudne su važnosti. Svaka pogreška može rezultirati značajnim financijskim gubicima, narušavanjem ugleda tvrtke ili, u najgorem slučaju, ugrozom sigurnosti korisnika. Sposobnost uočavanja i, ako je moguće, predviđanja pogrešaka stoga je od iznimne važnosti kako bi se mogle pravovremeno otkloniti. Predviđanje pogrešaka u programskoj podršci (engl. *software defect prediction,* SDP) odnosi se na primjenu analitičkih tehnika i algoritama strojnog učenja za identifikaciju dijelova softvera koji su skloni greškama. Ključno je za unapređenje kvalitete softvera, smanjenje troškova održavanja i povećanje zadovoljstva korisnika. Mogućnost predviđanja pogrešaka prvi je korak u stvaranju učinkovitih strategija za njihovo otklanjanje. Za učinkovito predviđanje grešaka koriste se metode strojnog učenja. Ove metode omogućuju izradu modela koji, na temelju različitih obilježja programske podrške, mogu predvidjeti pogreške. Temeljni preduvjet pri korištenju ovih metoda jest raspoloživost kvalitetnih podataka. U nastavku poglavlja, razmotrena je klasifikacija u strojnom učenju te osnovni principi i primjena u softverskoj detekciji pogrešaka. Pored toga, objašnjene su karakteristike skupova podataka te je dan pregled literature s analizom različitih klasifikacijskih algoritama i tehnika predobrade koje se koriste. Također se pruža cjeloviti uvid u problematiku predviđanja pogrešaka u programskoj podršci, od apstraktnih koncepata do primjene odgovarajućih metoda i tehnika.

2.1. Uvod u problem

Programska podrška ima važnu ulogu u našem svakodnevnom životu, od jednostavnih aplikacija na pametnim telefonima do složenih sustava za upravljanje kritičnim infrastrukturnim projektima. No, kako se softverski sustavi razvijaju i postaju sve složeniji, tako raste i mogućnost pojave pogrešaka. Pogreške u programskoj podršci najčešće potječu iz različitih faza razvoja, uključujući fazu dizajna koda. U ovoj fazi, pogreške mogu nastati zbog nesporazuma u zahtjevima korisnika, gdje specifikacije nisu jasno definirane ili su krivo interpretirane od strane programera. Neodgovarajuće arhitektonske odluke, poput loše strukture modula ili nepravilne raspodjele odgovornosti među komponentama, također mogu uzrokovati probleme. Nadalje, pogrešan izbor algoritama može dovesti do problema s radnim učinkom ili netočnim rezultatima. Nedosljednosti u primjeni dizajnerskih obrazaca mogu rezultirati teško održivim i nestabilnim kodom. Zbirni učinak ovih faktora može značajno povećati rizik od kvarova u softverskim sustavima.

Životni ciklus razvoja programske podrške (engl. *software development lifecycle,* SDLC)

definira faze kroz koje softverski sustav prolazi, uključujući planiranje, analizu zahtjeva, dizajn, programiranje, testiranje, objavljivanje i održavanje. Greške u softverskim sustavima mogu nastati u svakoj od ovih faza. U fazi planiranja, greške su često uzrokovane nedostatnom komunikacijom između dionika, što rezultira nejasnim ciljevima. Primjer toga je projekt službe FBI naziva Virtual Case File, gdje su loše definirani ciljevi i česte promjene zahtjeva doveli do propasti projekta [1]. U fazi analize zahtjeva, neodgovarajuća analiza može dovesti do kašnjenja i povećanja troškova. Projekt NextGen za nadzor zračnog prometa naišao je na poteškoće zbog nejasnih zahtjeva, što je uzrokovalo kašnjenja i višestruko prekoračenje proračuna . U fazi dizajna, loša arhitektura može uzrokovati probleme s kompatibilnošću i performansama, kao što se dogodilo kod operacijskog sustava Windows Vista tvrtke Microsoft. Složen i nedovoljno testiran dizajn rezultirao je brojnim problemima, uključujući probleme s kompatibilnošću softvera i hardvera te loše performanse sustava [2]. Tijekom programiranja, ljudske pogreške mogu rezultirati ozbiljnim sigurnosnim propustima. Primjer toga je greška u biblioteci OpenSSL, gdje je mala pogreška u kodu omogućila napadačima pristup osjetljivim informacijama kao što su lozinke i privatni ključevi [3]. Propusti u testiranju mogu dovesti do neuspjeha misija. NASA-in Mars Climate Orbiter izgubljen je zbog nedostatka odgovarajućeg testiranja, gdje je korištena kombinacija standardnih i imperijalnih jedinica, što je rezultiralo neuspjehom misije [4]. U fazi objavljivanja, neodgovorna konfiguracija može uzrokovati ozbiljne probleme. Neuspjeh lansiranja web stranice healthcare.gov rezultat je problema s integracijom i performansama, što je uzrokovalo ozbiljne probleme prilikom lansiranja web stranice [5]. Konačno, u fazi održavanja, nepravovremeno ažuriranje može omogućiti širenje virusa. Napad ucjenjivačkim softverom WannaCry dogodio se zbog neuspjeha u pravovremenom ažuriranju operacijskog sustava, što je omogućilo širenje virusa i nanijelo velike štete [6]. Razumijevanje i prepoznavanje potencijalnih izvora grešaka u svakoj fazi životnog ciklusa razvoja programske podrške ključno je za minimiziranje rizika od kvarova u softverskim sustavima. Takve pogreške mogu dovesti do kvarova ili proizvesti neočekivane rezultate [7,8]. Tako je, primjerice, lansiranje rakete Ariane 5 1996. godine završilo katastrofalnim neuspjehom zbog pogreške u softveru za kontrolu leta. Neispravan algoritam za pretvorbu uzrokovao je prekoračenje vrijednosti, što je rezultiralo eksplozijom rakete nekoliko sekundi nakon lansiranja, uzrokujući gubitak tereta vrijednog oko 370 milijuna dolara [9]. Slična situacija dogodila se s Therac-25 radijacijskim strojem između 1985. i 1987. godine, gdje su serije pogrešaka u softveru uzrokovale prekomjerne doze radijacije, što je rezultiralo smrću najmanje troje pacijenata i ozbiljnim ozljedama nekoliko drugih [10]. Ovi primjeri iz literature naglašavaju ozbiljnost problema koje mogu uzrokovati pogreške u programskoj podršci i važnost rigoroznih postupaka dizajna, kodiranja i testiranja u razvoju softvera. Kako bi se smanjili kvarovi i poboljšala kvaliteta softvera,

koriste se mnoge aktivnosti osiguranja kvalitete softvera koje su sažete u tablici 2.1. Takve aktivnosti obično koštaju oko 80% ukupnog proračuna projekta [11]. Kako bi se minimizirali troškovi, softverski inženjeri žele znati koji softverski moduli sadrže više defekata i prvo pregledati te module. Kao rezultat toga, predložene su tehnike predviđanja pogrešaka u programskoj podršci [12].

Tablica 2.1. Aktivnosti osiguranja kvalitete

Aktivnosti osiguranja kvalitete	**Opis**
Pregled koda	Detaljno ispitivanje izvornog koda od strane programera kako bi se otkrile greške i predložile poboljšanja.
Jedinično testiranje	Testiranje pojedinačnih komponenti ili modula softvera kako bi se osiguralo da rade ispravno.
Integracijsko testiranje	Testiranje kombinacije modula ili komponenti kako bi se osiguralo da međusobno pravilno surađuju.
Sustavno testiranje	Potpuno testiranje integriranog softverskog sustava kako bi se provjerila usklađenost sa specificiranim zahtjevima.
Validacija i verifikacija	Procesi osiguravanja da softver ispunjava svoje zahtjeve i da se razvija prema planu i specifikacijama.
Stres testiranje	Ispitivanje softverskog sustava pod ekstremnim uvjetima kako bi se provjerila njegova stabilnost i otpornost.
Testiranje sigurnosti	Provjera sigurnosnih aspekata softvera kako bi se osiguralo da sustav nije podložan napadima ili ranjivostima.
Analiza statičkog koda	Automatsko ispitivanje izvornog koda bez izvršavanja kako bi se pronašle moguće greške, ranjivosti ili odstupanja od standarda.
Kontinuirana integracija	Redovito integriranje i testiranje koda u središnjem repozitoriju kako bi se odmah otkrile i ispravile greške.
Automatizirano testiranje	Korištenje alata za automatsko izvođenje testova kako bi se ubrzao proces testiranja i osigurala dosljednost.
Testiranje upotrebljivosti	Provjera koliko je softver intuitivan i jednostavan za korištenje krajnjim korisnicima.

Problem predviđanja pogrešaka u programskoj podršci odnosi se na identificiranje dijelova koda koji su najvjerojatnije skloni pogreškama. Identifikacija softverskih komponenti koji su defektni od velike je važnosti jer olakšava daljnji razvoj i održavanje softvera. Iako su mnoge tehnike već predložene u literaturi o predviđanju pogrešaka u programskoj podršci, ovaj problem nije u potpunosti riješen i istraživači se još uvijek fokusiraju na razvoj preciznijih prediktora defekata. Nedavni rezultati pokazuju da bi se istraživači trebali koncentrirati na poboljšanje kvalitete podataka u cjelokupnom procesu razvoja i testiranja softvera kako bi se prevladala ograničenja postojećih modela predviđanja softverskih pogrešaka [13].

Statističke metode tradicionalno su bile često korištene za predviđanje pogrešaka u softveru. Ove metode obuhvaćaju različite tehnike analize podataka kao što su regresijska analiza, analiza varijance, testiranje hipoteza i druge. One se temelje na matematičkim modelima i statističkim testovima za identifikaciju uzoraka i predviđanje pogrešaka. Međutim, danas se češće preferiraju naprednije tehnike rudarenja podataka i strojnog učenja. Algoritmi strojnog učenja omogućuju razvoj modela koji automatski prepoznaju složene obrasce u velikim količinama podataka. Ove tehnike mogu naučiti iz podataka i prilagoditi se raznovrsnim uvjetima, što ih čini sposobnijima za preciznije predviđanje pogrešaka u softveru. Točnost strojnog učenja često je veća jer može efikasnije iskoristiti bogatstvo dostupnih i kvalitetnih podataka, učiti kontekstualne značajke i prilagoditi se promjenjivim uvjetima, za razliku od tradicionalnih statističkih metoda koje često zahtijevaju predefinirane pretpostavke i jednostavnije modele. Predviđanje pogrešaka može se promatrati kao klasifikacijski problem u domeni nadziranog strojnog učenja, gdje klase najčešće predstavljaju postojanje, odnosno izostanak pogreške u nekom programskom kodu. Prikupljanje povijesnih podataka o pogreškama, uključujući informacije o prethodnim verzijama softvera i prijavljenim greškama, ključno je za rješavanje problema predviđanja pogrešaka. Nakon prikupljanja, koriste se tehnike strojnog učenja za izgradnju modela na temelju ovih podataka. Takav model potom se primjenjuje na novi, nepoznati softverski kod kako bi se predvidio prisutnost pogreške ili vjerojatnost novih pogrešaka.

2.2. Klasifikacija kao zadatak nadziranog strojnog učenja

Klasifikacija je jedan od temeljnih zadataka nadziranog strojnog učenja koji za cilj ima kategorizirati podatke u unaprijed definirane klase. Cilj klasifikacije je izgraditi model koji može predvidjeti oznake za nove, nepoznate primjere na temelju naučenih obrazaca iz dostupnog skupa podataka. Razumijevanje osnovnih pojmova i koraka u klasifikaciji, kao i mjera za vrednovanje učinkovitosti modela, ključno je za uspješnu primjenu klasifikacijskih modela u raznim

područjima. Proces uključuje prikupljanje i predobradu podataka, odabir značajki, treniranje i testiranje modela. Nadzirano strojno učenje podrazumijeva rad s označenim podacima, pri čemu svaki primjer u skupu podataka ima pridruženu klasu.

Skup podataka (engl. *dataset*) ključna je komponenta u strojnom učenju, pružajući temelj za primjenu algoritama za klasifikaciju. Podatke često prikazujemo u obliku matrice $\boldsymbol{X}_{N\times d}$, gdje je N broj primjera, a d broj značajki kao što je prikazano na slici 2.1. Primjer (engl. *instance*) je pojedinačna podatkovna točka u skupu podataka, a svaki primjer opisan je pomoću značajki i pripadajuće oznake klase. Svaki primjer predstavlja jedan redak matrice $\boldsymbol{X}$. Značajke (engl. *features*) su atributi ili karakteristike koje opisuju svaki primjer u skupu podataka. Primjerice, značajke mogu uključivati broj linija koda u modulu, broj promjena koda, broj programera koji su radili na modulu, kompleksnost koda, broj prethodnih grešaka u modulu te vrijeme koje je proteklo od zadnje promjene koda. Značajke za primjer $\boldsymbol{x_i}$ prikazuju se kao vektor: $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \ldots, x_{id})$, gdje svaki element vektora $\boldsymbol{x_i}$ predstavlja određenu značajku koja opisuje *i*-ti primjerak.

$$X = \left(\begin{array}{cccc|c} x_{11} & x_{12} & \cdots & x_{1d} & l_1 \\ x_{21} & x_{22} & \cdots & x_{2d} & l_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nd} & l_N \end{array}\right)$$

Slika 2.1. Grafički prikaz skupa podataka

Klasifikator je algoritam koji stvara modele koji primjerima dodjeljuju oznaku klase. Među najčešće korištenim klasifikatorima su logistička regresija (engl. *logistic regression*), stablo odluke (engl. *decision tree,* DT*)*, stroj potpornih vektora (engl. *support vector machine,* SVM) i neuronske mreže (engl. *neural networks*) [14-16]. Logistička regresija koristi logističku funkciju za predviđanje vjerojatnosti pripadnosti primjera određenoj klasi. Stabla odlučivanja koriste hijerarhijsku strukturu gdje svaki unutarnji čvor predstavlja odluku temeljenu na vrijednosti značajke, a svaki list predstavlja klasu. Stroj potpornih vektora koristi hiperravninu za razdvajanje primjera različitih klasa u višedimenzionalnom prostoru. Neuronske mreže su kompleksni modeli inspirirani radom ljudskog mozga, posebno učinkoviti za složene klasifikacijske zadatke. Proces primjene klasifikatora uključuje nekoliko ključnih koraka:

1. Prikupljanje podataka: Prvi korak je podjela prikupljenih podataka u dva podskupa: jedan za treniranje i drugi za testiranje klasifikatora. Ova podjela osigurava da se model trenira na jednom dijelu podataka, dok se njegovo konačno performanse procjenjuju na neovisnom skupu, čime se izbjegava pretreniranje i osigurava točnija procjena modela na novim, neviđenim podacima.
2. Predobrada podataka: Uključuje čišćenje podataka, rukovanje nedostajućim vrijednostima, normalizaciju značajki i transformaciju podataka u prikladan format.
3. Odabir značajki: Obuhvaća izbor najrelevantnijih značajki koje će se koristiti za treniranje klasifikatora.
4. Odabir modela: Podešavanje hiperparametara klasifikatora s ciljem odabira prikladnog modela.
5. Treniranje modela: Odabrani model klasifikatora uči iz podataka identificirajući obrasce i relacije između značajki i ciljanih klasa.
6. Testiranje modela: Konačno testiranje se provodi pomoću podskupa za testiranje kako bi se ocijenila učinkovitost modela.

U procjeni učinkovitosti često se koriste sljedeće mjere performansi: broj ispravno klasificiranih pozitivnih primjera (engl. *True Positives*, TP), broj ispravno klasificiranih negativnih primjera (engl. *True Negatives*, TN), broj pogrešno klasificiranih negativnih primjera kao pozitivnih (engl. *False Positives*, FP) te broj pogrešno klasificiranih pozitivnih primjera kao negativnih (engl. *False Negatives*, FN). Mjere performanse se mogu vrlo lako prikazati matricom. Matrica zabune (engl. *confusion matrix*) je tablica koja prikazuje broj točnih i netočnih predikcija klasifikatora za svaku klasu, omogućavajući detaljniju analizu izvedbe modela. U slučaju binarne klasifikacije, matrica izgleda kao što je prikazano na slici 2.2.

Stvarne / Predviđene	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Slika 2.2. Matrica zabune

Iz ovih vrijednosti mogu se izračunati različite mjere performansi. Točnost (engl. *accuracy*) predstavlja omjer ispravno klasificiranih primjera prema ukupnom broju primjera,

$$\text{Točnost} = \frac{(\text{TP}+\text{TN})}{(\text{FP}+\text{FN}+\text{TP}+\text{TN})}. \tag{2-1}$$

Preciznost (engl. *precision*) mjeri omjer ispravno predviđenih pozitivnih primjera prema ukupnom broju primjera predviđenih kao pozitivni,

$$\text{Preciznost} = \frac{\text{TP}}{(\text{FP}+\text{TP})}\ . \tag{2-2}$$

Odziv (engl. *recall*) predstavlja omjer ispravno predviđenih pozitivnih primjera prema ukupnom broju stvarno pozitivnih primjera,

$$\text{Odziv} = \frac{\text{TP}}{(\text{FN}+\text{TP})}. \tag{2-3}$$

F1 mjera (engl. *F1 score*) je harmonijska sredina preciznosti i odziva,

$$\text{F1 mjera} = 2 * \frac{(\text{Preciznost}*\text{Odziv})}{(\text{Preciznost}+\text{Odziv})}, \tag{2-4}$$

Ove mjere su dizajnirane za binarne probleme klasifikacije. No, mogu se primijeniti i na višeklasne probleme koji su raznim shemama dekompozicije razloženi na više binarnih problema. Kao primjer, pretpostavlja se da postoji binarni klasifikator za klasifikaciju softverskih modula kao "pogreška" ili "nije pogreška". Rezultati modela mogu se prikazati pomoću matrice zabune, iz koje možemo izračunati mjere performansi poput točnosti, preciznosti, odziva i F1 mjera.

Actual / Predicted	Pogreška	Nije pogreška
Pogreška	50	10
Nije pogreška	5	100

Slika 2.3. Primjer matrice zabune

1. Točnost

$$\text{Točnost} = \frac{TP + TN}{FP + FN + TP + TN} = \frac{50 + 100}{5 + 100 + 5 + 10} \approx 0.909$$

2. Preciznost

$$\text{Preciznost} = \frac{TP}{FP + TP} = \frac{50}{50 + 5} \approx 0.909$$

3. Odziv

$$\text{Odziv} = \frac{TP}{FN + TP} = \frac{50}{50 + 10} \approx 0.833$$

4. F1 mjera

$$\text{F1 mjera} = 2 \times \left(\frac{\text{Preciznost} \times \text{Odziv}}{\text{Preciznost} + \text{Odziv}}\right) = 2 \times \left(\frac{0.909 \times 0.833}{0.909 + 0.833}\right) \approx 0.870$$

Slika 2.4. Primjer izračuna mjera za vrednovanje performansi klasifikacijskog modela

Model je ocijenjen visokom točnošću od 90.9%, što pokazuje da su softverski moduli općenito dobro klasificirani kao pogreške ili ne-pogreške. Također, postignuta je visoka preciznost od 90.9%, što znači da kada je modul klasificiran kao pogreška, velika je vjerojatnost da je taj modul zaista pogreška. Dobar odziv modela iznosi 83.3%, što ukazuje na to da su stvarne pogreške dobro prepoznate, iako je 16.7% pogrešaka propušteno. F1 mjera od 87% pokazuje dobru uravnoteženost između preciznosti i odziva, što znači da su pogreške dobro prepoznate uz minimalan broj pogrešno klasificiranih modula. Ove mjere performansi omogućuju sveobuhvatan pregled učinkovitosti klasifikacijskog modela i mogu se koristiti za daljnje poboljšanje modela prema specifičnim potrebama.

2.2.1. Problem neuravnoteženosti klasa

Klasifikacija u nadziranom strojnim učenju često se suočava s problemom neuravnoteženosti klasa (engl. *class imbalance*). Ovaj problem nastaje kada u skupu podataka postoji značajna razlika u broju primjera između klasa. Na primjer, u medicinskoj dijagnostici može biti mnogo više zdravih pacijenata nego onih s određenom bolešću, što rezultira neuravnoteženim podacima [17]. Neuravnoteženost klasa često se i javlja prirodno u mnogim stvarnim skupovima podataka. U

kriminalističkim istragama, broj slučajeva prijevare može biti znatno manji od broja slučajeva bez prijevare. Slično, u detekciji kvara opreme, kvarovi su rijetki u odnosu na normalne radne uvjete [18]. Posljedice neuravnoteženosti klasa uključuju sklonost modela prema većinskoj klasi, gdje model može naučiti da uvijek predviđa većinsku klasu jer će tako imati visoku ukupnu točnost, ali će biti neučinkovit za predviđanje manjinske klase. Također, smanjena učinkovitost za kritične zadatke, kao što je točno predviđanje manjinske klase u dijagnosticiranju bolesti, može dovesti do ozbiljnih posljedica [19]. Problem neuravnoteženosti klasa često je prisutan i u području predviđanja pogrešaka u programskoj podršci, gdje modeli često moraju rješavati izazove nejednakog broja grešaka u softverskim projektima.

Postoji nekoliko pristupa za rješavanje problema neuravnoteženosti klasa. Preuzorkovanje manjinske klase (engl. *oversampling*) uključuje dodavanje instanci manjinske klase kako bi se uravnotežio broj instanci između klasa. Jedna od popularnih tehnika je SMOTE (engl. *Synthetic Minority Over-sampling Technique*), koja sintetički stvara nove primjere manjinske klase kombiniranjem postojećih primjera [20]. Poduzorkovanje većinske klase (engl. *undersampling*) uključuje smanjenje broja instanci većinske klase kako bi se uravnotežio broj instanci, što može uključivati nasumično uklanjanje primjera većinske klase, ali to može dovesti do gubitka važnih informacija [21]. Pristupi na razini podataka mijenjaju distribuciju preuzorkovanjem ili poduzorkovanjem instanci kako bi se postigla ravnoteža među klasama. S druge strane, pristupi na razini algoritama prilagođavaju način rada modela kako bi bolje rukovali neuravnoteženim podacima, primjerice, dodjeljivanjem različitih težina klasama ili promjenom kriterija za klasifikaciju.

Standardne metrike poput točnosti nisu prikladne za procjenu učinkovitosti modela kada su podaci neuravnoteženi. Umjesto toga, koriste se metrike koje bolje odražavaju sposobnost modela da klasificira primjere manjinske klase, uključujući preciznost i odziv [18]. ROC-AUC (engl. *Receiver Operating Characteristic - Area Under Curve*) je mjera koja ocjenjuje sposobnost modela da razlikuje između različitih klasa. ROC krivulja prikazuje omjer točno predviđenih pozitivnih primjera u odnosu na lažno pozitivne primjere pri različitim pragovima klasifikacije. AUC predstavlja površinu ispod ROC krivulje, gdje vrijednost bliska 1 ukazuje na visokokvalitetni model koji dobro razlikuje između klasa, dok vrijednost bliska 0.5 sugerira da model nije bolji od slučajnog odabira. Ova mjera je neovisna o distribuciji klasa i stoga pruža objektivnu procjenu performansi modela [19].

Problem neuravnoteženosti klasa predstavlja izazov u nadziranom strojnom učenju jer može značajno utjecati na performanse klasifikacijskih modela. Razumijevanje uzroka i posljedica neuravnoteženosti, kao i primjena odgovarajućih tehnika za rješavanje ovog problema, ključni su

za razvoj učinkovitih i pouzdanih modela. Kroz kombinaciju preuzorkovanja, izmjene algoritama i korištenja naprednih metoda vrednovanja, moguće je postići uravnoteženije i preciznije klasifikacijske modele, čime se povećava njihova primjenjivost u stvarnim situacijama gdje je neuravnoteženost česta pojava.

2.2.2. Unutarnje karakteristike skupova podataka

Unutarnje karakteristike skupova podataka predstavljaju ključne elemente koji definiraju strukturu i sadržaj podataka, omogućujući bolje razumijevanje njihove kompleksnosti i specifičnosti, što je ključno za odabir i prilagodbu algoritama za klasifikaciju. Prva važna karakteristika je broj primjeraka u skupu podataka. Veći broj primjeraka često doprinosi boljoj generalizaciji modela, dok manji skupovi podataka mogu dovesti do pretreniranja (engl. *overfitting*). Druga bitna karakteristika je broj značajki koje opisuju svaki primjerak. Značajke mogu biti numeričke, kategorijske ili binarne, a njihov broj i tip određuju kompleksnost analize. Nejednaka raspodjela klasa, odnosi se na omjer primjera između različitih klasa, često izražen kao omjer neuravnoteženosti (engl. *imbalance ratio*, IR). Neuravnoteženi skupovi podataka, gdje su klase neravnomjerno zastupljene, mogu predstavljati izazov za mnoge algoritme strojnog učenja.

Izvor podataka, odnosno podrijetlo skupa podataka, također je važan aspekt i može uključivati javne repozitorije, vlastite izvore ili simulirane podatke. Izvor podataka može utjecati na njegovu pouzdanost, kvalitetu i relevantnost. Tipovi značajki, kao što su numeričke, kategorijske, tekstualne ili vremenske serije, mogu značajno utjecati na odabir tehnika predobrade te modela. Normalizacija i standardizacija podataka su svođenja vrijednosti značajki na slične raspone ili distribucije, čime se poboljšava učinkovitost algoritama za strojno učenje. Nedostajuće vrijednosti su još jedan izazov, a prisutnost nedostajućih vrijednosti u skupu podataka zahtijeva metode za njihovo rješavanje, poput uklanjanja instanci. Redundantnost i korelacija među značajkama također su ključne karakteristike, jer prisustvo redundantnih ili visoko koreliranih značajki može utjecati na performanse modela, zahtijevajući primjenu tehnika smanjenja dimenzionalnosti ili odabira značajki. Osim ovih osnovnih karakteristika, unutarnje karakteristike skupova podataka također obuhvaćaju pojave poput šuma, malih disjunkata i preklapanja klasa.

Preklapanje klasa (engl. *class overlap*) nastaje kada primjeri različitih klasa imaju slične vrijednosti značajki, što otežava modelu da ih pravilno razdvoji. Ovo se često događa u kompleksnim skupovima podataka gdje granice između klasa nisu jasno definirane. Preklapanje klasa može se mjeriti korištenjem metoda kao što su koeficijent preklapanja i analiza komponenti varijance [22]. Jedan od pristupa rješavanju preklapanja klasa je korištenje nelinearnih

klasifikatora kao što su neuronske mreže i podrška vektorskih strojeva (SVM), koji imaju sposobnost učenja nelinearnih granica između klasa.

Mali disjunkti (engl. *small disjuncts*) odnose se na manje grupe unutar skupa podataka koje imaju specifične karakteristike. Ovi mali disjunkti mogu biti problematični jer model često favorizira veće grupe podataka, ignorirajući male disjunkte. Mali disjunkti mogu se mjeriti analizom distribucije podataka i identifikacijom malih skupova unutar klasa [23]. Pristupi za rješavanje ovog problema uključuju tehnike preuzorkovanja i korištenje ansambla klasifikatora (engl. *ensemble methods*) kao što su slučajne šume i potiskivanje (engl. *boosting*), koji bolje prilagođavaju učenje različitim grupama podataka.

Šum **(**engl. *Noise***)** u podacima odnosi se na prisutnost pogrešnih ili nereprezentativnih primjera koji mogu ometati proces učenja. Šum može biti rezultat ljudskih grešaka, senzornih pogrešaka ili drugih nepredviđenih faktora. Šum se može mjeriti pomoću statističkih metoda za detekciju odstupanja i analiza odstupanja (engl. *outlier detection*) [24]. Tehnike za rješavanje šuma uključuju predobradu podataka, kao što su filtriranje šuma i korištenje robusnih algoritama koji su manje osjetljivi na šum, poput robustnih regresijskih metoda i metoda temeljenih na teoriji informacija.

Vrednovanje i mjerenje unutarnjih karakteristika podataka ključna je za razumijevanje kompleksnosti zadatka klasifikacije. Na primjer, koeficijent preklapanja može kvantificirati koliko se klase preklapaju, dok analiza distribucije može otkriti prisutnost malih disjunkata. Mjerenje šuma može uključivati analizu varijance i identifikaciju odstupanja pomoću metoda kao što su z-score i Mahalanobisova udaljenost [25]. Preklapanje klasa, mali disjunkti i šum nisu poželjne karakteristike jer nepovoljno utječu na performanse klasifikacijskih modela. Preklapanje klasa može dovesti do zbunjenosti modela prilikom razdvajanja klasa, što rezultira većim brojem pogrešnih klasifikacija. Mali disjunkti, koji predstavljaju rijetko prisutne podskupove podataka unutar klase, mogu uzrokovati da model loše generalizira, jer su ti podskupovi nedovoljno zastupljeni za učenje pravilnih obrazaca. Šum, koji se manifestira kao varijanca ili odstupanja u podacima, može smanjiti točnost modela jer model može naučiti obrasce koji nisu reprezentativni za stvarne podatke, što vodi do pretreniranja i loše generalizacije na novim podacima. Pristupi ublažavanju njihovih nepoželjnih učinaka često uključuju kombinaciju predobrade podataka, prilagodbe algoritama i korištenja naprednih metoda. Na primjer, predobrada može uključivati tehnike čišćenja podataka, dok prilagodbe algoritama mogu uključivati modificiranje kriterija za podjelu stabla odlučivanja ili primjenu prilagođenih funkcija gubitka. Korištenje ansambla, kao što su grupiranje (engl. *bagging*) i potiskivanje (engl. *boosting*), može pomoći u smanjenju utjecaja šuma i poboljšanju robusnosti modela.

Razumijevanje unutarnjih karakteristika podataka i primjena odgovarajućih tehnika za njihovo rješavanje od suštinskog su značaja za postizanje visoke učinkovitosti klasifikacijskih modela. Kroz pažljivu analizu i primjenu naprednih metoda, moguće je prevladati izazove koje donose preklapanje klasa, mali disjunkti i šum, čime se povećava točnost i pouzdanost modela u raznim stvarnim aplikacijama.

2.3. Pregled literature

U ovom poglavlju objašnjeni su skupovi podataka i tehnike strojnog učenja koje se koriste za predviđanje grešaka u softveru. Skupovi podataka pružaju informacije o softverskim modulima, uključujući metrike koda i povijest grešaka. Tehnike strojnog učenja variraju u brzini, složenosti i potrebama za resursima. Njihova učinkovitost ocjenjuje se kroz metrike poput točnosti i preciznosti. U poglavlju su analizirani izazovi poput neuravnoteženosti klasa i potrebe za optimizacijom hiperparametara, s ciljem pružanja pregleda dostignuća i izazova u predviđanju grešaka u softveru.

2.3.1. Korišteni skupovi podataka

Najčešće korišteni skupovi podataka za SDP dolaze iz javnih repozitorija kao što su NASA MDP (Metrics Data Program) [26] i PROMISE [27]. Ovi skupovi podataka sadrže informacije o softverskim modulima, uključujući detalje o greškama koje su se pojavile u tim modulima. Ovi repozitoriji omogućuju istraživačima pristup bogatim i raznolikim skupovima podataka koji su ključni za treniranje i vrednovanje modela za predviđanje pogrešaka. NASA MDP sadrži podatke o različitim softverskim projektima razvijenim unutar NASA-e. To uključuje projekte koji se odnose na svemirske misije, satelite, letjelice, sustave za kontrolu leta i druge sofisticirane sustave visoke pouzdanosti. Podaci u NASA MDP-u obuhvaćaju metrike koda kao što su broj linija koda, broj funkcija, kompleksnost koda te informacije o broju prijavljenih grešaka u tim projektima. PROMISE repozitorij također nudi slične podatke, ali pokriva širi spektar softverskih projekata iz različitih industrija. Ovdje se mogu naći projekti koji obuhvaćaju softver za financijske sustave, telekomunikacije, zdravstvo, obrazovanje i druge sektore. Informacije o metrikama koda i podacima o greškama uključene su radi podrške istraživanjima u području predviđanja grešaka i analize performansi softvera.

Stvaranje takvih skupova podataka uključuje prikupljanje i bilježenje detaljnih informacija tijekom cijelog životnog ciklusa razvoja softvera. To uključuje prikupljanje metrika koda, podataka o promjenama koda, izvještaja o greškama i informacija o testiranju. Svaki put kad se

pronađe i ispravi defekt, bilježi se detaljan zapis koji uključuje vrstu defekta, mjesto u kodu gdje je defekt pronađen, vrijeme otkrivanja i vrijeme potrebno za ispravak.

Skupovi podataka za SDP obično sadrže sljedeće značajke:

1. Metrike koda (engl. *code metrics*):

- Broj linija koda (engl. *Lines of Code* , LOC*)***:** Broj linija koda u modulu.
- Ciklomatska složenost (engl. *Cyclomatic Complexity*)**:** Mjera složenosti koda koja se koristi za određivanje broja neovisnih putanja kroz program. Računa se korištenjem grafa kontrolnog toka programa, gdje čvorovi predstavljaju blokove koda, a bridovi predstavljaju putanje kontrolnog toka. Ciklomatska složenost računa se kao

$$V(G) = E - N + 2P. \tag{2-5}$$

  U grafu kontrolnog toka, E predstavlja broj bridova, N je broj čvorova, a P označava broj povezanih komponenti (za jedan program, P je obično 1).

- Halstead metrike (engl. *Halstead Metrics***)**: Metrike koje mjere složenost koda pomoću operatora i operanada:
  - Broj jedinstvenih operatora (η1)
  - Broj jedinstvenih operanada **(**η2**)**
  - Ukupan broj operatora (N1)
  - Ukupan broj operanada (N2)

  Koristi niz formula kako bi kvantitativno izrazila različite elemente složenosti programske implementacije. Ove formule povezuju različite mjerne pojmove kao što su rječnik programa (η), duljina programa (N), volumen (V), složenost (D), napor (E), vrijeme za implementaciju (T) i broj isporučenih grešaka (B).

  1. Rječnik programa (η):

$$\eta = \eta 1 + \eta 2 . \tag{2-6}$$

  2. Duljina programa (N)**:**

$$N = N1 + N2 . \tag{2-7}$$

3. Volumen (V)**:** Volumen programa mjeri veličinu implementacije algoritma

$$V = N\log 2\eta \ . \tag{2-8}$$

4. Složenost (D)**:** Složenost mjeri kompleksnost programa u smislu potrebnog napora za pisanje ili razumijevanje koda

$$D = \frac{\eta 1}{2} \cdot \frac{N2}{\eta 2} \ . \tag{2-9}$$

5. Napor (E)**:** Napor predstavlja količinu mentalne aktivnosti potrebne za prevođenje programa u ispravnu implementaciju

$$E = D \cdot V \ . \tag{2-10}$$

6. Vrijeme za implementaciju (T**)**: Procjenjuje vrijeme potrebno za implementaciju programa, uz pretpostavku prosječne brzine od 18 sekundi po jedinici mentalnog napora

$$T = \frac{E}{18} \ . \tag{2-11}$$

7. Broj isporučenih grešaka (B)**:** Procjenjuje broj grešaka u programu na temelju njegovog volumena

$$B = \frac{E^{\frac{2}{3}}}{3000} \ . \tag{2-12}$$

- Ulaz/Izlaz (engl. *Fan-in/Fan-out*)**:** Mjeri broj modula koji pozivaju neki modul (ulaz) i broj modula koje taj modul poziva (izlaz).
- Dubina stabla nasljeđivanja (engl. *Depth of Inheritance Tree - DIT*)**:** Dubina nasljednog stabla, mjeri koliko je klasa duboko u nasljednoj hijerarhiji.

  Broj podklasa (engl. *Number of Children - NOC*)**:** Broj podklasa koje direktno nasljeđuju neku klasu.

2. Metrike zavisnosti (engl. *dependency metrics*):

- Povezanost između objekata (engl. *Coupling Between Objects - CBO*)**:** Broj klasa s kojima je neka klasa povezana.
- Odgovor za klasu (engl. *Response for a Class - RFC*)**:** Broj metoda koje se mogu potencijalno pozvati kao odgovor na poruku upućenu objektu.
- Nedostatak kohezije metoda (engl. *Lack of Cohesion in Methods - LCOM*)**:** Mjera koliko su metode klase međusobno povezane.

3. Procesne metrike (engl. *process metrics*):

- Broj promjena (engl. *Number of Changes***)**: Broj promjena koje je modul prošao.
- Promjene koda (engl. *Code Churn*)**:** Količina dodanog, izmijenjenog i obrisanog koda tijekom vremena.
- Broj grešaka (engl. *Number of Defects*)**:** Broj prijavljenih grešaka za modul.
- Starost koda (engl. *Age of Code*)**:** Vrijeme od zadnje promjene koda.

Ovi podaci omogućavaju istraživačima da razvijaju i testiraju modele za predikciju softverskih grešaka koristeći razne karakteristike koje opisuju složenost koda, količinu promjena, povezanost između modula i druge faktore koji mogu utjecati na pojavu grešaka.

2.3.2. Korišteni algoritmi strojnog učenja

Rješavanje problema predikcije softverskih pogrešaka kao klasifikacijskog problema postalo je bitno područje istraživanja u programskom inženjerstvu. Razni autori su koristili različite pristupe, metode strojnog učenja i tehnike predobrade podataka kako bi unaprijedili preciznost predikcija. Jedan od najranijih algoritama korištenih za SDP je naivni Bayes (engl. *naive Bayes*, NB). Zbog svoje jednostavnosti i brzine, NB je popularan, ali je često manje precizan od složenijih metoda. Istraživanja pokazuju da se NB može poboljšati integracijom drugih tehnika kao što su asocijacijska pravila i logistička regresija. Na primjer, kombinacija NB-a s drugim metodama često rezultira poboljšanim performansama, no osnovna pretpostavka o neovisnosti atributa ostaje ograničenje ovog pristupa [13,25,28–33]. Nadalje DT i njihova proširenja poput slučajnih šuma (engl. *Random Forests*) također su široko korištena za SDP. Ovi algoritmi su korisni zbog svoje interpretabilnosti i sposobnosti rukovanja nelinearnim odnosima među podacima. Međutim, problemi pretreniranja mogu se pojaviti ako se ne koriste pravilne tehnike regularizacije. Slučajne šume, zbog svoje strukture, pružaju bolju otpornost na pretreniranje te su često preferirane u

problemima koje zahtijevaju kompleksniju analizu i obradu podataka [14-16]. Visoku točnost u raznim istraživanjima zbog svoje sposobnosti da pronađe nelinearne granice između klasa pokazao je stroj potpornih vektora. Ipak, stroj potpornih vektora zahtijeva pažljivo podešavanje hiperparametara i može biti računalno intenzivni za velike skupove podataka [14-16,20, 28,31,33]. Uz spomenute metode, ansambl metode također su često korištene u istraživanjima. To su skup tehnika strojnog učenja koje kombiniraju predikcije više modela kako bi se poboljšala ukupna performansa predikcije. Ideja iza ansambl metoda je da kombiniranjem različitih modela, koji mogu imati različite pristranosti i varijabilnosti, možemo postići bolju točnost i robusnost nego što bi to mogao postići bilo koji pojedinačni model. Ansambl metode se oslanjaju na različite pristupe za stvaranje i kombiniranje predikcija, kao što su grupiranje, potiskivanje i slaganje.

Pregled literature pokazuje da su različiti autori koristili razne algoritme u različitim situacijama od kojih su se neki pokazali uspješnijima od drugih. Primjerice, naivni Bayes i slučajne šume su algoritmi koji pružaju brza i pouzdana rješenja za jednostavnije probleme, dok neuronske mreže nude mogućnosti za složenije zadatke, ali uz znatno veće troškove resursa. Ansambli su se pokazali pokazale su se vrlo učinkovitima, ali zahtijevaju pažljivo podešavanje i značajnu računalnu snagu. Pandey i suradnici [15] ističu važnost kvalitetnih podataka i optimizacije modela za postizanje boljih rezultata, dok različite tehnike strojnog učenja kao što su slučajne šume, SVM i neuronske mreže pružaju različite prednosti i izazove u primjeni na predviđanje pogrešaka u programskoj podršci [34]. Nadalje, u [35] Jin naglašava značaj integriranje domenskog znanja za unaprjeđenje prediktivne točnosti modela u kontekstu različitih projekata, koristeći SVM s intergiranjem znanja o problemu, ističući izazove poput optimizacije parametara i potrebe za velikim količinama kvalitetnih podataka za treniranje. Optimizacija hiperparametara često je ključna za postizanje maksimalnih performansi svih algoritma strojnog učenja.

2.3.3. Korištene tehnike predobrade skupova podataka

Korištene tehnike predobrade skupova podataka istražuje različite pristupe koji se često koriste za poboljšanje performansi modela u analizi pogrešaka u programskoj podršci. Mnoge studije koriste normalizaciju kako bi osigurale da svi atributi imaju jednak utjecaj na model. To je posebno važno za algoritme koji su osjetljivi na mjerilo podataka poput algoritama (primjerice, SVM) temeljenih na udaljenosti.

Tehnike poput analize glavnih komponenata (engl. *Principal Component Analysis,* PCA) i odabira značajki koriste se za smanjenje broja atributa i time smanjenje kompleksnosti modela. Ovo može poboljšati učinkovitost i točnost modela [14,15]. Različiti filtri, ugrađene metode i

omotači su najčešće korištene u istraživanju predviđanja pogrešaka u programskoj podršci. Primjerice, Jureczko i Madevski su u [36] istraživali filtre za rangiranje značajki u predviđanju pogrešaka u programskoj podršci, što im omogućava da identificiraju najvažnije značajke za predviđanje grešaka. S druge strane, su He i Zhang u [37] istraživali omotače kao što su genetski algoritmi za traženje optimalnog podskupa značajki. Nadalje Khoshgoftaar je u [38] kombinirao filtre i omotače kako bi prilagodio skup značajki za precizno predviđanje pogrešaka u programskoj podršci. Zatim tehnike poput preuzorkovanja manjinske klase (SMOTE) i poduzorkovanja većinske klase koriste se za stvaranje uravnoteženih skupova podataka, što poboljšava performanse klasifikatora [15,28-30].

Pregled literature pokazuje da su različiti autori koristili širok spektar metoda strojnog učenja i tehnika predobrade kako bi riješili problem predviđanja pogrešaka u programskoj podršci. Y.Ma et al. su u [31] istražili različite mjere složenosti podataka i njihov utjecaj na učinkovitost klasifikacijskih algoritama, koristeći mjere poput Fisherovog diskriminantnog omjera (F1) i volumena preklapanja (F2) za vrednovanje odvojenosti klasa u prostoru značajki. G. Czibula koristi u [13] metode rudarenja asocijativnih pravila za identifikaciju potencijalnih defekata u softverskom kodu, kombinirajući algoritme s tehnikama strojnog učenja za poboljšanje točnosti predikcije defekata.

Nadalje I. Arora u [28] identificira nekoliko ključnih problema koji ometaju učinkovitost modela za predviđanje pogrešaka u programskoj podršci, uključujući potrebu za identificiranjem pouzdanih atributa, razvijanjem standardnih mjera za procjenu performansi, te rješavanjem problema klasnog nesrazmjera. U [33] Błaszczyński ističe važnost lokalnih karakteristika podataka i prilagodbe tehnika prema tim karakteristikama kako bi se poboljšala performansa klasifikatora na neuravnoteženim skupovima podataka, koristeći metode poput ponovnog uzrokovanja, generiranja sintetičkih podataka i učenja osjetljivog na troškove.

2.4. Kritički osvrt

Iz pregleda literature, mogu se uočiti ključni izazovi i nesuglasice u području predviđanja pogrešaka u programskoj podršci pomoću strojnog učenja. Prvi problem odnosi se na izbor algoritama i postupaka predobrade podataka. Nema jasnog konsenzusa o tome koji algoritmi najbolje odgovaraju ovom problemu. Različiti istraživači koriste različite tehnike strojnog učenja, poput logističke regresije, stabala odlučivanja, stroja potpornih vektora i neuronskih mreža, svaki s različitim stupnjem uspjeha. Raznolikost pristupa može se pripisati različitim skupovima podataka i metodologijama vrednovanja koje se koriste u različitim studijama. Ovakva situacija

otežava usporedbu rezultata i identificiranje najboljih praksi.

Još jedan značajan problem je neuravnoteženost klasa u skupovima podataka, gdje je broj primjera jedne klase znatno veći od broja primjera druge klase. Iako su predložene različite tehnike za rješavanje ovog problema, kao što su preuzorkovanje ili poduzorkovanje, ne postoji jedinstveno rješenje. Ovaj problem dodatno kompliciraju činjenica da različiti skupovi podataka imaju različite karakteristike koje negativno utječu na izvedbu modela. Kvaliteta i karakteristike skupova podataka također predstavljaju izazov. Skupovi podataka koji opisuju SDP problem često imaju visoku dimenzionalnost, prisutnost šuma i nedostajuće vrijednosti, što otežava proces učenja modela. Međutim, u literaturi se ne posvećuje dovoljno pozornosti ovim karakteristikama i njihovom utjecaju na performanse modela. Istraživači često pretpostavljaju da će se modeli dobro generalizirati na različite skupove podataka, što nije uvijek slučaj. Nadalje, postupci predobrade podataka nisu uvijek detaljno opisani. Dok neki autori ističu važnost čišćenja podataka, odabira značajki i transformacije podataka, mnoge studije ne opisuju ove korake dovoljno detaljno. Ovo može dovesti do nedosljednosti u rezultatima i otežava usporedbu različitih istraživanja.

Zaključno, problem predviđanja softverskih pogrešaka je izuzetno složen i zahtijeva sustavan i standardiziran pristup. Potrebna su daljnja istraživanja koja će se fokusirati na temeljitu analizu utjecaja različitih algoritama, tehnika predobrade i karakteristika podataka na performanse modela. Samo kroz takav pristup može se postići bolje razumijevanje ovog problema i razviti učinkovitije metode za predviđanje pogrešaka u programskoj podršci. Uvođenje standardiziranih procedura za vrednovanje i dokumentaciju istraživanja također bi značajno doprinijelo ovom području, omogućujući istraživačima da pouzdanije uspoređuju rezultate i donose zaključke.

3. OSTVARENO PROGRAMSKO RJEŠENJE

U ovom poglavlju, cilj rada je objašnjen kroz implementaciju i opis uobičajenog tijeka učenja iz skupova podataka koji opisuju predviđanje pogrešaka u programskoj podršci. Ovaj tijek učenja organiziran je u nekoliko ključnih koraka koji zajedno čine cjeloviti okvir (engl. *framework*) za rješavanje problema predviđanje pogrešaka u programskoj podršci. Na temelju odabranih značajki podataka, algoritama strojnog učenja te odgovarajućih postupaka predobrade podataka, nastoji se razviti što učinkovitiji model za predviđanje.

Prvi korak u okviru odnosi se na pripremu podataka. Podaci prikupljeni iz različitih izvora prolaze kroz proces predobrade koji uključuje čišćenje podataka, uklanjanje nedostajućih vrijednosti, te transformaciju podataka u oblik pogodan za analizu. To je nužan korak kako bi se osigurala kvaliteta podataka koja direktno utječe na uspješnost kasnijih faza. Drugi korak uključuje odabir modela**.** U ovoj fazi treniraju se i podešavaju različiti algoritmi strojnog učenja s ciljem identifikacije onih koji pružaju najbolje performanse. Hiperparametri algoritama prilagođavaju se kako bi se optimizirala točnost predikcija. Prije nego što se poduzmu dodatni koraci, poput odabira značajki ili rukovanja neuravnoteženim klasama, najprije se evaluira koji algoritam najbolje odgovara podacima. Ako su performanse zadovoljavajuće, odabir značajki ili rukovanje s problemom neuravnoteženosti klasa neće biti potrebne. U slučaju nezadovoljavajućih performansi slijedi odabir značajki, koji se provodi kako bi se identificirale one varijable koje imaju najveći utjecaj na točnost modela. Značajke se odabiru koristeći tehnike kao što su analiza glavnih komponenti ili druge heurističke metode. Odabir prikladnih značajki ključan je za smanjenje kompleksnosti modela i poboljšanje njegove interpretabilnosti. Nakon toga, provodi se podešavanje algoritama strojnog učenja. Ovaj korak uključuje odabir najprikladnijih algoritama za predviđanje, kao i podešavanje njihovih hiperparametara s ciljem optimizacije performansi modela. Algoritmi se prilagođavaju specifičnostima podataka kako bi se postigla najbolja moguća točnost predikcije. U slučaju neuravnoteženih klasa, što je često u predviđanju pogrešaka u programskoj podršci, nužno je primijeniti specifične metode kako bi se izbjegla pristranost modela prema dominantnim klasama. U suprotnom, model bi mogao uspješno predviđati učestalije klase, ali bi bio neuspješan u otkrivanju rijetkih, ali kritičnih pogrešaka. Kako bi se to izbjeglo, koriste se različite tehnike, poput preuzorkovanja, poduzorkovanja, SMOTE. Ove metode omogućuju modelu točnije predviđanje rjeđe zastupljenih klasa, što je ključno za kvalitetno predviđanje pogrešaka u programskoj podršci. Treniranje modela je sljedeći korak, gdje se na temelju pripremljenih podataka i odabranih značajki model trenira kako bi naučio prepoznati obrasce koji vode do pojave pogrešaka u programskoj podršci. Treniranje modela zahtijeva odgovarajuće

vrednovanje performansi kako bi se osiguralo da model dobro generalizira na nove podatke. Na temelju provedenih koraka, okvir omogućava strukturiran i sveobuhvatan pristup problemu predviđanja pogrešaka u programskoj podršci . Svi koraci zajedno doprinose stvaranju pouzdanog i točnog modela koji može biti ključan alat u unapređenju kvalitete programske podrške.

Slika 3.1. Okvir za tijek učenja

3.1. Korišteni algoritmi za klasifikaciju

Za rješavanje problema predviđanja pogrešaka u programskoj podršci, često se primjenjuju različiti algoritmi klasifikacije. Ovi algoritmi omogućuju analizu podataka i identifikaciju potencijalnih pogrešaka, a njihova prilagodba i podešavanje ključnih parametara igraju značajnu ulogu u postizanju što točnijih rezultata.

Jedan od često korištenih algoritama je k-najbližih susjeda (engl. *k-Nearest Neighbors*, k-NN). Ovaj algoritam funkcionira tako da za svaki novi uzorak identificira *k* najbližih susjeda u

skupu podataka i na temelju većinske klase tih susjeda klasificira novi uzorak. Ključni parametar koji treba podesiti je broj susjeda, označen kao *k*. Optimalna vrijednost *k* može varirati ovisno o specifičnosti podataka. Manje vrijednosti *k* (primjerice, tri ili pet) mogu učiniti algoritam osjetljivim na šum, dok veće vrijednosti mogu dovesti do gubitka preciznosti jer uključuju i udaljenije susjede. Prilagođavanjem *k* vrijednosti moguće je postići ravnotežu između točnosti i otpornosti na šum. Za razliku od *k*-najbližih susjeda, naivni Bayes temelji se na probabilističkom pristupu, koristeći Bayesov teorem uz pretpostavku neovisnosti među značajkama. Ovaj algoritam je vrlo učinkovit u situacijama gdje su podaci višedimenzionalni, kao što je slučaj s tekstualnim podacima. Ključna odluka pri korištenju naivnog Bayesa je izbor odgovarajuće distribucije. Na primjer, Gaussov model često se koristi za kontinuirane podatke. Podešavanje ovih parametara osigurava da algoritam bude optimiziran za specifične karakteristike podataka. Stablo odluke pristupa klasifikaciji tako da gradi hijerarhijsku strukturu odluka, gdje svaka odluka dijeli podatke na manje podskupove na temelju određene značajke. Ovdje se ključna prilagodba odnosi na kontrolu složenosti modela. Parametri kao što su maksimalna dubina stabla i minimalni broj uzoraka po listu igraju presudnu ulogu. Ako je stablo previše duboko, model može postati previše specifičan za trening podatke, što vodi do prekomjernog prilagođavanja. S druge strane, preplitko stablo može biti previše općenito. Konačno, stroj potpornih vektora klasificira podatke tako da traži optimalnu hiperravninu koja najbolje razdvaja klase u prostoru značajki. Stroj potpornih vektora zahtijeva podešavanje nekoliko ključnih parametara, među kojima su *C* i funkcija jezgre. Parametar *C* kontrolira kompromis između maksimalizacije margine (udaljenosti između klasa) i broja pogrešno klasificiranih primjera. Veće vrijednosti *C* smanjuju dopuštenje za pogreške, što može poboljšati točnost, ali i povećati rizik od prekomjernog prilagođavanja. Funkcija jezgre, s druge strane, definira način na koji se podaci transformiraju u višedimenzionalni prostor, omogućujući algoritmu da se bolje nosi sa složenim i nelinearnim odnosima među značajkama. Prilagođavanje ovih hiperparametara omogućuje finu optimizaciju modela kako bi se postigli najbolji rezultati. Svi ovi algoritmi, kada se pravilno primijene i prilagode, nude moćne alate za predviđanje pogrešaka u programskoj podršci. Njihova učinkovitost ovisi ne samo o samom algoritmu, već i o tome koliko su dobro prilagođeni specifičnostima problema i podataka. Optimalno podešavanje hiperparametara ključ je uspješnog predviđanja i identifikacije pogrešaka, čime se može značajno poboljšati kvaliteta i pouzdanost programske podrške.

3.2. Predobrada skupova podataka odabirom značajki

Odabir značajki ključan je korak u predobradi podataka jer omogućuje smanjenje dimenzionalnosti skupa podataka, što olakšava treniranje modela i poboljšava njegovu interpretabilnost. Problem odabira značajki javlja se kada skup podataka sadrži velik broj značajki, od kojih mnoge mogu biti irelevantne ili redundantne. Korištenjem odgovarajućih tehnika, nastoji se identificirati podskup značajki koji najbolje doprinosi točnosti klasifikacijskog modela. Metode za odabir značajki obično se dijele na filtre, omotače, ugrađene metode i hibridne metode, pri čemu su filtri i omotači najpoznatije. Filtri rangiraju pojedine značajke na temelju mjera općih karakteristika podataka, poput dosljednosti, udaljenosti, količine informacija i korelacije, te odabiru unaprijed zadani broj najbolje rangiranih značajki. Jedan od ključnih parametara u ovakvom pristupu je broj odabranih značajki, a njegovo podešavanje nije trivijalan zadatak. Iako odabir najbolje rangiranih značajki može povećati učinkovitost klasifikacije, to ne garantira uvijek postizanje najviše razine uspješnosti. Naime, rangiranjem značajki može se dogoditi da se odbace one koje su same po sebi slabo relevantne, ali u kombinaciji s drugima značajno olakšavaju razlikovanje primjeraka različitih klasa. Među filtrima, Pearsonov koeficijent korelacije, zajednička informacija i ANOVA F-test često su korišteni zbog svoje jednostavnosti i učinkovitosti. Pearsonov koeficijent korelacije, primjerice, može se primijeniti za procjenu snage linearne veze između kontinuiranih značajki. Ako dvije ili više nezavisnih značajki imaju visok stupanj korelacije, tada se mogu smatrati redundantnim ili dupliciranim značajkama te se mogu odbaciti. Kada su nezavisne značajke visoko korelirane, promjena u jednoj značajki uzrokovala bi promjenu u drugoj, pa bi performanse algoritma strojnog učenja značajno fluktuirale. Jedan od poznatijih filtara, koji je prigodan za kategoričke značajke, jest zajednička informacija između značajke i oznake. Zajednička informacija procjenjuje količinu informacija koju jedna varijabla pruža o drugoj, rangirajući značajke prema količini informacije koju dijele s ciljanom varijablom, gdje veća zajednička informacija označava veću važnost značajke za klasifikaciju. ANOVA F-test koristi se za usporedbu srednjih vrijednosti između dvije ili više skupina i određivanje je li razlika između njih statistički značajna, pri čemu se značajke rangiraju prema vrijednosti F-testa, preferirajući one koje pokazuju značajne razlike između klasa.

S druge strane, među omotačima, slijedna pretraga unaprijed (engl. *sequential forward selection*, SFS) ističe se kao jednostavna i popularna metoda koja počinje s praznim skupom značajki, a zatim iterativno dodaje značajke koje najviše poboljšavaju performanse klasifikatora, sve dok se ne postigne zadovoljavajuća razina točnosti modela ili dok se ne dodaju sve značajke. Iako SFS može biti računalno intenzivan, omogućuje prilagodbu modela specifičnom skupu

podataka, često dovodeći do dobrih rezultata. Ove metode zajedno omogućuju učinkovit odabir značajki, što pridonosi boljoj izvedbi klasifikacijskih modela. U ostvarenom programskoj rješenju, primijenjeni su različiti filtri i omotači za odabir značajki. Filtri su korišteni za početnu procjenu važnosti značajki na temelju korelacije, zajedničke informacije i ANOVA F-testa, dok su omotači, poput slijedne pretrage unaprijed, omogućili iterativni odabir značajki koji poboljšava performanse modela. Ovaj pristup omogućio je poboljšanje performansi klasifikacijskih modela kroz preciznije odabiranje značajki koje doprinose točnosti modela.

3.3. Rukovanje s neuravnoteženim skupovima podataka

Problem neuravnoteženih skupova podataka često dovodi do pristranosti modela prema većinskoj klasi, što rezultira smanjenom točnošću u prepoznavanju manjinske klase. Kako bi se ovaj problem ublažio, koriste se različite tehnike, poput nasumičnog preuzorkovanja, nasumičnog poduzorkovanja i algoritma SMOTE, koje se često primjenjuju za postizanje bolje ravnoteže među klasama.

Nasumično preuzorkovanje podrazumijeva povećanje broja primjeraka iz manje zastupljene klase. To se postiže tako da se postojeći primjerci manjinske klase nasumično dupliciraju, čime se umjetno povećava njihova zastupljenost u skupu podataka. Na primjer, ako je u skupu podataka prisutno samo 100 primjeraka manjinske klase, a 1000 primjeraka većinske klase, nasumičnim preuzorkovanjem može se broj primjeraka manjinske klase povećati na 1000, dupliciranjem postojećih 100 primjeraka sve dok se ne postigne željeni omjer. Iako ovaj pristup može dovesti do uravnoteženosti između klasa, postoji rizik od prekomjernog prilagođavanja jer model može naučiti specifičnosti dupliciranih primjeraka umjesto generalnih obrazaca koji vrijede za cijelu klasu. S druge strane, nasumično poduzorkovanje smanjuje broj primjeraka iz dominantne klase, uklanjanjem dijela podataka kako bi se postigla ravnoteža. To se postiže nasumičnim uklanjanjem primjeraka iz većinske klase, čime se smanjuje njezina dominacija u skupu podataka. Na primjer, ako u skupu podataka postoji 1000 primjeraka većinske klase i 100 primjeraka manjinske klase, nasumičnim poduzorkovanjem može se broj primjeraka većinske klase smanjiti na 100. Iako ovaj pristup može smanjiti pristranost modela prema većinskoj klasi, postoji rizik od gubitka važnih informacija jer uklanjanje primjeraka može značiti i uklanjanje korisnih podataka koji su bitni za razlikovanje klasa. Ovaj rizik je posebno visok ako se uklone primjerci koji predstavljaju rubne ili atipične slučajeve unutar većinske klase. Za razliku od ovih jednostavnih pristupa, SMOTE nudi sofisticiraniji pristup rješavanju problema neuravnoteženih skupova podataka. Umjesto jednostavnog dupliciranja postojećih primjeraka, SMOTE generira nove

sintetičke primjerke za manjinsku klasu. Algoritam funkcionira tako da za svaki primjerak iz manjinske klase odabere jednog ili više najbližih susjeda iz iste klase. Zatim se između tog primjerka i njegovih susjeda stvara nova točka koja se pozicionira na nekoj udaljenosti između tih točaka, stvarajući novi sintetički primjerak. Ova metoda ne samo da povećava broj primjeraka manjinske klase, nego također proširuje varijabilnost unutar te klase, što može pomoći modelu da nauči generalnije obrasce i smanji rizik od prekomjernog prilagođavanja. Prednost algoritma SMOTE je u tome što smanjuje pristranost prema većinskoj klasi, istovremeno zadržavajući složenost i informativnost skupa podataka. U ostvarenom programskom rješenju, korištene su različite tehnike za rješavanje problema neuravnoteženih skupova podataka. Nasumično preuzorkovanje i nasumično poduzorkovanje primijenjeni su za početnu prilagodbu ravnoteže između klasa, pri čemu je nasumično preuzorkovanje povećalo broj primjeraka manjinske klase dupliciranjem postojećih podataka, dok je nasumično poduzorkovanje smanjilo broj primjeraka većinske klase uklanjanjem dijela podataka. Za sofisticiraniji pristup, korišten je SMOTE, koji je generirao nove sintetičke primjerke za manjinsku klasu stvaranjem točaka između postojećih primjeraka i njihovih najbližih susjeda. Ove metode su omogućile poboljšanje ravnoteže između klasa, što je rezultiralo boljoj učinkovitosti modela u prepoznavanju manjinske klase.

4. EKSPERIMENTALNA ANALIZA I REZULTATI

U ovom poglavlju opisana je provedena eksperimentalna analiza koja za cilj ima vrednovanje učinkovitosti različitih algoritama za klasifikaciju u kontekstu predviđanja pogrešaka u programskoj podršci. U prvom dijelu poglavlja su prikazane postavke i metodologija korištena u eksperimentu. U drugom dijelu poglavlja analizirane su performanse klasifikacijskih algoritama prije provedbe predobrade podataka, s naglaskom na identifikaciju algoritma koji pokazuje najbolje rezultate. Treći dio fokusiran je na analizu učinka odabira značajki na performanse klasifikatora, uz razmatranje različitih brojeva odabranih značajki. U posljednjem dijelu poglavlja analiziran je učinak različitih metoda uzorkovanja na performanse klasifikacijskih algoritama, uključujući nasumično preuzorkovanje, nasumično poduzorkovanje i algoritam SMOTE. Ovim poglavljem namjerava se pružiti sveobuhvatan uvid u učinkovitost različitih pristupa klasifikaciji u zadatku predviđanja pogrešaka u programskoj podršci, uzimajući u obzir razne tehnike predobrade podataka.

4.1. Postavke i metodologija eksperimentalne analize

Skupovi podataka korišteni u eksperimentalnoj studiji potječu iz PROMISE repozitorija i predstavljaju javno dostupne NASA-ine skupove podataka za predviđanje pogrešaka u programskoj podršci. Svi skupovi podataka predstavljaju probleme binarne klasifikacije, dok su značajke izvedene iz metrika složenosti programske podrške, poput mjera broja linija koda, McCabe-ovih metrika, Halstead-ovih metrika i drugih. Odabrano je ukupno 12 skupova podataka, čije su karakteristike prikazane u tablici 4.1.

Tablica 4.1. Karakteristike skupova podataka

Skup podataka	**Broj primjera**	**Broj značajki**	**IR**
Cm1	344	42	7.19
Jm1	9591	21	4.45
Kc1	2095	21	5.45
Kc3	200	39	4.56
Mc1	8737	38	127.49
Mc2	125	39	1.84
Mw1	263	37	8.74

Pc1	735	37	11.05
Pc2	1493	36	92.31
Pc3	1099	37	6.97
Pc4	1379	37	6.75
Pc5	16962	38	32.79

Kako bi se potvrdila prisutnost spomenutih poteškoća, u tablici 4.2 prikazane su apsolutne vrijednosti mjera složenosti. Detaljne formule i objašnjenja korištenih mjera mogu se pronaći u prilogu P.4.1. Iako se ne preporučuje oslanjati samo na apsolutne vrijednosti mjera složenosti podataka, jasno je da se problemi preklapanja klasa i malih disjunkcija manifestiraju unutar SDP skupova podataka, budući da se u gotovo nijednom aspektu ne postiže minimalni stupanj složenosti. Međutim, stvarni stupnjevi ovih intrinzičnih karakteristika teško se odrede, jer su vrijednosti tih mjera namijenjene usporedbi s istim mjerama na drugim problemima ili za izvođenje korelacija s uspjehom klasifikacije.

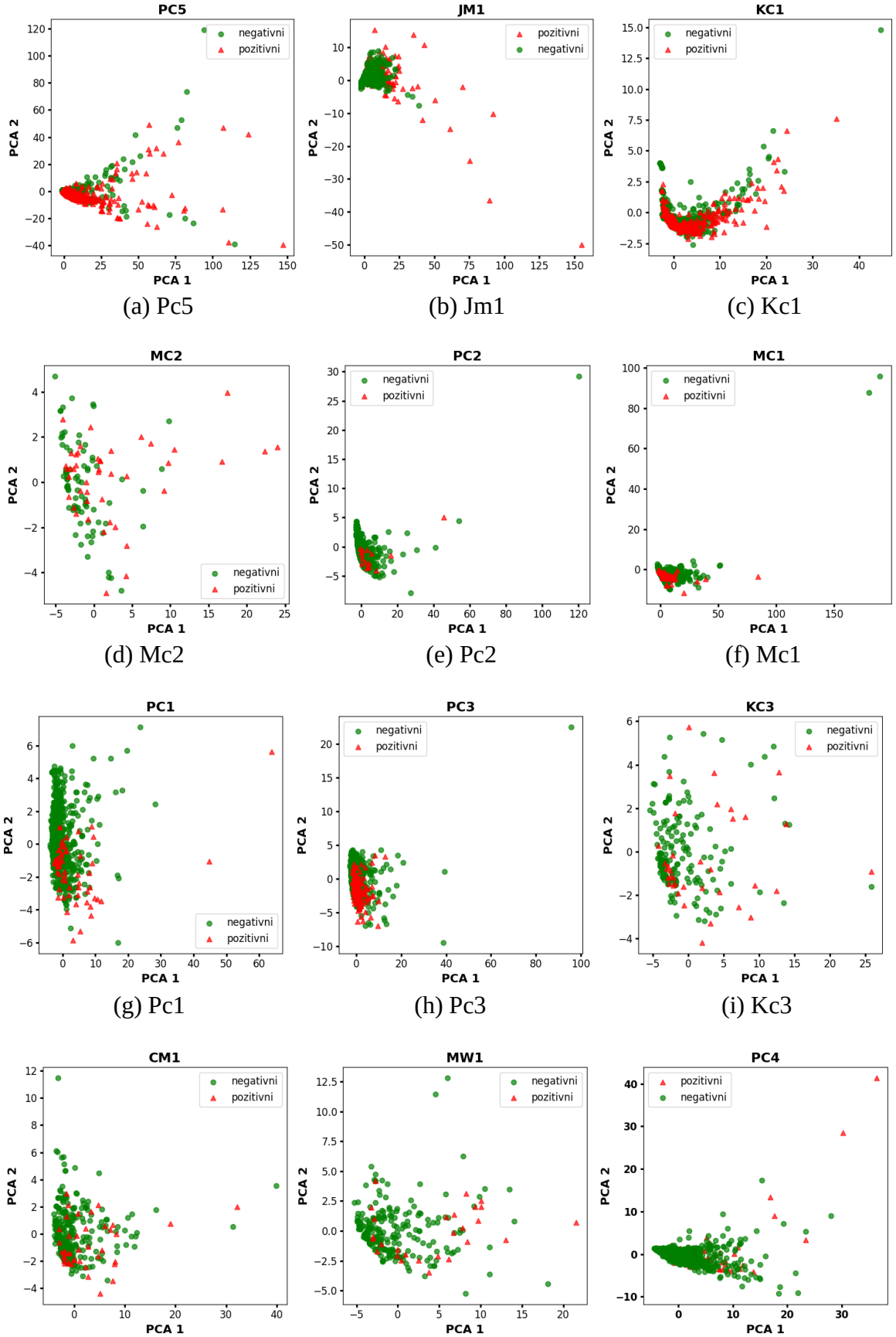
Tablica 4.2. Mjere složenosti korištenih skupova podataka

Skup podataka	**F1**	**F2**	**F3**	**Broj disjunkta**	**Prosječna veličina disjukta**	**N2**	**N3**	**LSCAvg**
Cm1	0.26	2.26e-06	0.27	41	8.39	0.57	1.78	0.30
Jm1	0.13	1.11e-09	0.00	1428	6.72	0.71	0.3	0.00
Kc1	0.53	1.03e-03	0.07	262	8.00	1.25	0.19	0.02
Kc3	0.24	7.59e-05	0.11	32	6.25	1.32	0.3	0.03
Mc1	1.54	1.49e-14	0.5	63	138.68	0.06	0.01	0.3
Mc2	0.35	2.66e-09	0.15	24	5.21	0.45	0.38	0.02
Mw1	0.60	3.09e-06	0.25	28	9.4	0.88	0.19	0.05
Pc1	0.58	4.02e-10	0.36	55	13.36	1.18	0.13	0.03
Pc2	1.40	5.34e-15	0.78	24	62.21	0.33	0.21	0.33
Pc3	0.46	3.66e-20	0.23	103	10.67	0.52	0.19	0.02
Pc4	0.51	1.04e-10	0.12	98	14.07	0.71	0.19	0.01
Pc5	1.66	9.62e-08	0.09	285	59.52	0.39	0.03	0.13

Podešavanje hiperparametara provedeno je za sve klasifikatore tijekom koraka unakrsne validacije kako bi se bolje razumjelo kako klasifikatori funkcioniraju na neuravnoteženim problemima i kako bi se smanjio učinak odabira neodgovarajućih parametara. Budući da algoritam NB ne sadrži hiperparametre, faza podešavanja je izostavljena.

Uz izbor skupova podataka, literatura sugerira da se vrednovanje performansi klasifikacije često provodi putem unakrsne validacije. Stoga su odabir i vrednovanje modela provedeni korištenjem ponovljene stratificirane unakrsne validacije s 5 presjeka i 3 ponavljanja, slično kao u [39]. Izvorni skupovi podataka podijeljeni su u skupove za treniranje i testiranje u omjeru 0.8:0.2 prije vrednovanja. Samo je skup za treniranje korišten za unakrsnu validaciju i podešavanje hiperparametara, a najbolji modeli su vrednovani na testnom skupu. Za svaku kombinaciju klasifikatora i skupa podataka provedeno je 10 neovisnih izvođenja eksperimenta kako bi se osiguralo pravedno vrednovanje i smanjio utjecaj stohastičkog ponašanja evaluiranih klasifikatora.

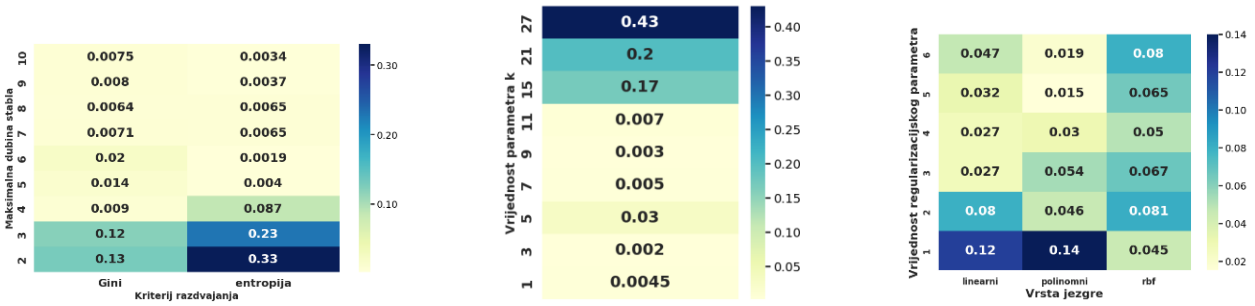
S obzirom na višedimenzionalnu prirodu SDP skupova podataka u pogledu broja značajki, njihova struktura teško se uočava. Kako bi se dobio približan uvid, primijenjena je analiza glavnih komponenti (PCA) za komprimiranje skupova podataka u podprostor s manjim brojem dimenzija, uz očuvanje većine relevantnih informacija, odnosno što vjernije zadržavanje izvorne strukture. Napominje se da su u eksperimentalnoj analizi korišteni izvorni, nereducirani skupovi podataka te da je analiza glavnih komponenti korištena isključivo za eksplorativnu analizu podataka. Analiza glavnih komponenti je nenadzirana tehnika linearne transformacije koja se često koristi za odabir značajki i smanjenje dimenzionalnosti. Međutim, druga popularna primjena analize glavnih komponenata uključuje eksplorativnu analizu podataka. Osnovni cilj analize glavnih komponenata je pronalaženje smjerova maksimalne varijance u visokodimenzionalnim podacima i njihovo projiciranje na novi podprostor s jednakim ili manjim brojem dimenzija od izvornog. Ortogonalne osi (glavne komponente) novog podprostora mogu se interpretirati kao smjerovi maksimalne varijance uz uvjet da su osi novih značajki međusobno ortogonalne. Ovisno o broju dimenzija na koji se izvorni skupovi podataka smanjuju, odabiru se samo glavne komponente koje sadrže većinu informacija, odnosno varijance. Na slici 4.1 prikazani su korišteni skupovi podataka za zadatak SDP, reducirani na dvodimenzionalni prostor značajki. Na temelju prikazanih slika uočena je neuravnoteženost podataka, pri čemu je klasa negativnih uzoraka znatno brojnija od pozitivnih. Također, u većini grafova nije uočena jasna separacija između pozitivnih i negativnih uzoraka, što dodatno otežava klasifikaciju i negativno utječe na točnost modela.



Slika 4.1. Skupovi podataka reducirani na dvije dimenzije pomoću tehnike PCA

4.2. Usporedba klasifikatora

U ovom poglavlju prikazani su rezultati usporedbe različitih klasifikatora u zadatku predviđanja pogrešaka u programskoj podršci. Analizirani su klasifikatori KNN, NB, DT i SVM, uz detaljno podešavanje hiperparametara za svaki od njih. Toplinske karte na slici 4.2. prikazuju učestalost najboljih postavki parametara za svaku kombinaciju skupa podataka i klasifikatora u svim izvođenjima. Svaka vrijednost na toplinskoj karti prikazuje udio instanci u kojima je određena kombinacija parametara bila najbolje rangirana, a prikazana je u odnosu na ukupan broj eksperimentalnih zadataka. Rezultati pokazuju da su manje vrijednosti za maksimalnu dubinu stabla odluke preferirane bez obzira na kriterij podjele, pri čemu entropija marginalno nadmašuje gini nečistoću. S druge strane, veće vrijednosti za broj susjeda u k-najbližih susjeda bolje odgovaraju za SDP probleme. Na kraju, vrijednost regularizacijskog parametra C = 1 u SVM-u pokazala se kao najbolje izvedena postavka neovisno o vrsti jezgre, pri čemu je jezgra s radijalnom baznom funkcijom pokazao lošije rezultate u odnosu na linearne i polinomske jezgre.



(a) Najbolje postavke za DT-a (b) Najbolje postavke za k-NN (c) Najbolje postavke za SVM

Slika 4.2. Relativna učestalost najboljih hiperparametara klasifikatora na stvarnim podacima

Budući da je AUC metrika služila kao osnova za podešavanje parametara, tablica 4.3. prikazuje prosječne AUC rezultate dobivene kroz sve pokuse. Nekoliko zasebnih mjera izvedenih iz prosječnih AUC rezultata za sve skupove podataka prikazano je na dnu tablice kako bi se olakšala usporedba evaluiranih. Mjera označena s d_2 predstavlja Euklidsku udaljenost od hipotetskog savršenog klasifikatora, što je interpretacija modela koji postiže AUC rezultat od 1 na svim skupovima podataka. Najbolja vrijednosti prikazana je podebljana za d_2, pri čemu niža vrijednost implicira bolji rang. Najprikladniji algoritam se pokazao NB za predviđanje pogrešaka programske podrške. Međutim, postignute apsolutne vrijednosti AUC rezultata su relativno niske, što ukazuje na to da se svi klasifikatori muče s ispravnim klasificiranjem značajnog dijela instanci iz korištenih skupova podataka.

Tablica 4.3. Izvedbe klasifikatora prema prosječnim AUC mjerama

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Cm1	0.54 ± 0.11	**0.65** ± 0.08	0.56 ± 0.08	0.47 ± 0.13
Jm1	0.63 ± 0.01	**0.64** ± 0.02	0.59 ± 0.02	0.63 ± 0.05
Kc1	0.71 ± 0.04	**0.78** ± 0.02	0.61 ± 0.04	0.57 ± 0.05
Kc3	0.60 ± 0.10	**0.67** ± 0.10	0.64 ± 0.10	0.46 ± 0.13
Mc1	0.72 ± 0.06	**0.91** ± 0.02	0.74 ± 0.07	0.56 ± 0.26
Mc2	0.60 ± 0.11	**0.67** ± 0.13	0.64 ± 0.09	0.60 ± 0.09
Mw1	0.61 ± 0.12	**0.76** ± 0.12	0.62 ± 0.09	0.43 ± 0.15
Pc1	0.54 ± 0.07	**0.71** ± 0.08	0.63 ± 0.07	0.53 ± 0.10
Pc2	0.52 ± 0.07	**0.84** ± 0.12	0.56 ± 0.11	0.51 ± 0.19
Pc3	0.65 ± 0.06	**0.72** ± 0.05	0.63 ± 0.04	0.41 ± 0.18
Pc4	0.59 ± 0.03	**0.78** ± 0.04	0.71 ± 0.04	0.51 ± 0.06
Pc5	0.89 ± 0.01	**0.95** ± 0.00	0.74 ± 0.02	0.55 ± 0.08
Prosjek	0.63 ± 0.01	**0.76** ± 0.02	0.60 ± 0.03	0.52 ± 0.00
Medijan	0.61 ± 0.01	**0.74** ± 0.00	0.63 ± 0.01	0.52 ± 0.01
d2	1.22	**0.91**	1.21	1.52

Provedena je detaljna eksperimentalna analiza kako bi se utvrdilo kako različiti klasifikatori djeluju u zadatku predviđanja pogrešaka u programskoj podršci. Analizirani su i uspoređeni klasifikatori KNN, NB, DT i SVM. Kako bi se osigurala pravedna usporedba, za sve klasifikatore provedeno je podešavanje hiperparametara. Eksperimentalni rezultati pokazuju kako se ovi klasifikatori nose s izazovima koje predstavljaju SDP zadaci. Korišteni su različite mjere kao što su F1 mjera, preciznost i odziv, kako bi se evaluirale performanse klasifikatora.

Tablica 4.4. Izvedbe klasifikatora prema prosječnim F1 mjerama

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Cm1	0.80 ± 0.04	**0.81** ± 0.05	**0.81** ± 0.04	**0.81** ± 0.04
Jm1	0.78 ± 0.03	**0.78** ± 0.01	0.77 ± 0.01	0.75 ± 0.01
Kc1	**0.83** ± 0.01	0.82 ± 0.02	0.81 ± 0.01	0.80 ± 0.01
Kc3	0.74 ± 0.04	**0.77** ± 0.05	0.73 ± 0.07	0.75 ± 0.07
Mc1	**0.99** ± 0.00	0.96 ± 0.01	**0.99** ± 0.00	0.98 ± 0.00
Mc2	**0.66** ± 0.03	**0.66** ± 0.04	0.65 ± 0.09	0.62 ± 0.09
Mw1	**0.86** ± 0.02	0.79 ± 0.05	0.80 ± 0.02	0.82 ± 0.02
Pc1	**0.89** ± 0.01	0.88 ± 0.01	0.88 ± 0.02	0.88 ± 0.01
Pc2	**0.98** ± 0.02	0.97 ± 0.01	**0.98** ± 0.01	0.97 ± 0.01
Pc3	**0.84** ± 0.01	0.81 ± 0.09	0.83 ± 0.02	0.81 ± 0.02
Pc4	**0.88** ± 0.01	0.84 ± 0.02	**0.88** ± 0.02	0.87 ± 0.01
Pc5	**0.97** ± 0.01	0.96 ± 0.01	**0.97** ± 0.01	0.96 ± 0.00
Prosjek	0.83 ± 0.00	**0.84** ± 0.01	0.83 ± 0.00	0.83 ± 0.00
Medijan	**0.84** ± 0.00	0.83 ± 0.00	0.83 ± 0.01	0.82 ± 0.00
d2	**0.64**	0.66	0.69	0.72

Tablica 4.5. Izvedbe klasifikatora prema prosječnoj preciznosti

Skup podatak	**KNN**	**NB**	**DT**	**SVM**
Cm1	0.78 ± 0.04	**0.82** ± 0.04	0.81 ± 0.03	0.76 ± 0.04
Jm1	0.77 ± 0.01	**0.78** ± 0.01	0.77 ± 0.01	0.80 ± 0.02
Kc1	0.82 ± 0.01	0.82 ± 0.01	0.81 ± 0.02	**0.83** ± 0.03
Kc3	0.72 ± 0.04	**0.78** ± 0.06	0.72 ± 0.08	0.73 ± 0.11
Mc1	**0.99** ± 0.00	0.98 ± 0.01	**0.99** ± 0.00	0.98 ± 0.00
Mc2	0.68 ± 0.03	**0.72** ± 0.06	0.66 ± 0.09	0.69 ± 0.07
Mw1	**0.86** ± 0.02	**0.86** ± 0.02	0.82 ± 0.01	0.78 ± 0.03
Pc1	**0.89** ± 0.02	0.88 ± 0.01	0.88 ± 0.02	0.85 ± 0.02
Pc2	**0.98** ± 0.01	**0.98** ± 0.01	0.97 ± 0.01	0.97 ± 0.01
Pc3	0.83 ± 0.01	**0.85** ± 0.02	0.83 ± 0.02	0.76 ± 0.02
Pc4	0.88 ± 0.01	0.84 ± 0.02	0.88 ± 0.02	**0.89** ± 0.01
Pc5	**0.97** ± 0.00	**0.97** ± 0.00	**0.97** ± 0.00	**0.97** ± 0.00
Prosjek	0.80 ± 0.00	**0.81** ± 0.01	**0.81** ± 0.00	**0.81** ± 0.00
Medijan	**0.83** ± 0.01	0.82 ± 0.00	0.79 ± 0.00	0.79 ± 0.01
d2	0.68	**0.62**	0.69	0.69

Tablica 4.6. Izvedbe klasifikatora prema prosječnom odzivu

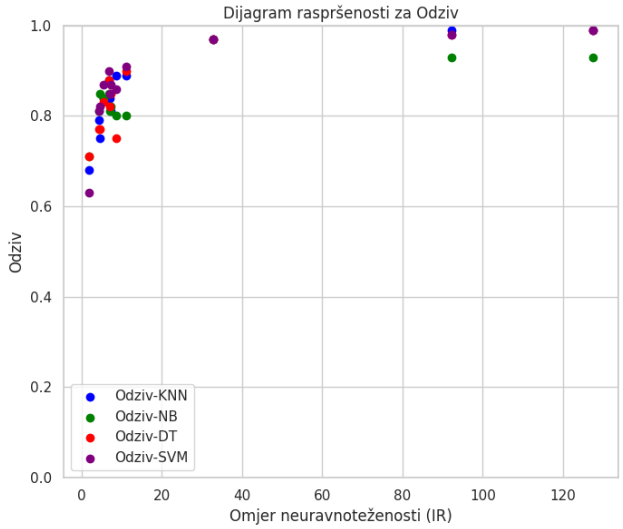
Skup podatak	**KNN**	**NB**	**DT**	**SVM**
Cm1	0.83 ± 0.02	0.81 ± 0.05	0.82 ± 0.05	**0.87** ± 0.02
Jm1	0.80 ± 0.01	0.81 ± 0.01	0.76 ± 0.01	**0.82** ± 0.01
Kc1	**0.85** ± 0.01	0.83 ± 0.01	0.81 ± 0.01	**0.85** ± 0.01
Kc3	0.79 ± 0.04	0.78 ± 0.04	0.74 ± 0.05	**0.81** ± 0.05
Mc1	**0.99** ± 0.00	0.94 ± 0.00	**0.99** ± 0.00	**0.99** ± 0.00
Mc2	0.68 ± 0.03	**0.69** ± 0.05	0.66 ± 0.08	0.67 ± 0.06
Mw1	**0.88** ± 0.01	0.76 ± 0.07	0.78 ± 0.03	0.87 ± 0.01
Pc1	**0.91** ± 0.01	0.88 ± 0.01	0.88 ± 0.01	**0.91** ± 0.00
Pc2	**0.99** ± 0.00	0.95 ± 0.01	0.98 ± 0.00	**0.99** ± 0.00
Pc3	0.85 ± 0.01	0.71 ± 0.07	0.83 ± 0.01	**0.87** ± 0.03
Pc4	0.89 ± 0.01	0.84 ± 0.03	0.88 ± 0.01	**0.90** ± 0.01
Pc5	**0.97** ± 0.00	**0.97** ± 0.00	**0.97** ± 0.00	**0.97** ± 0.00
Prosjek	0.77 ± 0.01	0.75 ± 0.02	0.75 ± 0.00	**0.85** ± 0.01
Medijan	**0.87** ± 0.00	0.82 ± 0.00	0.81 ± 0.01	**0.87** ± 0.00
d2	0.68	**0.62**	0.69	0.69

Kombinirajući sve relevantne informacije, rezultati F1 mjere ukazuju na to da su svi klasifikatori postigli slične ukupne rezultate, pri čemu se NB i SVM izdvajaju kao klasifikatori s najvišim prosječnim vrijednostima. Sličan obrazac uočen je i kod preciznosti, gdje je NB ostvario najviši prosječni rezultat, dok je stroj potpornih vektora zadržao vodeću poziciju u prepoznavanju pozitivnih klasa (odziv). Na kraju, analiza Euklidske udaljenosti od savršenog klasifikatora pokazala je da su NB i SVM najbliži idealnim rezultatima, što dodatno naglašava njihovu efikasnost u ovom kontekstu. Ovi rezultati sugeriraju da, iako su svi klasifikatori pokazali određeni

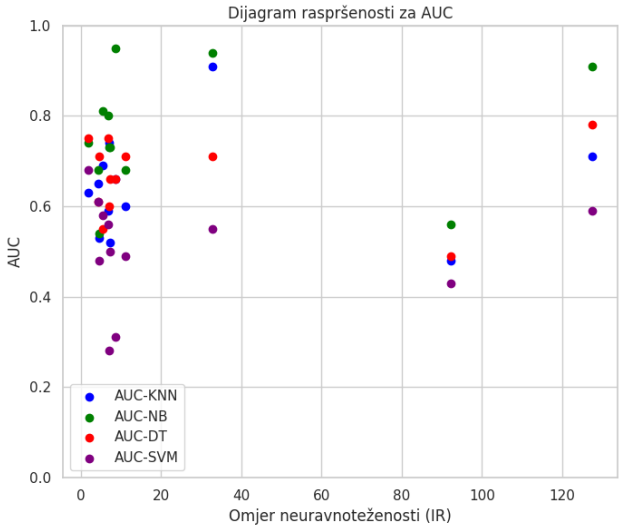
stupanj uspješnosti, NB i SVM pokazuju veću robusnost i prilagodljivost izazovima koje predstavljaju SDP zadaci. Na temelju ovih rezultata, može se zaključiti da su navedeni algoritmi posebno prikladni za rješavanje problema predviđanje pogrešaka u programskoj podršci, dok su drugi klasifikatori, kao što su KNN i DT, također pokazali solidne performanse, ali s određenim ograničenjima u specifičnim aspektima zadatka.

Prikazani rezultati u tablicama 4.3. do 4.6. ukazuju na razlike u izvedbi različitih klasifikatora prema evaluacijskim metrikama. Očekivano, s obzirom na neuravnoteženu prirodu problema SDP-a, primijećeno je da su točnost i specifičnost često bile veće od F1 mjere i osjetljivosti (TPR). Ovaj rezultat proizlazi iz načina na koji klasifikatori obrađuju neuravnotežene podatke. Kod neuravnoteženih skupova, točnost može biti visoka čak i kada model loše klasificira manje prisutne klase, jer većina predviđanja dolazi iz dominantne klase. F1 mjera, koja balansira između preciznosti i osjetljivosti, pokazala je niže vrijednosti, osobito kod skupova gdje model nije uspješno prepoznao manjinske instance. Primjerice, F1 mjera za skupove poput *Mc2* i *Pc2* bila je relativno niža u usporedbi s njihovim stopama preciznosti i odziva, što ukazuje na mogućnost poboljšanja u ravnoteži između pogrešno pozitivnih i pogrešno negativnih klasifikacija. Analiza AUC rezultata pokazala je da, unatoč tome što su NB i SVM postigli najbolje prosječne vrijednosti AUC-a, apsolutne vrijednosti ostaju relativno niske. Ovaj podatak potvrđuje da nijedan klasifikator nije idealan za ove zadatke zbog specifičnih izazova koje SDP problemi nose, uključujući neuravnoteženost klasa i varijabilnost karakteristika različitih skupova podataka. Razlike u preciznosti pokazale su da je NB pokazao veću konzistentnost, dok je SVM pokazao oscilacije ovisno o skupu podataka. Te oscilacije mogle bi biti povezane s osjetljivošću SVM-a na parametre poput postavljanja granice između klasa. S druge strane, rezultati odziva pokazali su da su SVM i NB postigli dobre rezultate u prepoznavanju pozitivnih klasa. SVM se pokazao nešto boljim u prepoznavanju pozitivnih instanci u neuravnoteženim skupovima, što bi moglo biti posljedica njegove sposobnosti uvođenja maksimalne margine između klasa.

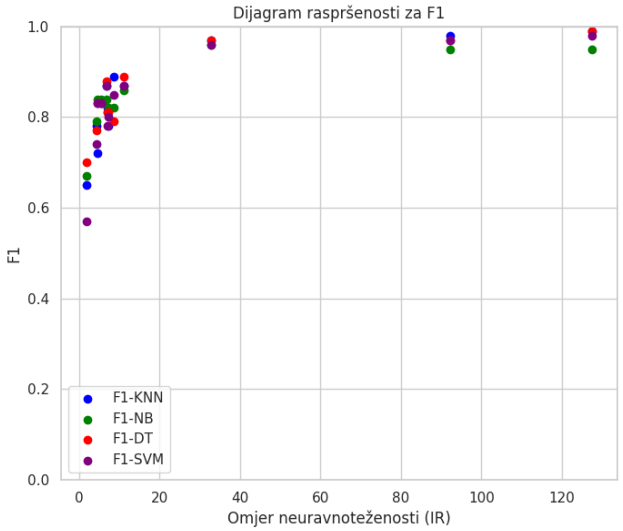
Zaključno, iako neuravnoteženost podataka utječe na sve metrike, uočeno je da su NB i SVM klasifikatori koji su se najbolje prilagodili tim izazovima. To je potvrđeno boljim rezultatima u većini prikazanih metrika, dok su KNN i DT pokazali slabije rezultate u određenim aspektima, osobito kada je bilo potrebno uravnotežiti preciznost i osjetljivost. Slika 4.3 prikazuje dijagrame raspršenosti prema kojima s porastom omjera neuravnoteženosti odziv, preciznost i F1 mjera rastu, dok AUC ostaje stabilan ili se smanjuje. Ovo se odražava kao poboljšanje u sposobnosti modela da prepoznaje rijetke pozitivne uzorke, dok AUC, koji mjeri rangiranje, nije značajno pogođen promjenama u raspodjeli klasa.



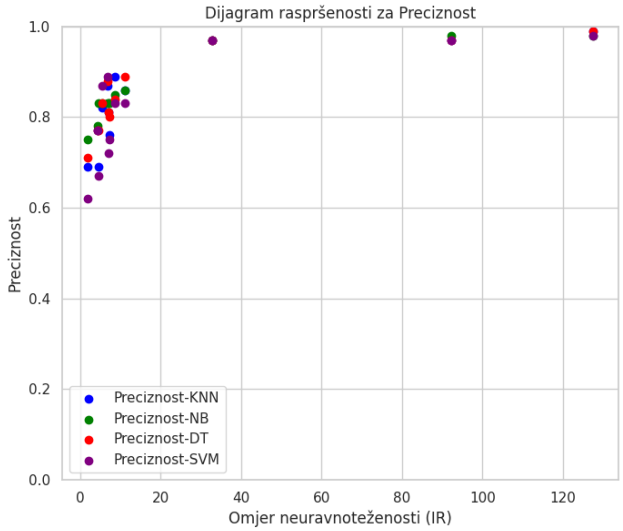
(a) Dijagrama raspršenosti za Odziv



(b) Dijagrama raspršenosti za AUC



(c) Dijagrama raspršenosti za F1 mjeru



(d) Dijagrama raspršenosti za Preciznost

Slika 4.3. Dijagrama raspršenosti za Odziv, AUC, F1 mjeru i Preciznost

4.3. Analiza učinka odabira značajki

Odabir značajki ključan je za poboljšanje performansi modela u strojnom učenju. Analizom različitih metoda odabira značajki utvrđeno je kako se performanse modela razlikuju ovisno o korištenom algoritmu i broju odabranih značajki. U tablicama 4.7, 4.8, 4.9 i 4.10 prikazane su kombinacije broja značajki i algoritma koja daje najvišu točnost za različite metode odabira značajki. Za svaki skup podataka koriste se četiri različite metode odabira značajki: Pearsonov

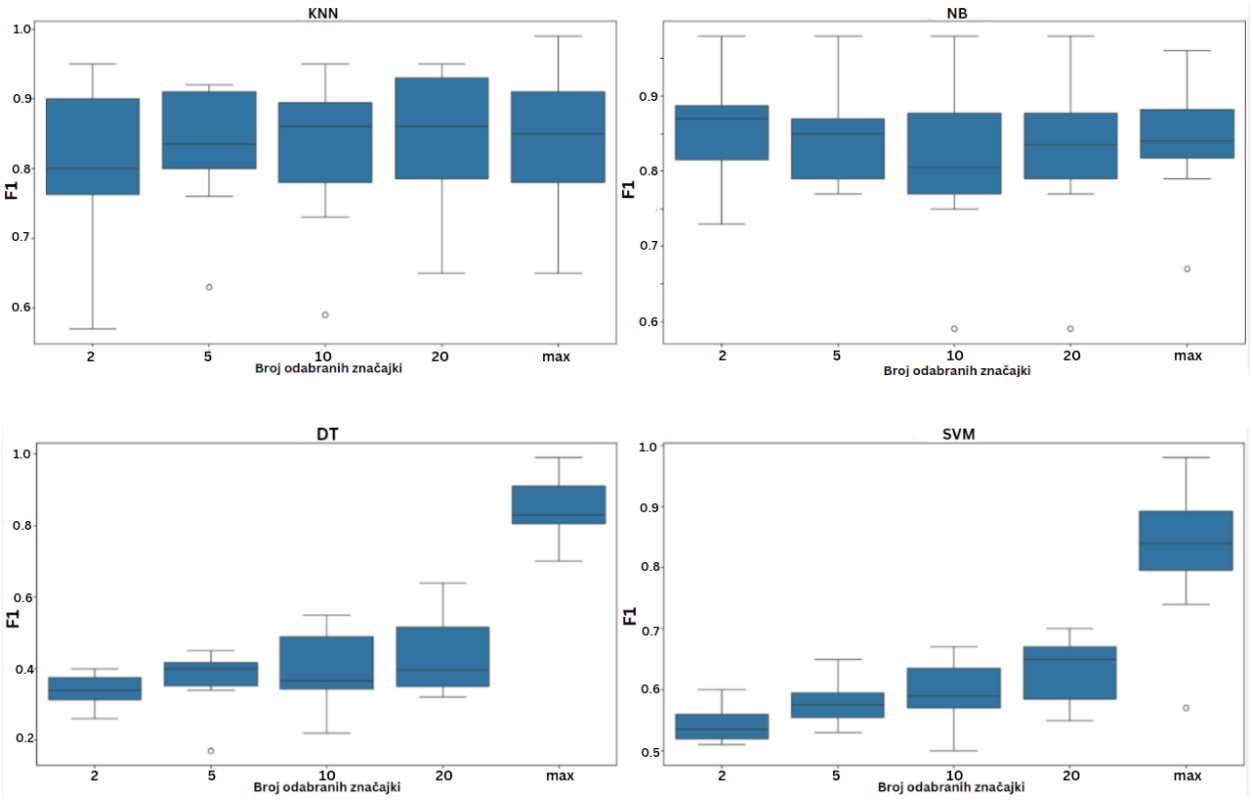
koeficijent korelacije, zajednička informacija ANOVA F-test i slijedna pretraga unaprijed.

U tablici 4.7 KNN pokazuje dobre performanse s različitim brojem značajki, ali se najbolje rezultate postiže s pet značajki za većinu skupova podataka. Veći broj značajki često ne donosi značajna poboljšanja. Nasuprot tome, NB često pokazuje vrlo dobre rezultate s manjim brojem značajki, posebno sa dva značajke. U nekoliko slučajeva, povećanje broja značajki do 20 poboljšava točnost. Performanse stabla odluke variraju ovisno o broju značajki. Optimalan broj značajki se razlikuje među skupovima podataka, a često su to pet ili 10 značajki, dok previše značajki može smanjiti točnost. SVM pokazuje dobre performanse s većim brojem značajki. U većini slučajeva, 20 značajki je optimalno za postizanje visoke točnosti Na dijagramima 4.4 prikazane su F1 mjere algoritama u ovisnosti o broju odabranih značajki, pri čemu je na osi *x* prikazan broj odabranih značajki, a na osi *y* F1 mjera, dok posljednji stupac predstavlja slučaj kada su odabrane sve značajke. Za algoritam DT primijećeno je povećanje F1 mjere s porastom broja značajki, pri čemu je najbolji rezultat postignut kada su odabrane sve značajke. Kod algoritma SVM, zabilježeno je povećanje F1 mjere s većim brojem značajki, a najviša F1 mjera postignuta je kada su sve značajke korištene. Za KNN algoritam, F1 mjera je ostala stabilna bez većih promjena, neovisno o broju odabranih značajki. Kod NB algoritma, F1 mjera je varirala ovisno o broju značajki, ali je najveći porast zabilježen kada su sve značajke korištene.

Tablica 4.7. F1 mjera klasifikatora uz primjenu Pearsonove korelacije

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Pc4	0.85 ± 0.01 (+0.03)	**0.87** ± 0.01 (+0.03)	0.45 ± 0.05 (-0.43)	0.63 ± 0.03 (-0.24)
Cm1	0.82 ± 0.01 (+0.02)	**0.85** ± 0.03 (+0.04)	0.32 ± 0.06 (-0.49)	0.51 ± 0.04 (-0.30)
Mw1	**0.86** ± 0.03 (+0.00)	0.83 ± 0.01 (+0.04)	0.30 ± 0.08 (-0.50)	0.52 ± 0.02 (-0.30)
Kc3	0.73 ± 0.02 (+0.01)	**0.78** ± 0.04 (+0.01)	0.32 ± 0.08 (-0.41)	0.56 ± 0.03 (-0.19)
Mc2	0.60 ± 0.02 (-0.06)	**0.63** ± 0.00 (-0.03)	0.37 ± 0.13 (-0.28)	0.59 ± 0.03 (-0.03)
Pc2	**0.98** ± 0.00 (+0.00)	0.96 ± 0.00 (-0.01)	0.20 ± 0.07 (-0.78)	0.65 ± 0.01 (-0.32)
Pc3	**0.80** ± 0.01 (+0.04)	**0.80** ± 0.02 (-0.01)	0.30 ± 0.10 (-0.53)	0.63 ± 0.01 (-0.17)
Pc1	**0.88** ± 0.01 (+0.01)	0.87 ± 0.00 (-0.01)	0.39 ± 0.03 (-0.42)	0.61 ± 0.03 (+0.05)
Pc5	**0.97** ± 0.00 (+0.00)	0.96 ± 0.00 (+0.00)	0.44 ± 0.05 (-0.53)	0.71 ± 0.02 (-0.25)
Mc1	**0.99** ± 0.00 (+0.00)	0.97 ± 0.01 (+0.01)	0.57 ± 0.04 (-0.42)	0.71 ± 0.01 (-0.27)
Jm1	0.77 ± 0.00 (-0.01)	**0.78** ± 0.00 (+0.05)	0.34 ± 0.04 (-0.33)	0.55 ± 0.02 (-0.20)
Kc1	0.84 ± 0.01 (+0.01)	**0.85** ± 0.00 (+0.03)	0.32 ± 0.10 (-0.49)	0.71 ± 0.02 (-0.09)

Prosjek	0.76 ± 0.00 (-0.07)	**0.86** ± 0.00 (+0.02)	0.34 ± 0.02 (-0.49)	0.63 ± 0.01 (-0.20)
Medijan	0.83 ± 0.00 (-0.01)	**0.85** ± 0.00 (+0.02)	0.33 ± 0.01 (-0.50)	0.62 ± 0.00 (-0.20)
D2	0.68	**0.65**	2.15	1.41

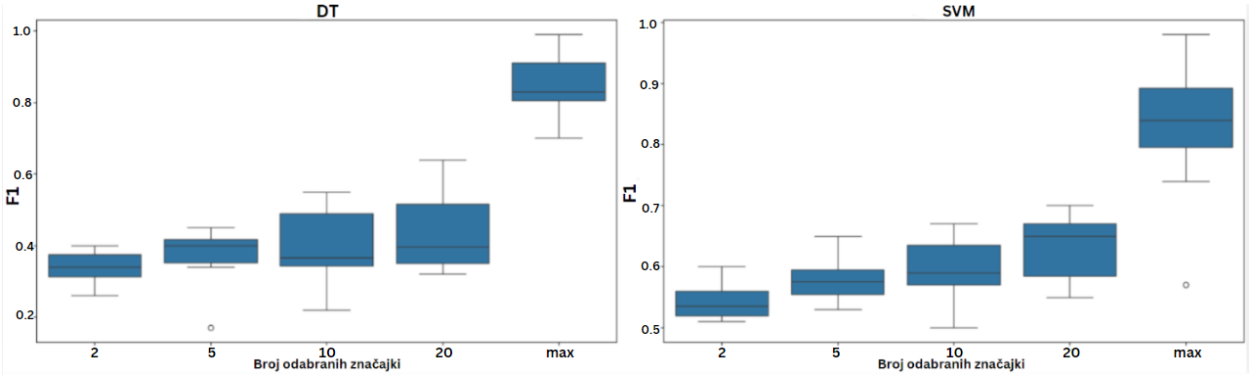


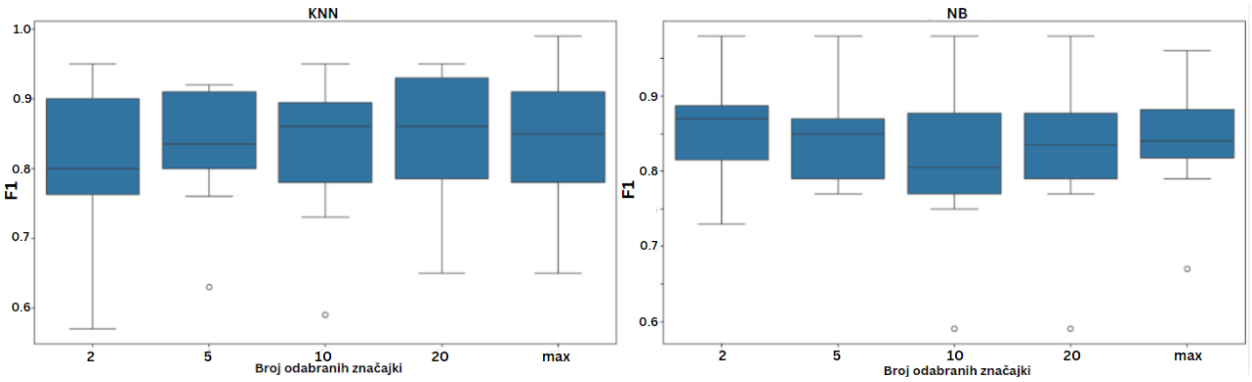
Slika 4.4. Dijagram pravokutnika za KNN, DT, SVM i NB za filter Pearsonova korelacija

Analiza filtera zasnovanih na zajedničkoj informaciji pokazala je da KNN postiže najveću točnost kada se koristi 20 značajki na većini skupova podataka, dok NB najveću točnost postiže s manjim brojem značajki (dva ili pet) za većinu skupova podataka. DT najbolje rezultate postiže s različitim brojem značajki, najčešće između pet i 20 značajki. SVM pokazuje stabilnu točnost s većim brojem značajki, pri čemu se 20 značajki pokazalo optimalnim za većinu skupova podataka. Dijagram pravokutnika na slici 4.5 prikazuje da KNN i NB imaju sličnu ili malo bolju F1 mjeru u odnosu na F1 mjeru sa svim značajkama dok SVM i KNN imaju znatno nižu koja se povećava s brojem značajki

Tablica 4.8. F1 mjera klasifikatora uz primjenu filtera zajedničke informacije

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Pc4	0.86 ± 0.02 (-0.02)	**0.87** ± 0.01 (+0.03)	0.45 ± 0.07 (-0.43)	0.56 ± 0.04 (-0.31)
Cm1	0.81 ± 0.02 (-0.01)	**0.84** ± 0.02 (+0.03)	0.25 ± 0.08 (-0.56)	0.56 ± 0.03 (-0.25)
Mw1	**0.87** ± 0.04 (+0.01)	0.83 ± 0.00 (+0.04)	0.30 ± 0.06 (-0.50)	0.50 ± 0.01 (-0.32)
Kc3	0.73 ± 0.03 (-0.01)	**0.78** ± 0.02 (+0.01)	0.30 ± 0.07 (-0.43)	0.54 ± 0.02 (-0.21)
Mc2	0.62 ± 0.01 (-0.04)	**0.64** ± 0.02 (-0.02)	0.43 ± 0.12 (-0.22)	0.57 ± 0.03 (-0.05)
Pc2	**0.98** ± 0.00 (0.00)	0.96 ± 0.00 (-0.01)	0.14 ± 0.05 (-0.84)	0.69 ± 0.01 (-0.28)
Pc3	0.79 ± 0.01 (-0.05)	**0.81** ± 0.02 (0.00)	0.33 ± 0.02 (-0.50)	0.65 ± 0.02 (-0.16)
Pc1	**0.87** ± 0.01 (-0.02)	**0.87** ± 0.00 (-0.01)	0.32 ± 0.09 (-0.56)	0.62 ± 0.04 (-0.26)
Pc5	**0.97** ± 0.00 (0.00)	0.96 ± 0.00 (0.00)	0.49 ± 0.04 (-0.48)	0.69 ± 0.01 (-0.27)
Mc1	**0.99** ± 0.00 (0.00)	0.97 ± 0.00 (+0.01)	0.55 ± 0.06 (-0.44)	0.70 ± 0.03 (-0.28)
Jm1	0.76 ± 0.00 (-0.02)	**0.77** ± 0.00 (-0.01)	0.34 ± 0.05 (-0.43)	0.56 ± 0.01 (-0.19)
Kc1	0.84 ± 0.00 (+0.01)	**0.86** ± 0.02 (+0.04)	0.35 ± 0.08 (-0.46)	0.71 ± 0.04 (-0.09)
Prosjek	0.79 ± 0.00 (-0.04)	**0.90** ± 0.00 (+0.06)	0.36 ± 0.01 (-0.47)	0.63 ± 0.01 (-0.20)
Medijan	0.83 ± 0.00 (-0.01)	**0.85** ± 0.00 (+0.02)	0.33 ± 0.01 (-0.50)	0.58 ± 0.00 (-0.24)
D2	**0.66**	**0.66**	2.26	1.34





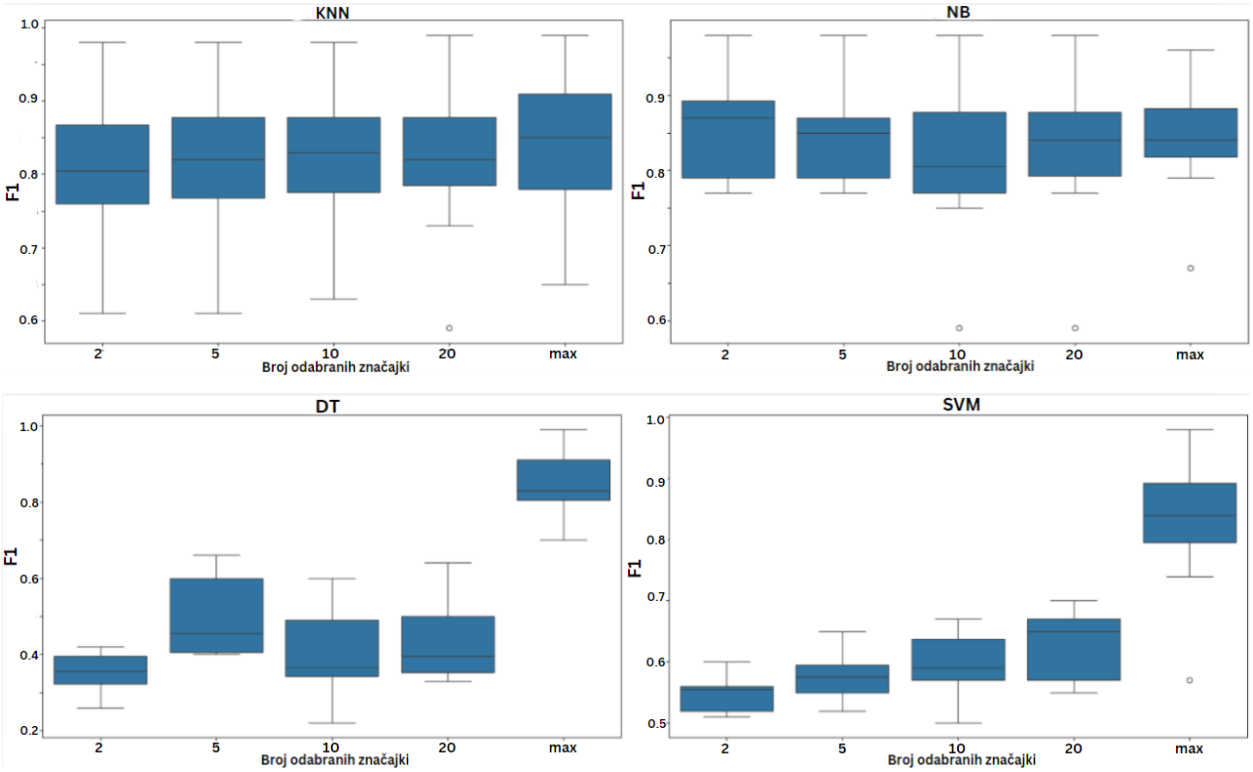
Slika 4.5. Dijagram pravokutnika za KNN, DT, SVM i NB za filter zajednička informacija

Odabir značajki filtrom zasnovanim na ANOVA F-testu rezultira da je DT ostvarilo najbolje rezultate s pet značajki, dok su SVM I KNN pokazali najbolje performanse postižu s većim brojem značajki, poput 20 značajki. NB najveću točnost pokazuje s dvije značajke.Na slici 4.6 prikazane su F1 mjere algoritama ovisno o broju odabranih značajki. Za DT i SVM algoritme, F1 mjera raste s porastom broja značajki, a najbolji rezultati su postignuti korištenjem svih značajki. Kod KNN, F1 mjera ostaje stabilna bez značajnih promjena. Za NB, F1 mjera varira, ali se najveći porast bilježi kada su korištene sve značajke.

Tablica 4.9. F1 mjera klasifikatora s primjenom ANOVA F-testa

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Pc4	0.85 ± 0.01 (-0.03)	**0.88** ± 0.00 (+0.04)	0.45 ± 0.08 (-0.43)	0.55 ± 0.02 (-0.32)
Cm1	0.82 ± 0.01 (+0.02)	**0.85** ± 0.01 (+0.04)	0.27 ± 0.12 (-0.54)	0.55 ± 0.05 (-0.26)
Mw1	**0.86** ± 0.03 (0.00)	0.83 ± 0.01 (+0.04)	0.25 ± 0.09 (-0.55)	0.56 ± 0.01 (-0.24)
Kc3	0.73 ± 0.02 (-0.01)	**0.78** ± 0.04 (+0.01)	0.34 ± 0.06 (-0.39)	0.54 ± 0.02 (-0.21)
Mc2	0.60 ± 0.02 -(0.06)	**0.67** ± 0.00 (+0.01)	0.32 ± 0.10 (-0.33)	0.53 ± 0.03 (-0.09)
Pc2	**0.98** ± 0.00 (0.00)	0.96 ± 0.01 (-0.01)	0.18 ± 0.04 (-0.80)	0.66 ± 0.01 (-0.31)
Pc3	0.79 ± 0.01 (-0.05)	**0.80** ± 0.01 (-0.01)	0.30 ± 0.10 (-0.53)	0.67 ± 0.03 (-0.14)
Pc1	**0.88** ± 0.01(-0.01)	0.87 ± 0.00 (-0.01)	0.35 ± 0.03 (-0.53)	0.60 ± 0.04 (-0.08)
Pc5	**0.97** ± 0.00 (0.00)	0.96 ± 0.00 (0.00)	0.44 ± 0.03 (-0.53)	0.72 ± 0.01 (-0.24)
Mc1	**0.99** ± 0.00 (0.00)	0.97 ± 0.00 (+0.01)	0.58 ± 0.03 (-0.41)	0.69 ± 0.01 (-0.29)
Jm1	0.77 ± 0.00 (-0.01)	**0.78** ± 0.00 (0.00)	0.33 ± 0.02 (-0.44)	0.55 ± 0.01 (-0.25)
Kc1	0.84 ± 0.01 (-0.01)	**0.85** ± 0.01 (-0.02)	0.37 ± 0.08 (-0.46)	**0.69** ± 0.02 (-0.11)

Prosjek	0.76 ± 0.00 (-0.07)	**0.86** ± 0.01 (+0.02)	0.41 ± 0.01 (-0.42)	0.63 ± 0.01 (-0.20)
Medijan	0.83 ± 0.01 (-0.01)	**0.87** ± 0.00 (+0.04)	0.33 ± 0.00 (-0.50)	0.58 ± 0.01 (-0.24)
D2	0.68	**0.60**	2.21	1.34



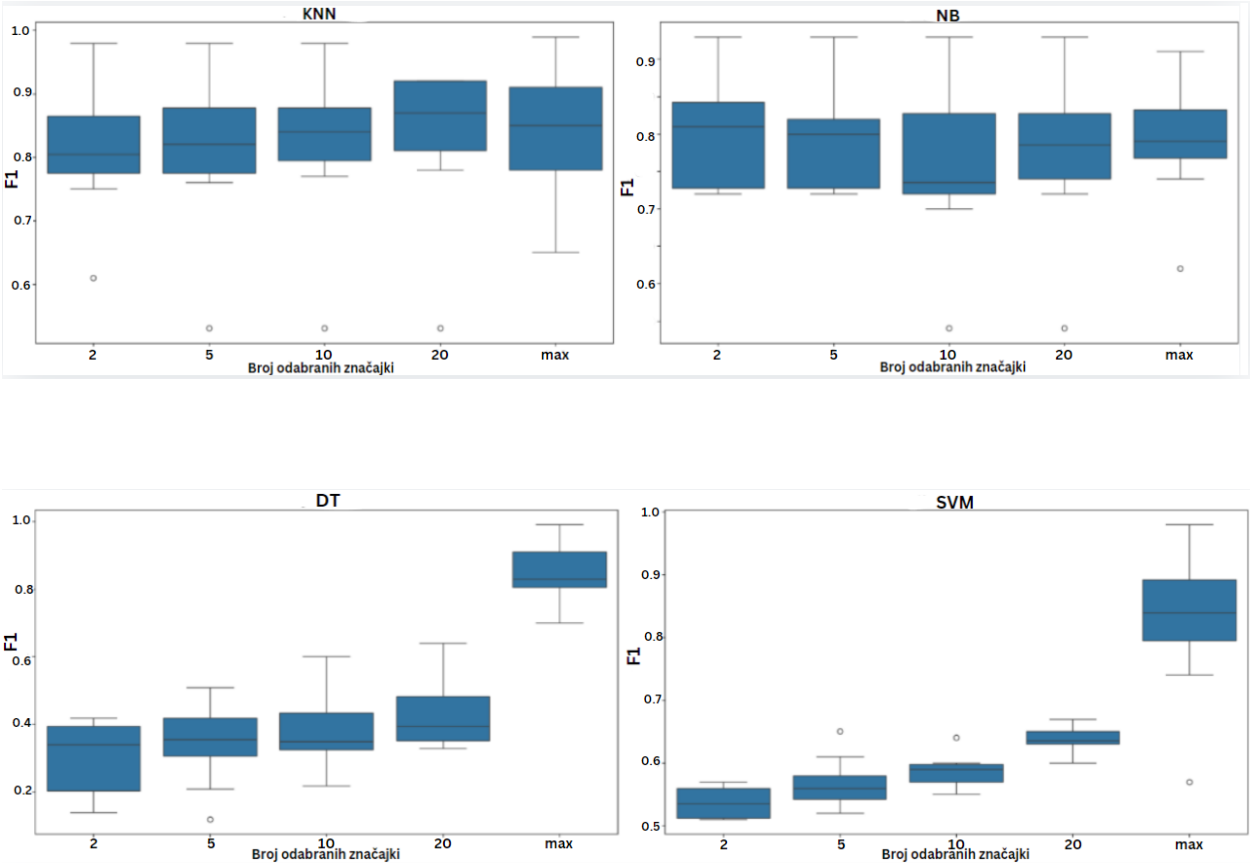
Slika 4.6. Dijagram pravokutnika za KNN, DT, SVM i NB za filter ANOVA F-test

Najbolji rezultati kod slijedne pretrage unaprijed postignuti su korištenjem KNN i SVM s 10 ili 20 značajki. NB i DT ostvarili su veću točnost pri manjem broju značajki. Kod algoritama DT i SVM, F1 mjera se povećava kako raste broj značajki, a najviši rezultati su postignuti kad su korištene sve značajke. Za KNN, F1 mjera ostaje gotovo nepromijenjena bez većih oscilacija. Kod algoritma NB, F1 mjera varira, ali je najbolji rezultat ostvaren kada su sve značajke uključene kao što je prikazano na slici 4.7.

Tablica 4.10. F1 mjera klasifikatora s primjenom slijedne pretrage unaprijed

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Pc4	0.87 ± 0.01 (-0.01)	**0.88** ± 0.01 (+0.04)	0.45 ± 0.07 (-0.43)	0.57 ± 0.03 (-0.30)
Cm1	**0.81** ± 0.01 (+0.01)	0.80 ± 0.00 (-0.01)	0.23 ± 0.05 (-0.58)	0.55 ± 0.05 (-0.26)

Mw1	**0.87** ± 0.03 (+0.01)	0.84 ± 0.01 (+0.05)	0.33 ± 0.08 (-0.47)	0.53 ± 0.01 (-0.29)
Kc3	0.75 ± 0.02 (+0.01)	**0.79** ± 0.02 (+0.02)	0.25 ± 0.13 (-0.48)	0.54 ± 0.02 (-0.21)
Mc2	0.58 ± 0.03 (-0.08)	**0.61** ± 0.02 (-0.05)	0.43 ± 0.10 (-0.22)	0.58 ± 0.03 (-0.04)
Pc2	**0.98** ± 0.00 (0.00)	0.97 ± 0.01 (0.00)	0.22 ± 0.07 (-0.76)	0.68 ± 0.01 (-0.29)
Pc3	0.77 ± 0.03 (-0.07)	**0.78** ± 0.00 (-0.03)	0.30 ± 0.10 (-0.53)	0.67 ± 0.01 (-0.14)
Pc1	**0.88** ± 0.01 (-0.01)	**0.88** ± 0.00 (0.00)	0.28 ± 0.12 (-0.60)	0.60 ± 0.04 (-0.28)
Pc5	**0.97** ± 0.00 (0.00)	**0.97** ± 0.00 (+0.01)	0.47 ± 0.06 (-0.50)	0.71 ± 0.01 (-0.25)
Mc1	**0.99** ± 0.00 (0.00)	**0.98** ± 0.00 (+0.02)	0.55 ± 0.07 (-0.44)	0.70 ± 0.01 (-0.28)
Jm1	**0.77** ± 0.00 (-0.01)	**0.77** ± 0.00 (-0.01)	0.32 ± 0.10 (-0.45)	0.55 ± 0.01 (-0.20)
Kc1	0.85 ± 0.03 (+0.02)	**0.86** ± 0.02 (+0.04)	0.30 ± 0.11 (-0.51)	0.70 ± 0.02 (-0.10)
Prosjek	0.78 ± 0.00 (-0.05)	**0.90** ± 0.00 (+0.06)	0.30 ± 0.01 (-0.53)	0.64 ± 0.01 (-0.19)
Medijan	0.83 ± 0.01 (-0.01)	**0.86** ± 0.01 (+0.03)	0.31 ± 0.02 (-0.52)	0.59 ± 0.00 (-0.23)
D2	0.69	**0.64**	2.26	1.30



Slika 4.7. Dijagram pravokutnika za KNN, DT, SVM i NB za omotač slijedna pretraga unaprijed

Analiza pokazuje značajne razlike u performansama modela ovisno o broju značajki i metodi odabira značajki. KNN često postiže najbolje rezultate s većim brojem značajki, dok NB pokazuje poboljšanje performansi s manjim brojem značajki. SVM najčešće postiže visoke točnosti s većim brojem značajki, dok DT pokazuje varijabilnost u performansama ovisno o broju značajki. Optimalan broj značajki ovisi o specifičnom algoritmu i primjeni. Veći broj značajki obično poboljšava performanse, posebno za algoritme poput KNN i SVM. Međutim, neki algoritmi poput NB mogu biti učinkovitiji s manjim brojem značajki, što ukazuje na potrebu za pažljivim odabirom broja značajki i metodologije prema specifičnim potrebama modela.

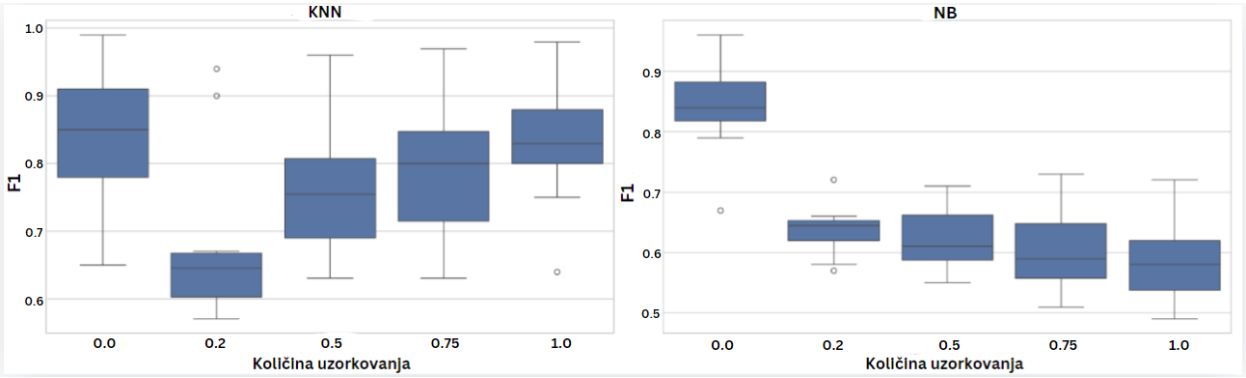
4.4. Analiza učinka uzorkovanja

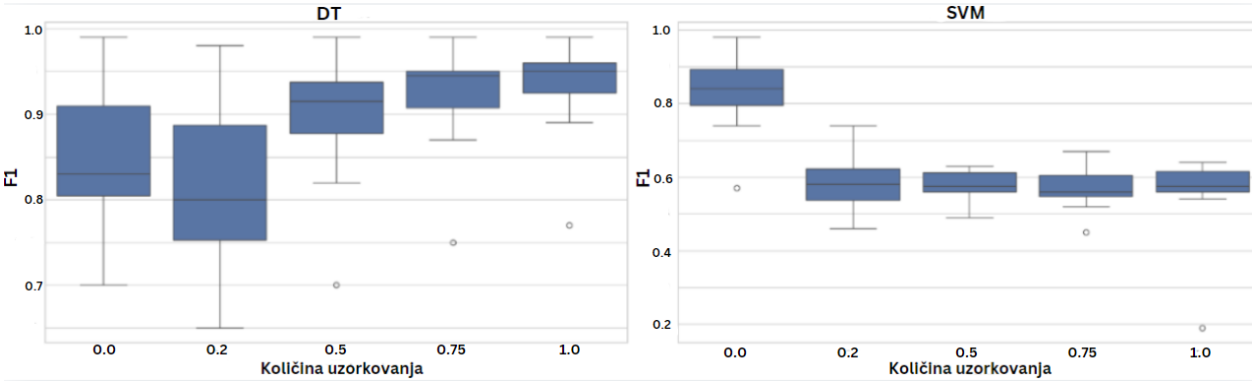
U ovoj analizi razmotreni su učinci različitih tehnika uzorkovanja na performanse modela strojnog učenja. Proučavane su tri ključne metode uzorkovanja: nasumično preuzorkovanje, nasumično poduzorkovanje i SMOTE. U ovoj analizi naglasak je na promjenama u rezultatima ovisno o vrsti uzorkovanja i brojnosti uzoraka manjinske klase, pružajući uvid u prikladne strategije za poboljšanje performansi modela. Kod nasumičnog preuzorkovanja, KNN pokazuje zadovoljavajuće rezultate, iako se performanse značajno razlikuju ovisno o skupu podataka. Preuzorkovanjem manjinske klase KNN uspijeva bolje prepoznati primjere te klase, što poboljšava ukupnu točnost modela. Međutim, s povećanjem uzoraka manjinske klase, dolazi do točke u kojoj dodatni primjeri više ne doprinose poboljšanju točnosti modela, već se bilježi stagnacija. Nasuprot tome, NB se pokazuje vrlo učinkovitim. Dodavanjem više primjera manjinske klase, ovaj algoritam poboljšava svoje sposobnosti klasifikacije jer se povećava broj dostupnih informacija za učenje. Preuzorkovanje poboljšava uravnoteženost između klasa, omogućujući NB precizniju analizu odnosa među varijablama. Najbolje performanse pokazuje DT kao što je prikazano u tablici 4.11. Povećanje broja primjera manjinske klase poboljšava sposobnost modela da točno klasificira sve klase, jer DT u potpunosti koristi sve dostupne informacije u skupovima podataka. Preuzorkovanje osigurava bolja uravnotežena stabla, što poboljšava točnost i robusnost modela. SVM pokazuje solidne performanse, ali nije na razini kao kod drugih algoritama. Povećanje broja primjera manjinske klase donekle pomaže SVM, ali se ne primjećuje značajno poboljšanje. Preuzorkovanje ponekad može zbuniti SVM, jer dodavanje previše primjera može dovesti do smanjenja njegove učinkovitosti, posebno ako podaci nisu dovoljno raznoliki. Slika 4.8 prikazuje dijagrame pravokutnika s rezultatima F1 mjere za različite algoritme u ovisnosti o količini uzrokovanja. DT pokazuje stalno poboljšanje performansi s porastom uzorkovanja, dok su najbolje F1 mjere postignute pri većim količinama. S druge strane, algoritam SVM postiže najbolje

rezultate bez uzorkovanja, no s povećanjem uzorkovanja dolazi do pada F1 mjera. Za algoritam KNN vidljiv je početni pad u performansama, no kasnije se F1 mjere poboljšavaju, čime se postižu bolji rezultati u odnosu na početne. NB postiže najviše F1 mjere bez uzorkovanja, ali s rastom količine uzoraka njegove performanse opadaju. Općenito, DT i KNN bolje reagiraju na povećanje uzorkovanja, dok SVM i NB gube na učinkovitosti.

Tablica 4.11. F1 mjera pri nasumičnom preuzorkovanju s omjerom 1:1

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Pc4	0.82 ± 0.00 (-0.06)	0.53 ± 0.00 (-0.31)	**0.96** ± 0.00 (+0.08)	0.54 ± 0.00 (-0.33)
Cm1	0.79 ± 0.01 (-0.01)	0.62 ± 0.02 (-0.19)	**0.94** ± 0.01 (+0.13)	0.56 ± 0.01 (-0.25)
Mw1	0.84 ± 0.00 (-0.02)	0.71 ± 0.01 (-0.08)	**0.95** ± 0.01 (+0.15)	0.56 ± 0.00 (-0.26)
Kc3	0.74 ± 0.01 (0.00)	0.58 ± 0.02 (-0.19)	**0.92** ± 0.01 (+0.19)	0.61 ± 0.01 (-0.14)
Mc2	0.63 ± 0.03 (-0.03)	0.54 ± 0.02 (-0.12)	**0.77** ± 0.02 (+0.12)	0.62 ± 0.00 (0.00)
Pc2	**0.98** ± 0.00 (0.00)	0.49 ± 0.01 (-0.48)	**0.98** ± 0.00 (0.00)	0.56 ± 0.00 (-0.41)
Pc3	0.83 ± 0.00 (-0.01)	0.72 ± 0.01 (-0.09)	**0.95** ± 0.00 (+0.12)	0.56 ± 0.00 (-0.25)
Pc1	0.88 ± 0.00 (-0.01)	0.61 ± 0.01 (-0.27)	**0.96** ± 0.00 (+0.08)	0.59 ± 0.02 (-0.29)
Pc5	0.97 ± 0.00 (0.00)	0.53 ± 0.00 (-0.43)	**0.99** ± 0.00 (+0.02)	0.63 ± 0.00 (-0.33)
Mc1	0.90 ± 0.00 (-0.09)	0.55 ± 0.02 (-0.41)	**0.99** ± 0.00 (0.00)	0.51 ± 0.00 (-0.47)
Jm1	0.75 ± 0.00 (-0.03)	0.50 ± 0.00 (-0.28)	**0.89** ± 0.00 (+0.12)	0.19 ± 0.04 (-0.56)
Kc1	0.83 ± 0.00 (0.00)	0.61 ± 0.01 (-0.21)	**0.91** ± 0.00 (+0.10)	0.60 ± 0.01 (-0.20)
Prosjek	0.78 ± 0.00 (-0.05)	0.67 ± 0.00 (-0.17)	**0.92** ± 0.01 (+0.09)	0.54 ± 0.00 (-0.29)
Medijan	0.83 ± 0.00 (-0.01)	0.55 ± 0.00 (-0.28)	**0.95** ± 0.00 (+0.12)	0.58 ± 0.00 (-0.24)
D2	0.72	1.41	**0.30**	1.60





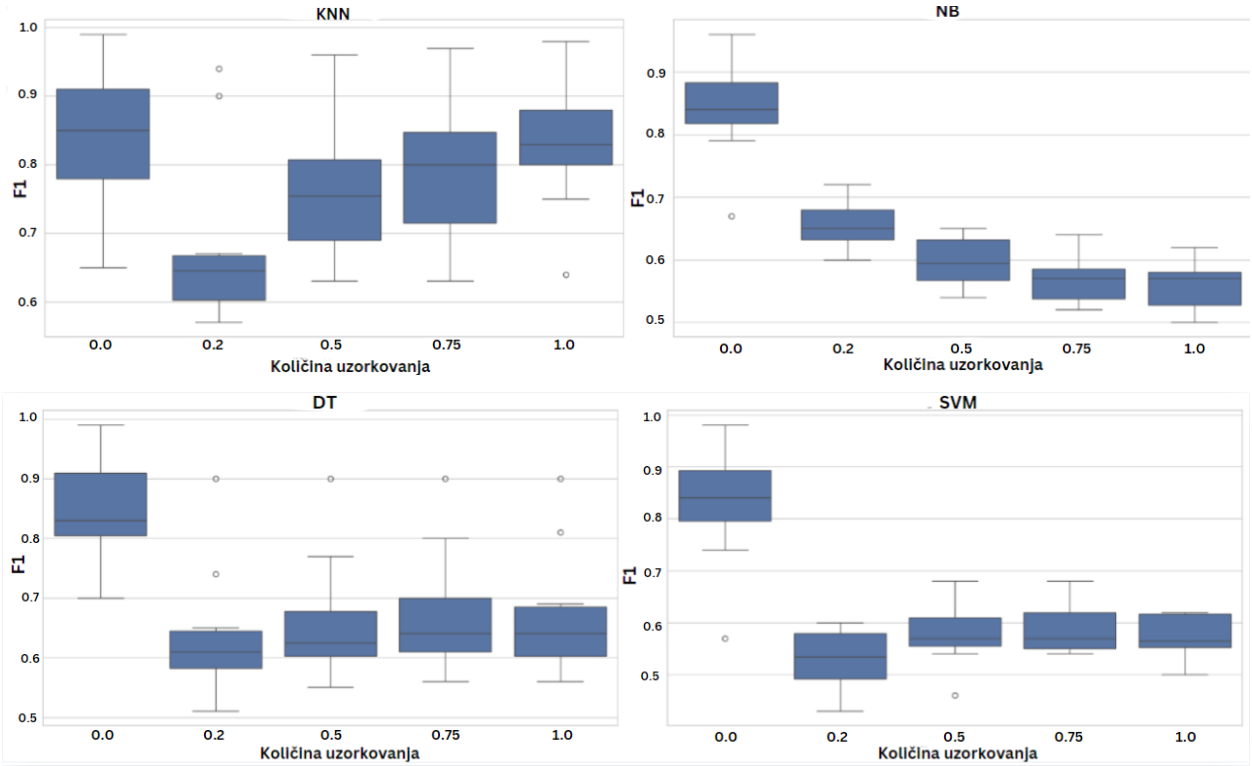
Slika 4.8. Dijagram pravokutnika za KNN, DT, SVM i NB za metodu nasumičnog preuzorkovanja

Kod nasumičnog poduzorkovanja, KNN značajno gubi na performansama. Poduzorkovanjem većinske klase smanjuje se količina informacija na temelju koje algoritam može uspoređivati bliskost susjeda, što rezultira slabijim prepoznavanjem obrazaca i lošijom klasifikacijom. Manji broj uzoraka većinske klase dovodi do gubitka preciznosti, posebno u složenijim skupovima podataka. Najveću F1 mjeru pokazuje DT kao sto je prikazano u tablici 4.12. Nasumično poduzorkovanje kod NB ne utječe značajno na smanjenje točnosti, kao kod drugih algoritama. Ovaj algoritam pokazuje otpornost na smanjenje uzoraka, jer poduzorkovanje ne remeti njegovu sposobnost predviđanja odnosa među varijablama. Iako poduzorkovanje ne poboljšava performanse, NB održava stabilne rezultate, što ga čini robusnim i u ovoj metodi uzorkovanja. Nasuprot tome, DT pokazuje značajno smanjenje performansi. Smanjenje broja uzoraka većinske klase ograničava količinu informacija dostupnih za generiranje stabla, što rezultira nepotpunim ili neuravnoteženim stablima. To dovodi do slabijih performansi u klasifikaciji, posebno kod većinskih klasa, koje sada imaju manje uzoraka za treniranje. SVM bilježi pad u performansama, jer poduzorkovanjem dolazi do gubitka informacija iz većinske klase. S obzirom na to da se stroj potpornih vektora oslanja na jasnu razdiobu između klasa, smanjenje uzoraka većinske klase otežava modelu postizanje prikladne granice između klasa. Poduzorkovanje tako može rezultirati slabijom sposobnošću modela da pravilno razlikuje klase. Dijagrami pravokutnika na slici 4.9 za metodu poduzrokovanja prikazuju značajan pad F1 mjere u odnosu na početnu za sve algoritme.

Tablica 4.12. F1 mjera pri nasumičnom poduzorkovanju s omjerom 1:1

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Pc4	0.57 ± 0.03 (-0.31)	0.52 ± 0.02 (-0.32)	**0.81** ± 0.01 (-0.07)	0.59 ± 0.01 (-0.28)

Cm1	0.52 ± 0.05 (-0.28)	0.50 ± 0.07 (-0.31)	**0.60** ± 0.05 (-0.21)	0.53 ± 0.01 (-0.28)
Mw1	0.62 ± 0.09 (-0.24)	**0.69** ± 0.04 (-0.10)	0.64 ± 0.08 (-0.16)	0.56 ± 0.00 (-0.26)
Kc3	0.57 ± 0.04 (-0.17)	0.56 ± 0.04 (-0.21)	**0.63** ± 0.05 (-0.10)	0.55 ± 0.01 (-0.20)
Mc2	0.61 ± 0.03 (-0.06)	0.58 ± 0.02 (-0.08)	0.56 ± 0.03 (-0.09)	**0.75** ± 0.02 (-0.13)
Pc2	0.66 ± 0.13 (-0.32)	0.70 ± 0.09 (-0.27)	**0.74** ± 0.10 (-0.24)	0.54 ± 0.01 (-0.43)
Pc3	0.62 ± 0.03 (-0.22)	0.64 ± 0.07 (-0.17)	**0.68** ± 0.03 (-0.15)	0.60 ± 0.03 (-0.21)
Pc1	0.67 ± 0.05 (-0.22)	0.60 ± 0.04 (-0.28)	**0.69** ± 0.05 (-0.19)	0.56 ± 0.01 (-0.32)
Pc5	**0.91** ± 0.01 (-0.06)	0.61 ± 0.05 (-0.35)	0.90 ± 0.01 (-0.07)	0.65 ± 0.00 (-0.31)
Mc1	0.77 ± 0.03 (-0.22)	0.67 ± 0.08 (-0.29)	**0.84** ± 0.04 (-0.15)	0.51 ± 0.01 (-0.47)
Jm1	0.61 ± 0.01 (-0.17)	0.50 ± 0.02 (-0.28)	0.61 ± 0.01 (-0.16)	**0.62** ± 0.02 (-0.12)
Kc1	**0.70** ± 0.01 (-0.13)	0.61 ± 0.01 (-0.21)	0.67 ± 0.01 (-0.14)	0.62 ± 0.01 (-0.18)
Prosjek	0.66 ± 0.02 (-0.17)	0.62 ± 0.00 (-0.22)	**0.74** ± 0.01 (-0.09)	0.52 ± 0.01 (-0.31)
Medijan	0.62 ± 0.01 (-0.22)	0.59 ± 0.03 (-0.24)	**0.66** ± 0.01 (-0.17)	0.58 ± 0.00 (-0.24)
D2	1.18	1.34	**1.16**	1.43



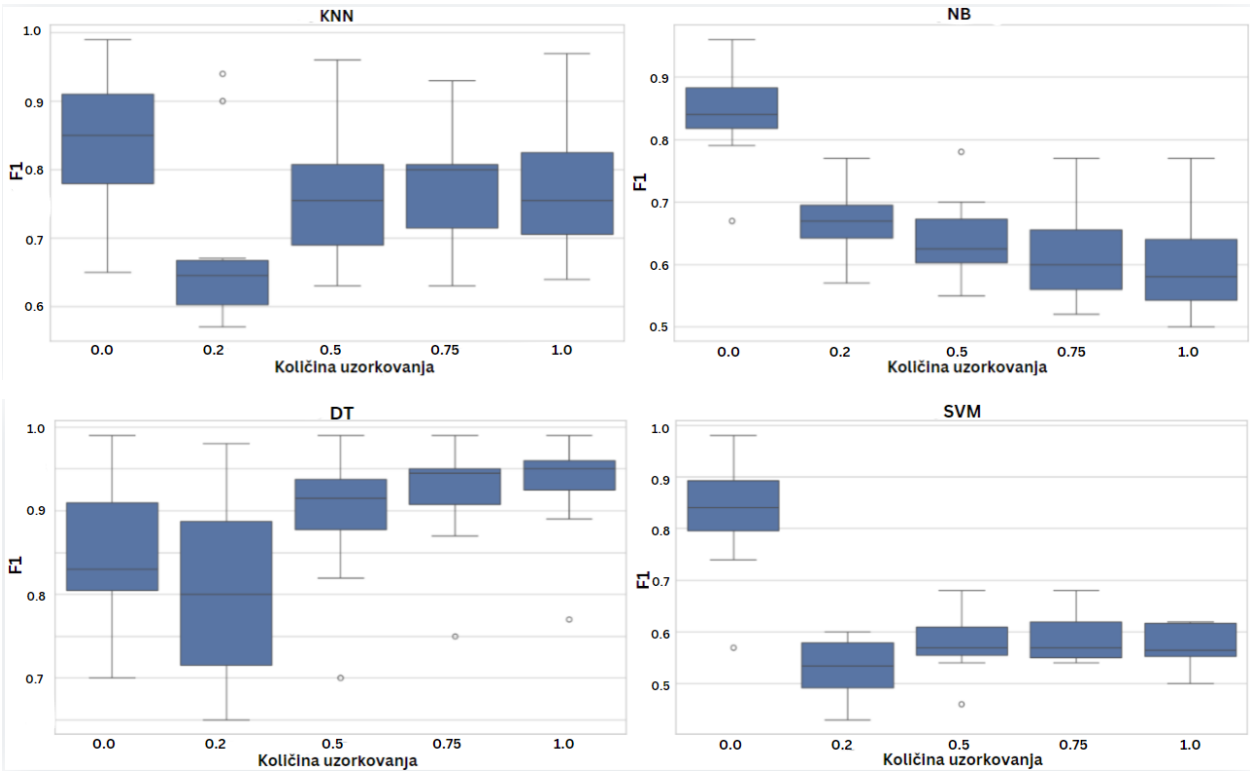
Slika 4.9. Dijagram pravokutnika za KNN, DT, SVM i NB za metodu nasumičnog poduzorkovanja

Kod primjene SMOTE-a, KNN pokazuje poboljšane performanse u odnosu na poduzorkovanje, jer sintetičko generiranje uzoraka manjinske klase omogućuje modelu bolje razumijevanje strukture tih podataka. Iako SMOTE pomaže u boljoj klasifikaciji manjinske klase, performanse nisu na razini nasumičnog preuzorkovanja, što ukazuje na to da KNN bolje funkcionira s pravim, a ne sintetičkim uzorcima. NB se također pokazuje stabilnim, ali s nešto nižim performansama nego kod preuzorkovanja. Sintetički podaci, iako pomažu u uravnoteženost klasa, ne utječu značajno na poboljšanje ovog algoritma. Ipak, SMOTE osigurava bolje rezultate od poduzorkovanja jer dodaje nove uzorke manjinske klase, što omogućuje algoritmu bolji uvid u obrasce. Najveću točnost pokazuje DT. Sintetički uzorci manjinske klase pomažu u izgradnji bolje strukturiranih stabala, što povećava sposobnost modela da točno klasificira manjinsku klasu. Iako SMOTE ne daje toliko visoke performanse kao preuzorkovanje, ipak se pokazuje učinkovitim u balansiranju klasa i poboljšanju ukupne točnosti. Mješovite rezultate pokazuje stroj potpornih vektora. Iako sintetički podaci pomažu u balansiranju klasa, SVM nije osobito učinkovit u radu sa sintetičkim uzorcima. Performanse su bolje nego kod poduzorkovanja, ali SMOTE ne daje toliko dobre rezultate kao kod nekih drugih algoritama, što ukazuje na osjetljivost SVM na strukturu podataka. Na slici 4.10 su prikazane F1 mjere za algoritme u odnosu na količinu uzorkovanja. Uočen je pad performansi za sve algoritma s povećanjem uzorkovanja, iako se kod KNN performanse poboljšavaju nakon početnog pada, dok kod NB-a F1 mjere kontinuirano opadaju.

Tablica 4.13. F1 mjera pri algoritmu SMOTE s omjerom 1:1

Skup podataka	**KNN**	**NB**	**DT**	**SVM**
Pc4	0.76 ± 0.01 (-0.12)	0.55 ± 0.01 (-0.29)	**0.92** ± 0.00 (+0.04)	0.70 ± 0.01 (-0.17)
Cm1	0.67 ± 0.02 (-0.13)	0.64 ± 0.02 (-0.17)	**0.87** ± 0.01 (+0.06)	0.57 ± 0.00 (-0.24)
Mw1	0.71 ± 0.01 (-0.15)	0.73 ± 0.01 (-0.06)	**0.88** ± 0.01 (+0.08)	0.61 ± 0.01 (-0.21)
Kc3	0.69 ± 0.02 (-0.05)	0.56 ± 0.02 (-0.21)	**0.85** ± 0.01 (+0.12)	0.55 ± 0.00 (-0.20)
Mc2	0.64 ± 0.03 (-0.02)	0.57 ± 0.02 (-0.09)	**0.72** ± 0.02 (+0.07)	0.59 ± 0.02 (-0.03)
Pc2	0.90 ± 0.00 (-0.08)	0.66 ± 0.01 (-0.31)	**0.99** ± 0.00 (+0.01)	0.57 ± 0.01 (-0.40)
Pc3	0.78 ± 0.01 (-0.06)	0.70 ± 0.02 (-0.11)	**0.88** ± 0.00 (+0.05)	0.63 ± 0.00 (-0.18)
Pc1	0.75 ± 0.01 (-0.14)	0.64 ± 0.01 (-0.24)	**0.90** ± 0.00 (+0.02)	0.59 ± 0.01 (-0.29)
Pc5	0.97 ± 0.00 (0.00)	0.53 ± 0.01 (-0.43)	**0.98** ± 0.00 (+0.01)	0.32 ± 0.00 (-0.64)
Mc1	0.95 ± 0.00 (-0.04)	0.77 ± 0.00 (-0.19)	**0.99** ± 0.00 (0.00)	0.53 ± 0.01 (-0.45)
Jm1	0.75 ± 0.00 (-0.03)	0.50 ± 0.01 (-0.28)	**0.83** ± 0.00 (+0.06)	0.18 ± 0.00 (-0.57)

Kc1	0.80 ± 0.01 (-0.03)	0.61 ± 0.01 (-0.21)	**0.87** ± 0.00 (+0.06)	0.60 ± 0.01 (-0.20)
Prosjek	0.67 ± 0.00 (-0.16)	0.62 ± 0.01 (-0.22)	**0.86** ± 0.00 (+0.03)	0.52 ± 0.01 (-0.31)
Medijan	0.76 ± 0.01 (-0.08)	0.59 ± 0.00 (-0.24)	**0.89** ± 0.00 (+0.06)	0.60 ± 0.01 (-0.22)
D2	0.89	1.30	**0.46**	1.61



Slika 4.10. Dijagram pravokutnika za KNN, DT, SVM i NB za metodu SMOTE

Zaključno, različite tehnike uzorkovanja imaju različit utjecaj na performanse algoritama strojnog učenja. Kod KNN, preuzorkovanje daje najbolje rezultate, dok poduzorkovanje značajno smanjuje točnost, a SMOTE djelomično poboljšava performanse. NB je vrlo stabilan i otporan na promjene, s najboljim rezultatima kod preuzorkovanja, dok poduzorkovanje ne utječe značajno na točnost. DT najbolje funkcionira s preuzorkovanjem, dok poduzorkovanje negativno utječe na performanse. SVM pokazuje solidne rezultate s preuzorkovanjem, ali je osjetljiv na poduzorkovanje i SMOTE, s najmanjim poboljšanjem kod sintetičkih podataka.

Analizom rezultata primjene različitih tehnika uzorkovanja na neuravnotežene skupove podataka uočeni su značajni trendovi u performansama modela. Omjer uzorkovanja, bilo preuzorkovanjem, poduzorkovanjem ili primjenom SMOTE-a, značajno utječe na metrike poput točnosti, preciznosti, odziva i F1 mjere. Točnost modela obično se smanjuje kako se povećava omjer preuzorkovanja. Na nižim omjerima uzorkovanja, kao što je 0.5, točnost je viša, jer model

je bolje uravnotežen između klasa. S povećanjem omjera uzorkovanja, primjerice na 1.0, dolazi do blagog smanjenja točnosti. Ovo smanjenje može se pripisati pretjeranom preuzorkovanju manjinske klase, što može uzrokovati da model postane skloniji prekomjernom generaliziranju. Što se tiče preciznosti, ona za većinsku klasu ostaje visoka kroz sve omjere uzorkovanja. S druge strane, preciznost za manjinsku klasu varira. Na nižim omjerima uzorkovanja, preciznost manjinske klase je niža zbog manjka primjera, dok se s povećanjem uzorkovanja, osobito s primjenom SMOTE-a, preciznost manjinske klase postepeno poboljšava. Međutim, iako preciznost raste, odziv za manjinsku klasu ne raste uvijek proporcionalno, što ukazuje na to da model i dalje ima poteškoća u prepoznavanju svih instanci manjinske klase. F1 mjere za većinsku klasu ostaje konzistentan i visok kroz sve metode uzorkovanja. Za manjinsku klasu, F1 mjere se u početku poboljšava s većim omjerima preuzorkovanja, no nakon određenog praga, primjerice 0.75 ili 1.0, dolazi do stagnacije ili blagog pada. Ovaj fenomen može se objasniti činjenicom da prekomjerno preuzorkovanje stvara sintetičke primjere koji možda ne predstavljaju stvarne obrasce u podacima, što može dovesti do lažno pozitivnih klasifikacija. Generalno, povećanjem omjera uzorkovanja dolazi do poboljšanja u klasifikaciji manjinske klase, no uz cijenu blagog smanjenja ukupne točnosti. Preciznost i F1 mjere za manjinsku klasu rastu do određenog praga, nakon čega performanse mogu stabilizirati ili čak pogoršati. Optimalni omjer uzorkovanja ovisi o specifičnostima podataka i modela, pri čemu se SMOTE često pokazuje učinkovitijim u balansiranju klasa bez značajnog gubitka informacija. Zaključno, primjena tehnika uravnoteženja klasa dovodi do značajnog poboljšanja performansi klasifikacije manjinskih klasa, uz potencijalno smanjenje točnosti za većinsku klasu. Izbor tehnike ovisi o specifičnostima problema i skupu podataka, a balansiranje između preciznosti i odziva predstavlja ključan izazov u prilagodbi modela strojnog učenja.

5. ZAKLJUČAK

Na temelju provedenog istraživanja o predviđanju pogrešaka u programskoj podršci korištenjem različitih klasifikatora, tehnika odabira značajki i tehnika uzorkovanja, zaključeno je da ne postoji univerzalna kombinacija koja uvijek donosi najbolje rezultate. Korišteni skupovi podataka bili su neuravnoteženi, što je znatno utjecalo na performanse modela, te je zbog toga bilo potrebno primijeniti specifične metode za rješavanje problema neuravnoteženosti klasa. Jedna od najučinkovitijih kombinacija pokazala se korištenje NB uz tehniku odabira značajki pomoću omotača slijedne pretrage unaprijed. Ova kombinacija posebno je došla do izražaja kada se radilo s manjim brojem značajki. NB kao algoritam efikasno obrađuje smanjeni broj značajki, omogućujući time bržu i točniju klasifikaciju. Slijedna pretraga unaprijed dodaje značajke jednu po jednu, čime osigurava da se u model uključe samo one značajke koje najviše pridonose preciznosti klasifikacije, čime se izbjegava nepotrebna složenost modela. Također, vrlo dobri rezultati postignuti su kombinacijom DT s tehnikom nasumičnog preuzorkovanja. Nasumično preuzorkovanje povećava broj primjeraka manjinske klase dupliciranjem postojećih podataka te na taj način smanjuje pristranost modela prema većinskoj klasi. DT, zahvaljujući svojoj hijerarhijskoj strukturi donošenja odluka, učinkovito koristi ovakve uravnotežene skupove podataka, što omogućuje preciznije predviđanje pogrešaka u programskoj podršci. Iako su ove kombinacije dale najbolje rezultate u ovom istraživanju, nije moguće odrediti jednu jedinstvenu kombinaciju klasifikatora, tehnika odabira značajki i tehnika uzorkovanja koja će uvijek osigurati najbolje performanse. Učinkovitost modela uvelike ovisi o specifičnim karakteristikama skupa podataka. Primjerice, dimenzionalnost podataka, ravnoteža između različitih klasa, prisutnost šuma te međusobna korelacija među značajkama ključni su faktori koji utječu na izbor odgovarajućih metoda. Osim toga, vrsta klasifikacijskog problema također igra važnu ulogu. Različite tehnike mogu biti prikladnije ovisno o tome radi li se o binarnoj klasifikaciji, gdje model predviđa između dvije klase, ili višeklasnoj klasifikaciji, gdje postoji više mogućih ishoda. Pri tome je potrebno prilagoditi metode kako bi se postigla optimalna preciznost modela. Na kraju, važni su i prioriteti u evaluaciji modela. U nekim situacijama točnost modela može biti najvažniji kriterij, dok u drugim slučajevima veći značaj mogu imati preciznost, odziv ili F1 mjera. Zbog toga se model mora prilagoditi specifičnim ciljevima i zadacima kako bi se postigla uravnotežena izvedba.

Zaključno, uspješno predviđanje pogrešaka u programskoj podršci zahtijeva prilagodbu različitih metoda specifičnostima podataka i zadatka. Premda su kombinacije NB s omotačem slijedne pretrage unaprijed i DT s nasumičnim preuzorkovanjem dale najbolje rezultate u ovom

istraživanju, jasno je da svaka nova situacija može zahtijevati drugačiji pristup. Potrebno je pažljivo ispitati karakteristike podataka, prirodu problema te ciljeve predviđanja kako bi se odabrale optimalne kombinacije tehnika za postizanje najboljih rezultata.

Na temelju provedenog istraživanja mogu se predložiti sljedeća poboljšanja i proširenja. Prvo, trebalo bi razmotriti uključivanje dodatnih klasifikatora poput XGBoost-a ili neuralnih mreža. Ovi algoritmi mogu ponuditi bolju generalizaciju, osobito na kompleksnijim skupovima podataka s velikom dimenzionalnošću. Njihova primjena mogla bi dovesti do poboljšanja u predviđanju pogrešaka u programskoj podršci. Drugo, analiza bi se mogla proširiti na dodatne skupove podataka. Na taj način bi se osigurala veća generalizacija rezultata te bi se mogla utvrditi učinkovitost predloženih kombinacija na različitim domenama. Treće, istraživanje dodatnih tehnika za uravnoteženu klasu moglo bi doprinijeti boljim rezultatima. Primjenom naprednijih metoda, poput ADASYN-a ili prilagođenih mjera troška, moglo bi se dodatno poboljšati ravnoteža između klasa i smanjiti pristranost modela. Nadalje, preporučuje se ispitivanje drugih tehnika za odabir značajki, primjerice genetskih algoritama ili tehnika temeljenih na važnosti značajki. Time bi se moglo otkriti dodatne značajke koje imaju ključnu ulogu u poboljšanju preciznosti klasifikacije. Uz to, analiza osjetljivosti modela na promjene parametara i različite pristupe uzorkovanju pridonijela bi boljoj optimizaciji i prilagodbi metoda specifičnostima svakog skupa podataka. Konačno, primjena razvijenih modela na stvarnim projektima omogućila bi uvid u njihovu praktičnu primjenjivost. Time bi se testirala njihova učinkovitost u stvarnim uvjetima rada, što bi pridonijelo otkrivanju novih izazova te potencijalnih ograničenja metoda. Ovakav pristup omogućio bi daljnje prilagodbe i poboljšanja kako bi se razvile još učinkovitije tehnike za predviđanje pogrešaka u programskoj podršci. Na taj način, kombinacijom teorijskih spoznaja i praktičnih iskustava, mogla bi se dodatno unaprijediti kvaliteta i pouzdanost predviđanja modela u budućim istraživanjima i primjenama.

LITERATURA

[1] T. O'Connor, FBI's Virtual Case File project, Federal Bureau of Investigation, SAD, 2005, dostupno na: https://spectrum.ieee.org/who-killed-the-virtual-case-file [2.9.2024.]

[2] P. Thurrott, Windows Vista Secrets: A Review, Paul Thurrott's Supersite for Windows, Indianapolis, 2007.

[3] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, Analysis of the Heartbleed OpenSSL Bug, Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security (CCS), str. (551-567), Scottsdale, Arizona, 2014.

[4] NASA, Mars Climate Orbiter Mishap Investigation Board Report, NASA, SAD, 1999, dostupno na: https://llis.nasa.gov/lesson/641 [2.9.2024.]

[5] A. Goldstein, Healthcare.gov: The Launch and the Fallout, The Washington Post, SAD, 2013, dostupno na: https://d3.harvard.edu/platform-rctom/submission/the-failed-launch-of-www-healthcare-gov [2.9.2024.]

[6] B. Krebs, Wannacry Ransomware Attack Analysis [online], Krebs on Security, SAD, 2017, dostupno na: https://ijgis.pubpub.org/pub/57la3v2w/release/1 [2.9.2024.]

[7] N. E. Fenton, M. Neil, A critique of software defect prediction models, IEEE Transactions on Software Engineering, br. 25, sv. 5, str. (675-689), svibanj 1999.

[8] V. A. Prakash, D. V. Ashoka, V. N. M. Aradya, Application of data mining techniques for defect detection and classification, u: Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA) 2014, Springer International Publishing, Cham, sv. 1, str. (387-395), 2015.

[9] J. L. Lions, ARIANE 5 Flight 501 Failure, Report by the Inquiry Board, European Space Agency, Pariz, 1996, dostupno na: https://www.esa.int/Newsroom/Press_Releases/Ariane_501_-_Presentation_of_Inquiry_Board_report [2.9.2024.]

[10] N. G. Leveson, T. Turner, An investigation of the Therac-25 accidents, IEEE Computer, br. 7, sv. 26, str. (18-41), srpanj 1993.

[11] G. Tassey, The economic impacts of inadequate infrastructure for software testing, National Institute of Standards and Technology (NIST), SAD, 2002, dostupno na: https://www.nist.gov/system/files/documents/director/planning/report02-3.pdf [2.9.2024.]

[12] J. Nam, W. Fu, S. Kim, T. Menzies, L. Tan, Heterogeneous defect prediction, IEEE Transactions on Software Engineering, br. 44, sv. 9, str. (874-896), 2018.

[13] G. Czibula, Z. Marian, I. G. Czibula, Software defect prediction using relational association rule mining, Information Sciences, br. 3,sv. 26, str. (261-275), 2014.

[14] A. Iqbal, S. Aftab, U. Ali, Z. Nawaz, L. Sana, M. Ahmad, A. Husen, Performance Analysis of Machine Learning, International Journal of Advanced Computer Science and Applications, br. 15, sv. 9, str. (1-21), 2019.

[15] D. Charte, M. A. Arroyo, A. Fernández, F. Herrera, Revisiting data complexity metrics based on morphology for overlap and imbalance: snapshot, new overlap number of balls metrics and singular problems prospect, Knowledge and Information Systems, br. 12, sv. 32, str. (2282-2295), 2020.

[16] D. Charte, M. A. Arroyo, A. Fernández, A. Hamdy, A. El-Laithy, F. Herrera, SMOTE and Feature Selection for More Effective Bug Severity Prediction, International Journal of Software Engineering and Knowledge Engineering, br.4, sv.28, str. (1-22) 2018.

[17] N. Japkowicz, S. Stephen, The class imbalance problem: A systematic study, Intelligent Data Analysis, br. 6, sv. 5, str. (429-449), 2002.

[18] H. He, E. A. Garcia, Learning from imbalanced data, IEEE Transactions on Knowledge and Data Engineering, br. 21, sv. 9, str. (1263-1284), rujan 2009.

[19] Y. Sun, A. K. Wong, M. S. Kamel, Classification of imbalanced data: A review, International Journal of Pattern Recognition and Artificial Intelligence, br. 23, sv. 4, str. (687-719), 2009.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer, SMOTE: Synthetic minority over-sampling technique, Journal of Artificial Intelligence Research, sv. 16, str. (321-357), 2002.

[21] X. Y. Liu, J. Wu, Z. H. Zhou, Exploratory undersampling for class-imbalance learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), br. 39, sv. 2, str. (539-550), travanj 2009.

[22] V. García, R. A. Mollineda, J. S. Sánchez, The class imbalance problem in pattern classification and learning, Pattern Recognition and Data Mining, Springer, str. (129-139), 2015.

[23] G. M. Weiss, Mining with rarity: a unifying framework, ACM SIGKDD Explorations Newsletter, br. 6, sv. 1, str. (7-19), 2004.

[24] X. Zhu, X. Wu, Q. Chen, Eliminating class noise in large datasets, u: Proceedings of the 19th International Conference on Machine Learning, str. (920-927), 2003.

[25] A. C. Lorena, L. P. F. Garcia, J. Lehmann, M. C. P. Souto, T. K. Ho, How Complex is your classification problem? A survey on measuring classification complexity, IEEE Transactions on Neural Networks and Learning Systems, br. 7, sv. 30, str. (1700-1712), 2019.

[26] Chapman, M., Callis, P., and Jackson, W. Metrics data program, nasa iv and v facility, 2004.

[27] J. S. Shirabad, T. J. Menzies, et al., The promise repository of software engineering databases, School of Information Technology and Engineering, University of Ottawa, Kanada, br.5, sv.1, str.(2-10), 2005.

[28] I. Arora, V. Tetarwala, A. Saha, Open Issues in Software Defect Prediction, Proceedings of the 5th International Conference on Advances in Computing and Communications, str. (101-106), Kochi, 2014.

[29] P. Afrić, L. Sikić, A. S. Kurdija, M. Silić, "REPD: Source Code Defect Prediction as Anomaly Detection, Proceedings of the 2020 International Conference on Software Engineering, str. (73-81), 2020.

[30] S. Gupta, A. Gupta, A set of measures designed to identify overlapped instances in software defect prediction, Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering, str. (147-157), 2017.

[31] Y. Ma, Y. Li, J. Lu, P. Sun, Y. Sun, X. Zhu, Data Complexity Analysis for Software Defect Detection, Software Quality Journal, br. 2, sv. 26, str. (383-406), 2018.

[32] V. H. Barella, L. P. F. Garcia, M. P. de Souto, A. C. Lorena, A. de Carvalho, Data Complexity Measures for Imbalanced Datasets, Expert Systems with Applications, sv. 106, str. (101-113), 2018.

[33] J. Błaszczyński, J. Stefanowski, Local Data Characteristics in Learning Classifiers from Imbalanced Data, Proceedings of the 14th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, str. (36-45), 2018.

[34] S. K. Pandey, R. B. Mishra, A. K. Tripathi, Machine learning based methods for software fault prediction: A survey, Information Sciences, br.5, sv. 546, str. (440-458), 2021.

[35] C. Jin, Cross-project software defect prediction based on domain adaptation learning and optimization, Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering, str. (453-462), 2021.

[36] M. Jureczko, L. Madeyski, Towards identifying software project clusters with regard to defect prediction, u: Proceedings of the 6th International Conference on Predictive Models in Software Engineering (Promise'10), str. (1-10), 2010.

[37] Z. He, J. Zhang, An empirical study on the effectiveness of features selection in software defect prediction, u: Proceedings of the 24th Asia-Pacific Software Engineering Conference (APSEC'17), str. (227-236), 2017.

[38] T. M. Khoshgoftaar, et al., Exploring the impact of feature selection on software defect prediction, Information Sciences, br.5, sv. 432, str. (119-136), 2018.

[39] T. G. Dietterich, Approximate statistical tests for comparing supervised classification learning algorithms, Neural Computation, br. 7,sv. 10, str. (1895-1923), 1998.

SAŽETAK

Cilj ovog diplomskog rada je istražiti problem predviđanja pogrešaka u programskoj podršci koristeći različite metode strojnog učenja. Glavni izazov bio je neuravnoteženost klasa unutar korištenih skupova podataka, što je negativno utjecalo na učinkovitost klasifikacijskih modela. U cilju poboljšanja performansi modela, primijenjene su tehnike odabira značajki (Pearsonov koeficijent korelacije, zajednička informacija, ANOVA F-test i slijedna pretraga unaprijed) te metode uzorkovanja (nasumično preuzorkovanje, nasumično poduzorkovanje i SMOTE). Najbolje rezultate postigla je kombinacija NB s tehnikom odabira značajki pomoću omotača slijedne pretrage unaprijed, kao i DT u kombinaciji s nasumičnim preuzorkovanjem. Unatoč tome, pokazalo se da ne postoji univerzalna metoda koja donosi optimalne rezultate za sve skupove podataka, već je potrebno prilagoditi pristup ovisno o specifičnostima problema.

Ključne riječi: klasifikacija, metode uzorkovanja, neuravnoteženost, odabir značajki, predviđanje pogrešaka.

ABSTRACT

Software defect prediction using machine learning

The aim of this thesis is to explore the problem of software defect prediction using various machine learning methods. The main challenge was the class imbalance within the datasets used, which negatively impacted the performance of classification models To improve model performance, feature selection techniques (Pearson correlation coefficient, mutual information, ANOVA F-test, and sequential forward selection) and sampling methods (random oversampling, random undersampling, and SMOTE) were applied. The best results were achieved by combining Naive Bayes with the sequential forward selection wrapper feature selection method, as well as decision trees combined with random oversampling. However, it was found that no universal method guarantees optimal results for all datasets, and the approach must be adapted based on the specific characteristics of the problem.

Keywords: class imbalance, defect prediction, feature selection, machine learning, sampling methods.

PRILOG

P.4.1. Mjere za računanje stupnja unutarnjih karakteristika skupova podataka

Maksimalni Fisherov omjer diskriminacije (F1) mjeri sposobnost značajki da razlikuju klase. Ova mjera računa se na temelju srednjih vrijednosti i varijanci dviju klasa. Za svaku značajku f_i, Fisherov omjer diskriminacije definiran je kao:

$$\mathrm{f} = \frac{(\mu 1 - \mu 2)^2}{\sigma_1^2 + \sigma_2^2},$$

gdje je f definirano za jednu dimenziju značajke, a $\mu 1$, $\mu 2$, σ_1^2 i σ_2^2 su srednje vrijednosti i varijance dviju klasa. Nije nužno da svaka značajka u višedimenzionalnom problemu pridonosi razdvajanju klasa. Problem je jednostavan sve dok postoji barem jedna razlikovna značajka. Stoga se za opis problema koristi najveća vrijednost f među svim dimenzijama značajki (m označava broj značajki):

$$F1 = max\{f_1, f_2, \ldots, f_m\}.$$

Volumen preklapajuće regije (F2) koristi se za mjerenje stupnja preklapanje distribucija vrijednosti značajki unutar klasa. F2 se može odrediti pronalaskom minimalnih i maksimalnih vrijednosti svake značajke f_i za obje klase (*c1* i *c2*). Volumen preklapajuće regije tada se izračunava na sljedeći način:

$$F2 = \prod_{i=1}^{m} \frac{\min(max(f_i, c_1), max(f_i, c_2)) - max(min(f_i, c_1), min(f_i, c_2))}{\max(max(f_i, c_1), max(f_i, c_2)) - max(min(f_i, c_1), min(f_i, c_2))}.$$

Maksimalna efikasnost pojedinačne značajke (F3) **se** određuje omjerom instanci izvan preklapajuće regije i ukupnog broja instanci. Za svaku značajku f_i , preklapajuća regija uključuje instance iz obje klase, dok nepreklapajuća regija sadrži samo instance iz jedne klase. Efikasnost svake značajke određuje se omjerom nepreklapajuće regije i ukupnog broja instanci. F3 se definira kao maksimalna pojedinačna efikasnost svih značajki:

$$F3 = \underset{i=1}{\overset{m}{max}} \frac{n - n_o(f_i)}{n},$$

gdje $no(f_i)$ označava preklapajuće regije za značajku f_i. Veća vrijednost ukazuje na manju složenost, a vrijednosti se kreću od nula do jedan.

Omjer udaljenosti najbližih susjeda unutar i izvan klase (N2) mjeri složenost problema kroz usporedbu udaljenosti između instanci unutar iste klase i instanci koje pripadaju različitim klasama. N2 se računa kao

$$N2 = \frac{\sum_{i=1}^{n} d(x_i, NN(x_i) \in y_i)}{\sum_{i=1}^{n} d(x_i, NN(x_i) \in y_j \neq y_i)},$$

gdje NN označava najbližeg susjeda. U prvom dijelu formule $NN(x_i) \in y_i$ označava najbližeg susjeda unutar iste klase y_i, dok $NN(x_i) \in y_j \neq y_i$ predstavlja najbližeg susjeda iz različite klase y_i. Niže vrijednosti N2 ukazuju na jednostavnije probleme, gdje su instance općenito bliže instancama vlastite klase nego instancama iz druge klase.

Stopa pogreške klasifikatora najbližeg susjeda (N3) mjeri pogreške koje klasifikator najbližeg susjeda pravi prilikom klasifikacije instanci. Ova se stopa računa korištenjem metode isključenja jednog uzorka (engl. *leave-one-out cross-validation*), pri čemu se mjeri koliko instanci je pogrešno klasificirano. Vrijednosti ove mjere kreću se od nula do jedan, pri čemu niže vrijednosti ukazuju na bolju klasifikacijsku točnost.

Prosječna veličina lokalnog skupa (LSCAvg) koristi se za procjenu blizine instanci prema granici odluke. Definira se kao

$$LS(x_i) = \{x_i | \mathrm{d}(x_i, x_j < d(x_i, ne(x_i))\},$$

gdje $ne(x_i)$ označava najbližu instancu koja pripada drugoj klasi. Veličina lokalnog skupa instance označava njezinu blizinu granici odluke i razmak između klasa. Prosječni veličina lokalnog skupa (LSCAvg) računa se na sljedeći način:

$$\mathrm{LSCAvg} = \frac{1}{n^2} \sum_{i=1}^{n} |\mathrm{LS}(x_i)|.$$

Niže vrijednosti LSCAvg ukazuju na složeniji skup podataka.