Klasifikacija žanrova knjiga primjenom tehnika obrade prirodnog jezika

Herceg, Ivana

Master's thesis / Diplomski rad

2024

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Split, Faculty of Science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:166:255935

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-12-29**

Repository / Repozitorij:

Repository of Faculty of Science

zir.nsk.hr

UNIVERSITY OF SPLIT

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO MATEMATIČKI FAKULTET

DIPLOMSKI RAD

KLASIFIKACIJA ŽANROVA KNJIGA PRIMJENOM TEHNIKA OBRADE PRIRODNOG JEZIKA

Ivana Herceg

Split, srpanj 2024.

Temeljna dokumentacijska kartica

Diplomski rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

KLASIFIKACIJA ŽANROVA KNJIGA PRIMJENOM TEHNIKA OBRADE PRIRODNOG JEZIKA

Ivana Herceg

SAŽETAK

Istražena je klasifikacija žanrova knjiga koristeći tehnike obrade prirodnog jezika. Cilj je bio razviti modele koji precizno predviđaju žanrove knjiga na temelju tekstualnih opisa, primjenom metoda poput TF-IDF-a, Word2Vec-a i algoritama kao što su logistička regresija i slučajna šuma. Rezultati pokazuju da kombinacija naprednih vektorizacija i klasifikacijskih algoritama pruža visoku točnost u klasifikaciji višestrukih žanrova. Identificirani su izazovi vezani uz neuravnoteženost podataka te su predložene smjernice za daljnja istraživanja.

Ključne riječi: Obrada prirodnog jezika, TF-IDF, Word2Vec, Doc2Vec, višestruka klasifikacija oznaka, logistička regresija, slučajna šuma, klasifikator s glasanjem

Rad sadrži: 78 stranice, 6 grafičkih prikaza, 24 tablice i 39 literaturnih navoda. Izvornik je na hrvatskom jeziku

Mentor: Prof. dr. sc. Branko Žitko, izvanredni profesor Prirodoslovno-matematičkog fakulteta, Sveučilište u Splitu

Ocjenjivači: Prof. dr. sc. Branko Žitko, redoviti profesor Prirodoslovno-matematičkog fakulteta, Sveučilište u Splitu

Prof. dr. sc. Ani Grubišić, redoviti profesor Prirodoslovno-matematičkog fakulteta, Sveučilište u Splitu

Lucija Bročić, asistentica

Rad prihvaćen: Rujan, 2024.

Basic documention card

Thesis

University of Split
Faculty of Science
Department of Informatics
Ruđera Boškovića 33, 21000 Split, Croatia

CLASSIFICATION OF BOOK GENRES USING NATURAL LANGUAGE PROCESSING TECHNIQUES

Ivana Herceg

ABSTRACT

The classification of book genres was explored using natural language processing techniques. The goal was to develop models that accurately predict book genres based on textual descriptions, employing methods such as TF-IDF, Word2Vec, and algorithms like logistic regression and Random Forest. The results indicate that the combination of advanced vectorization techniques and classification algorithms provides high accuracy in multi-label genre classification. Challenges related to data imbalance were identified, and guidelines for further research were proposed.

Keywords:	Natural Language Processing, TF-IDF, Word2Vec, Doc2Vec, multi-label classification, logistic regression, Random Forest, Voting Classifier
Thesis consists of :	78 pages, 6 figures, 24 tables, and 39 references. Original language: Croatian.
Mentor:	Prof. Branko Žitko, Ph.D. Full Professor University of Split, Faculty of Science
Reviewers:	Prof. Branko Žitko, Ph.D. Full Professor University of Split, Faculty of Science Prof. Ani Grubišić, Ph.D. Full Professor University of Split, Faculty of Science Lucija Bročić, Assistant

Thesis accepted: September, 2024

SADRŽAJ

1 Uvod

U posljednjih nekoliko godina, obrada prirodnog jezika (engl. NLP - Natural Language Processing) postala je jedno od najvažnijih područja unutar umjetne inteligencije. S obzirom na sve veću količinu tekstualnih podataka dostupnih online, potreba za automatiziranim sustavima koji mogu razumjeti, obraditi i klasificirati tekst postala je ključna. Jedan od izazova u NLP-u je klasifikacija tekstualnih dokumenata u više kategorija, posebno kada je riječ o problemima višestruke oznake, gdje svaki dokument može pripadati više od jedne klase.

U ovom diplomskom radu istražuje se problem klasifikacije žanrova knjiga na temelju njihovih opisa, koristeći napredne NLP tehnike i modele strojnog učenja. Primarni cilj rada je razviti i evaluirati modele koji mogu točno predvidjeti žanrove knjiga na temelju tekstualnih opisa, koristeći različite metode vektorizacije (TF-IDF, Word2Vec, Doc2Vec) i algoritme klasifikacije (logička regresija, slučajna šuma).

Ovaj rad će obuhvatiti sve faze procesa klasifikacije, uključujući prikupljanje i pretprocesiranje podataka, treniranje i evaluaciju modela, kao i analizu rezultata te prijedloge za daljnja istraživanja. Na taj način, doprinosi razumijevanju učinkovitosti različitih metoda u višestrukoj klasifikaciji oznaka u domeni književnih žanrova.

U drugom poglavlju daje se pregled ključnih istraživanja i tehnika iz područja obrade prirodnog jezika, s posebnim naglaskom na metode vektorizacije kao što su TF-IDF, Word2Vec i Doc2Vec, te na primjenu naprednih modela poput BERT-a. Također su analizirane različite metode klasifikacije teksta. Sljedeće poglavlje opisuje korištene alate i metode, uključujući različite pristupe vektorizaciji teksta i algoritme strojnog učenja. Opisane su metode poput TF-IDF-a i Word2Vec-a te algoritmi kao što su logička regresija, slučajna šuma i klasifikator s glasanjem. U četvrtom poglavlju opisan je skup podataka koji će biti korišten za klasifikaciju žanrova knjiga, a u petom poglavlju detaljno je prikazan proces implementacije modela, od pretprocesiranja podataka do vektorizacije tekstualnih opisa i treniranja modela, uz poseban fokus na optimizaciju i balansiranje podataka kako bi se postigla što veća točnost klasifikacije. Šesto poglavlje bavi se evaluacijom modela, prikazujući rezultate analiza uz korištenje standardnih metrika poput točnosti, preciznosti i F1 mjere. Na kraju se razmatraju izazovi poput neuravnoteženosti podataka i složenosti višestrukih oznaka. Zaključno poglavlje donosi pregled ključnih rezultata rada, ističe prednosti korištenih metoda i daje prijedloge za buduća istraživanja s ciljem poboljšanja performansi modela u klasifikaciji višestrukih oznaka.

2 Pregled literature

U ovom poglavlju daje se pregled ključnih istraživanja i tehnika iz područja obrade prirodnog jezika te klasifikacije teksta. Poseban naglasak stavljen je na metode vektorizacije, poput TF-IDF-a, Word2Vec-a i Doc2Vec-a, koje se koriste za pretvaranje teksta u numeričke prikaze. Također se obrađuju napredniji modeli, poput BERT-a, koji omogućuju dublje razumijevanje konteksta teksta te se koriste u zadacima višestruke klasifikacije oznaka. Pregled literature pruža uvid u dosadašnja istraživanja koja su temelj za implementaciju modela u ovom radu.

2.1 Pregled istraživanja u području obrade prirodnog jezika

Obrada prirodnog jezika ključna je tehnologija koja omogućuje računalima razumijevanje i generiranje ljudskog jezika, s brojnim primjenama u klasifikaciji tekstualnih podataka. Natural Language Toolkit (NLTK) jedna je od najpopularnijih i najkorištenijih biblioteka za obradu prirodnog jezika. Ova biblioteka nudi širok raspon alata i resursa za rad s tekstom, uključujući preprocesiranje podataka, tokenizaciju, lematizaciju i vektorizaciju. NLTK je posebno korisna za primjenu tradicionalnih metoda obrade jezika, poput TF-IDF (Term Frequency-Inverse Document Frequency), koja analizira učestalost pojavljivanja riječi u dokumentima. Iako je TF-IDF korisna za jednostavne zadatke klasifikacije, nedostatak je što ne može prepoznati semantičke veze među riječima, ograničavajući njezinu primjenu u složenijim zadacima [1].

Kako bi se prevladala ta ograničenja, razvijeni su napredniji modeli poput Word2Vec i Doc2Vec. Word2Vec pretvara riječi u vektorske reprezentacije koristeći kontekstualne informacije kako bi slične riječi bile smještene blizu u višedimenzionalnom prostoru. Ovaj pristup koristi se u višestrukoj klasifikaciji oznaka, omogućujući bolju semantičku obradu tekstova. Primjerice, Word2Vec model uspješno primjenjuje ove tehnike u zadacima klasifikacije, gdje se tekstovi kategoriziraju prema višestrukim oznakama, pokazujući bolje rezultate u odnosu na tradicionalne metode [2]. Doc2Vec proširuje ovu funkcionalnost, omogućujući modeliranje cjelokupnih dokumenata, što je korisno u zadacima koji uključuju duže tekstove, poput knjiga ili članaka [3].

S pojavom modela BERT (Bidirectional Encoder Representations from Transformers), NLP zadaci su dodatno poboljšani. BERT koristi dvosmjerni transformator koji analizira tekst u oba smjera, uzimajući u obzir kontekst riječi prije i poslije ciljne riječi, što omogućuje bolje razumijevanje složenih odnosa u tekstu. Ova tehnologija pokazuje značajna poboljšanja u

višestrukoj klasifikaciji oznaka, osobito u složenim tekstovima poput pravnih dokumenata i sinopsisa filmova [4].

Radovi koji se bave višestrukom klasifikacijom oznaka, kao što je predikcija filmskih žanrova pomoću NLP-a, koristeći kombinaciju Word2Vec-a i BERT-a, pokazuju da kombinacija ovih tehnologija rezultira visokom točnošću i preciznošću klasifikacije. Modeli kao što su slučajna šuma i BERT korišteni su kako bi se poboljšala klasifikacija filmova prema žanrovima, pri čemu je višestruka oznaka omogućila bolju kategorizaciju filmova koji pripadaju različitim žanrovima istovremeno [5]. Dodatna istraživanja pokazuju da kombinacija tradicionalnih metoda kao što je TF-IDF s modernim modelima kao što su Word2Vec i BERT može značajno poboljšati klasifikaciju tekstova u više kategorija [6].

Slika 1 Evolucija modela za obradu prirodnog jezika (NLP) [7]

Prikazana Slika 1 ilustrira napredak u metodama obrade prirodnog jezika, prikazujući kako su se modeli razvijali od jednostavnih statističkih metoda do složenih algoritama dubokog učenja. Napredak prikazan na ovoj vremenskoj traci omogućio je sve veću preciznost i učinkovitost u zadacima obrade teksta, kao što su klasifikacija, prepoznavanje entiteta, i generiranje jezika.

2.2 Pregled tehnika klasifikacije teksta

Klasifikacija teksta obuhvaća brojne tehnike koje se mogu podijeliti na tradicionalne i moderne pristupe. Ovisno o prirodi podataka i zadatku, biraju se različiti modeli.

Tradicionalne metode, poput naivni Bayes (engl. *Naive Bayes)*, metoda potpornih vektora (engl. *Support Vector Machines* - SVM) i logička regresija (engl. *Logistic Regression)*, oslanjaju se na ručno kreirane značajke i koriste matematičke pristupe za kategorizaciju teksta. naivni Bayes se često koristi u jednostavnijim zadacima klasifikacije zbog svoje efikasnosti kod malih skupova podataka i sposobnosti da dobro funkcionira s kategoriziranim podacima [8]. SVM je koristan za binarne klasifikacije jer maksimizira marginu između klasa, dok

logička regresija djeluje dobro za jednostavne zadatke klasifikacije s jasnim razdvajanjem klasa [9].

Ove metode često koriste TF-IDF kao tehniku za vektorizaciju teksta, koja pretvara tekst u numeričke značajke na temelju učestalosti pojavljivanja riječi. Iako je TF-IDF jednostavan i efikasan, ne uzima u obzir semantičke odnose među riječima, što je ograničenje u složenijim zadacima poput višestruke klasifikacije oznaka [10].

Napredniji pristupi koriste duboke neuronske mreže koje su posebno učinkovite u prepoznavanju složenih obrazaca unutar tekstualnih podataka. RNN-ovi (Rekurentne neuronske mreže) i LSTM-ovi (*Long Short-Term Memory* mreže) često se koriste za sekvencijalne podatke, poput rečenica ili paragrafa. RNN-ovi su korisni kada je potrebno analizirati podatke koji ovise o prethodnim elementima u sekvenci, kao što su rečenice ili članci [4].

CNN-ovi (Konvolucijske neuronske mreže), iako tradicionalno korišteni u obradi slika, pokazali su se vrlo učinkovitima i u obradi tekstualnih podataka jer mogu prepoznati ključne značajke unutar tekstualnih nizova. CNN-ovi su često primjenjivi za klasifikaciju kratkih tekstova ili kodiranje rečenica u vektore fiksne duljine [5].

Jedna od najnovijih i najmoćnijih tehnika klasifikacije teksta temelji se na transformerima, posebice modelu BERT (*Bidirectional Encoder Representations from Transformers*). Za razliku od RNN-ova, koji procesuiraju tekst redoslijedno, transformeri omogućuju paralelnu obradu riječi unutar rečenice, čime se poboljšava brzina i točnost analize. BERT koristi dvosmjerno kodiranje konteksta, što znači da uzima u obzir riječi prije i nakon ciljne riječi, omogućujući preciznije razumijevanje značenja riječi u kontekstu [2]. Ova tehnika je posebno učinkovita u višestrukoj klasifikaciji oznaka jer može simultano analizirati različite odnose unutar teksta.

Hibridni modeli kombiniraju prednosti tradicionalnih metoda s modernim tehnikama dubokog učenja. U nekim aplikacijama, kao što je predikcija filmskih žanrova, kombinacija metoda poput TF-IDF za vektorizaciju i dubokih neuronskih mreža, kao što su Word2Vec i BERT, daje bolje rezultate. Ovi hibridni pristupi omogućuju modelima da iskoriste prednosti različitih tehnika za poboljšanje točnosti klasifikacije [1].

Izbor tehnike klasifikacije teksta ovisi o prirodi zadatka:

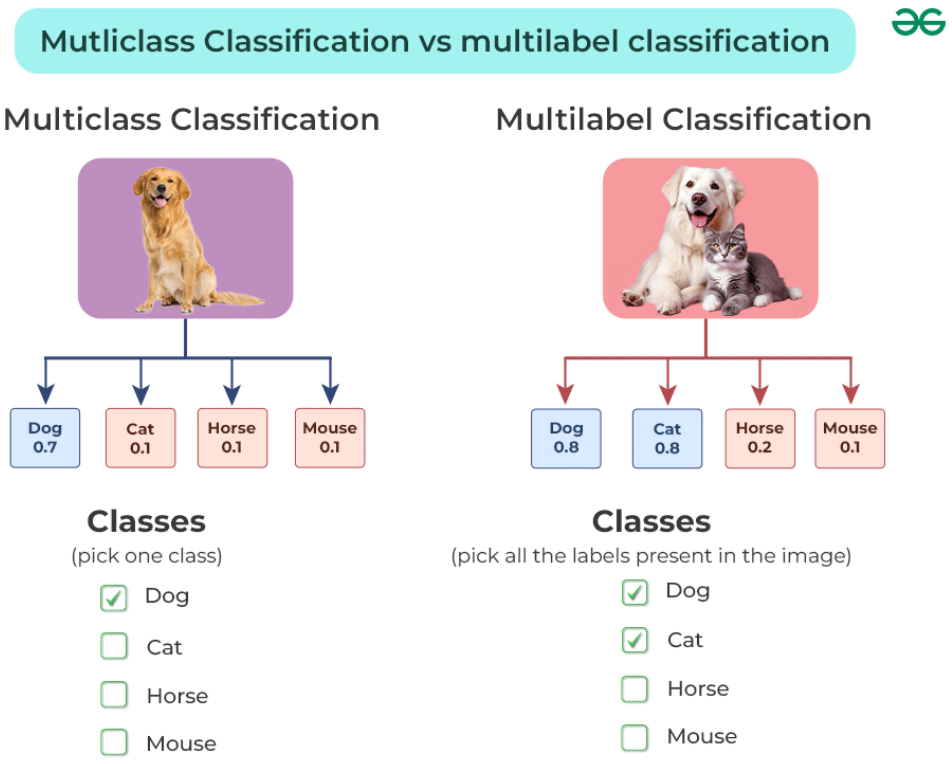
- Za jednostavne binarne klasifikacije često se koriste tradicionalne metode poput naivnog Bayesa i logičke regresije.
- Za zadatke koji zahtijevaju duboko razumijevanje konteksta i analizu sekvenci, RNN i LSTM su najbolji izbor.
- Kod višestruke klasifikacije oznaka i složenih zadataka koji zahtijevaju prepoznavanje višestrukih odnosa, transformeri poput BERT-a pokazuju izvanredne rezultate.

Klasifikacija teksta evoluirala je od jednostavnih metoda ručne izrade značajki do složenih modela dubokog učenja, omogućujući bolju analizu i kategorizaciju velikih količina podataka.

2.3 Metoda višestruke klasifikacije oznaka

Višestruka klasifikacija oznaka (Multi-Label Classification) predstavlja klasifikacijski problem u kojem jedan primjer može pripadati više od jedne klase ili kategorije. To je u suprotnosti s višeklasnom klasifikacijom (Multiclass Classification), gdje se svakom primjeru može dodijeliti samo jedna klasa iz skupa mogućih klasa. Višestruka klasifikacija oznaka je posebno važna u situacijama gdje podaci imaju složene karakteristike i mogu istovremeno pripadati više kategorija.

U višeklasnoj klasifikaciji svaki primjer može biti klasificiran u samo jednu klasu. Na primjer, razmotrite knjigu koja može biti označena kao "Fantazija", "Romansa", "Avantura" ili "Horor", ali ne može biti označena s više od jedne od tih klasa. U višestrukoj klasifikaciji oznaka, isti primjer može biti označen s više klasa, što znači da knjiga može istovremeno pripadati žanrovima "Fantazija" i "Avantura".



Slika 2 Razlika između klasifikacije s više klasa i klasifikacije s više oznaka [11]

Slika 2 prikazuje ključnu razliku između klasifikacije s više klasa i klasifikacije s više oznaka. U lijevom dijelu slike, koji se odnosi na klasifikaciju s više klasa, prikazana je slika psa s četiri moguće klase: pas, mačka, konj i miš. Ovdje model može dodijeliti samo jednu klasu svakom ulaznom podatku; u ovom primjeru, pas je klasificiran s najvećom vjerojatnošću od 70%, dok su ostale klase dobile niže vjerojatnosti. Ova vrsta klasifikacije koristi se kada svaki uzorak može pripadati samo jednoj klasi, čak i ako postoji više mogućih klasa. Nasuprot tome, desni dio slike prikazuje klasifikaciju s više oznaka, gdje je prikazana slika psa i mačke zajedno, s istim mogućim klasama. U ovom slučaju, model može dodijeliti više oznaka jednom ulaznom podatku; primjerice, model je dodijelio oznake "pas" i "mačka", svaku s vjerojatnošću od 80%. Ova vrsta klasifikacije koristi se kada jedan uzorak može istovremeno pripadati više klasa, omogućujući modelu da dodijeli sve relevantne oznake za taj uzorak.

Prednosti višestruke klasifikacije oznaka su:

- Fleksibilnost: Višestruka klasifikacija oznaka omogućuje modelima da klasificiraju podatke u više kategorija istovremeno, što je posebno korisno u aplikacijama gdje se

podaci ne mogu svrstati u jednu jedinu klasu. Na primjer, u klasifikaciji filmskih sinopsisa, jedan film može biti istovremeno označen kao akcija, triler i drama [8].

- Korištenje veza među oznakama: Višestruka klasifikacija oznaka omogućuje prepoznavanje odnosa među oznakama. Na primjer, žanrovi "romantika" i "drama" često su povezani, pa model može prepoznati ove odnose i učinkovitije klasificirati podatke [9].
- Bolje performanse: Korištenje naprednih tehnika, poput *Word2Vec* i *BERT*-a, omogućuje modelima da prepoznaju složene obrasce i semantičke veze unutar teksta, što rezultira boljim performansama u zadacima s višestrukim oznakama [10]. Primjer iz istraživanja o klasifikaciji filmskih žanrova pokazuje da kombinacija transformera i tradicionalnih metoda donosi visoku točnost klasifikacije.

Nedostaci višestruke klasifikacije oznaka su:

- Složenost modela: Višestruka klasifikacija oznaka uvodi dodatnu složenost prilikom oblikovanja modela, jer model mora simultano klasificirati podatke u više kategorija. To zahtijeva više resursa za treniranje modela i povećava rizik od prekomjerne prilagodbe (engl. *overfitting*) [12].
- Neravnoteža podataka: U višestrukoj klasifikaciji oznaka često dolazi do problema s neravnotežom podataka, gdje neke oznake imaju mnogo više primjera nego druge. To može rezultirati time da model favorizira češće oznake, zanemarujući manje zastupljene [2].
- Kompleksnost vrednovanja: Vrednovanje modela višestruke klasifikacije oznaka može biti složenije od jedne oznake. Standardne metrike kao što su točnost i preciznost moraju se prilagoditi za višestruku oznaku, često korištenjem mjera poput mikro i makro prosjeka [1].

Iako višestruka klasifikacija oznaka nudi fleksibilnost i mogućnost složenih klasifikacija, suočava se s izazovima kao što su složenost modela i neravnoteža podataka. Unatoč tome, primjena naprednih modela, poput *BERT*-a i *Word2Vec*-a, omogućava značajna poboljšanja u mnogim domenama, čineći višestruku klasifikaciju oznaka ključnom tehnikom u modernim NLP aplikacijama.

3 Opis korištenih alata i metoda

U ovom poglavlju opisane su ključne metode i alati korišteni za klasifikaciju žanrova knjiga na temelju tekstualnih opisa. Prvo su prikazane metode vektorizacije teksta, poput TF-IDF-a, Word2Vec-a i Doc2Vec-a, koje pretvaraju tekstualne podatke u numeričke reprezentacije. Zatim su objašnjeni algoritmi strojnog učenja, uključujući logističku regresiju, slučajnu šumu i klasifikator s glasanjem, koji su korišteni za treniranje modela klasifikacije. Posebna pažnja posvećena je balansiranju podataka kako bi se poboljšala točnost modela u kontekstu višestrukih oznaka.

3.1 Vektorizacija teksta

Vektorizacija teksta predstavlja osnovni postupak u obradi prirodnog jezika (NLP) kojim se tekst pretvara u numeričke vektore kako bi postao razumljiv algoritmima strojnog učenja. Ovaj proces je ključan za razne NLP zadatke poput klasifikacije teksta, pretraživanja informacija, analize sentimenta i mnogih drugih aplikacija. Klasične metode vektorizacije uključuju *CountVectorizer*, koji pretvara tekst u rijetki vektor temeljen na učestalosti riječi unutar dokumenta. Ova metoda zanemaruje redoslijed riječi i kontekst, ali je jednostavna i učinkovita za osnovne zadatke poput klasifikacije i rudarenja podataka [13].

Međutim, CountVectorizer ima svoje nedostatke jer ne uzima u obzir važnost pojedinih riječi unutar cijelog korpusa. Ovaj problem riješen je primjenom TF-IDF (*Term Frequency-Inverse Document Frequency*) metode, koja vektorizira tekst tako da dodjeljuje težine riječima u odnosu na njihovu frekvenciju u dokumentu, ali i u cijelom korpusu. Na taj način, TF-IDF smanjuje važnost često korištenih riječi (poput veznika i prijedloga) te omogućuje boljitak modelima pri identificiranju ključnih termina unutar dokumenata [14].

Naprednije metode vektorizacije, poput *Word2Vec* i *Doc2Vec*, značajno su unaprijedile ovu domenu omogućujući stvaranje gustih vektora riječi i dokumenata, koji zadržavaju semantičke odnose među riječima. Word2Vec, posebno, koristi neuronske mreže za kreiranje vektorskih prikaza riječi, gdje svaka riječ u prostoru ima svoju poziciju temeljenu na kontekstu u kojem se pojavljuje. Ova tehnika omogućuje modelima da prepoznaju i iskorištavaju semantičke sličnosti između riječi, čime se postiže dublje razumijevanje značenja. Primjerice, vektori riječi "kralj" i "kraljica" bit će blizu jedan drugome u vektorskom prostoru, reflektirajući njihov sličan semantički značaj [15]. S druge strane, Doc2Vec širi ovu ideju na cijele dokumente,

omogućujući reprezentaciju većih tekstualnih jedinica, kao što su cijeli tekstovi knjiga ili članci.

Još jedan važan aspekt vektorizacije su vektorske baze podataka, koje omogućuju pohranu i pretragu tekstualnih podataka prema njihovim vektorskim prikazima. Ove baze podataka koriste se za semantičku pretragu, što omogućuje pronalazak dokumenata na temelju značenja, a ne samo podudaranja ključnih riječi. Takav pristup postaje sve važniji u aplikacijama poput *chatbot*-ova i sustava za pretraživanje informacija, gdje se očekuje da sustavi razumiju kontekst i pružaju relevantne odgovore temeljene na značenju korisničkih upita.

Osim ovih metoda, vektorizacija koristi i modele temeljene na transformerima poput *BERT*-a, koji koristi dvosmjeran kontekstualni prikaz riječi. BERT omogućuje preciznije razumijevanje značenja riječi unutar njihovog specifičnog konteksta u rečenici, čime se značajno poboljšavaju performanse u složenim NLP zadacima poput razumijevanja jezika i prevođenja [2].

Vektorizacija teksta, dakle, nije samo proces pretvaranja riječi u brojeve, već i ključni alat za modeliranje i razumijevanje prirodnog jezika. Razvoj ovih tehnika omogućio je značajan napredak u sposobnosti strojeva da interpretiraju i obrađuju ljudski jezik, otvarajući nove mogućnosti za primjenu u širokom spektru industrija.

3.1.1 Vektorizacija prebrojavanja

Vektorizacija prebrojavanja (engl. CountVectorizer) je jednostavna, ali moćna tehnika za pretvaranje tekstualnih podataka u numeričke značajke temeljem broja pojavljivanja riječi u tekstu. Ova tehnika je osnovni alat u obradi prirodnog jezika i koristi se za pripremu tekstualnih podataka za modele strojnog učenja. CountVectorizer stvara matricu gdje su redci dokumenti, a stupci jedinstvene riječi iz skupa dokumenata, dok vrijednosti unutar matrice predstavljaju broj pojavljivanja svake riječi u pojedinom dokumentu.

Glavna funkcija CountVectorizera je vreća riječi (engl. BOW bag-of-words) pristup koji zanemaruje redoslijed riječi i fokusira se isključivo na njihovu učestalost. Kada CountVectorizer obradi skup dokumenata, kreira rječnik jedinstvenih riječi koje se pojavljuju u korpusu, a zatim broji koliko se puta svaka riječ pojavljuje u svakom dokumentu [16]. Ova tehnika omogućava da tekstualni podaci budu pretvoreni u numeričke vektore koje modeli strojnog učenja mogu koristiti za daljnju analizu.

Na primjer, ako imamo dva dokumenta:

- "The cat sat on the mat."
- "The dog barked at the cat."

Nakon primjene CountVectorizera, rječnik može izgledati ovako: ['barked', 'cat', 'dog', 'mat', 'sat', 'the'], a rezultirajuća matrica bi se mogla prikazati kako je prikazano u *Tablici 1*.

	barked	cat	dog	mat	sat	the
Dokument 1	0	1	0	1	1	2
Dokument 2	1	1	1	0	0	2

Tablica 1. Rječnik nakon primjene CountVectorizer-a

U *Tablici 1* svaki redak predstavlja dokument, a svaki stupac frekvenciju pojavljivanja određene riječi u tom dokumentu. Ovaj jednostavan pristup omogućuje brzu i efikasnu ekstrakciju značajki iz teksta za korištenje u modelima strojnog učenja.

CountVectorizer nudi mnoge opcije za prilagodbu, omogućujući korisnicima da kontroliraju način na koji će se vektori kreirati. Primjerice, korisnici mogu postaviti maksimalan broj značajki (max_features), definirati minimalnu učestalost riječi (min_df), ili isključiti uobičajene riječi (tzv. stop words) koje nemaju značajnu informativnu vrijednost, poput riječi "the" i "and". Korištenje tih parametara omogućava učinkovito rukovanje velikim skupovima podataka i smanjenje dimenzionalnosti.

IBM dokumentacija pokazuje kako se CountVectorizer može prilagoditi za uklanjanje uobičajenih riječi (stop_words='english') i ograničavanje broja značajki (max_features=5000). Time se fokus stavlja na najrelevantnije riječi u skupu podataka, što može biti ključno kod analize velikih korpusa [17].

CountVectorizer podržava i rad s n-gramima, što znači da korisnici mogu ne samo brojati pojavljivanja pojedinačnih riječi, nego i kombinacija riječi (npr. bigrami ili trigrami). Ovaj pristup je koristan kada je redoslijed riječi važan, kao što je slučaj u analizi osjećaja ili detekciji fraza. Na primjer, umjesto da se broji koliko puta se pojedinačne riječi pojavljuju, CountVectorizer može brojati pojavljivanja fraza poput "good movie" ili "not bad", što je posebno korisno kod analize sentimenta [16].

Iako je CountVectorizer koristan za mnoge zadatke, ima i svoja ograničenja. Jedno od najvećih ograničenja je to što ne uzima u obzir kontekst u kojem se riječi pojavljuju. Na primjer, riječi poput "bank" može značiti financijsku instituciju (banka) ili obalu rijeke, ovisno o kontekstu u

kojem se koristi mogu imati različita značenja, ali CountVectorizer ih tretira kao isti entitet. Osim toga, CountVectorizer može generirati vrlo velike i rijetke matrice kada se koristi s velikim skupovima podataka, što može otežati daljnju obradu i povećati računalne resurse potrebne za modeliranje [18].

CountVectorizer se često koristi kao prvi korak u pripremi podataka za modele strojnog učenja, osobito u fazama ekstrakcije značajki. Na primjer, koristi se za prevođenje tekstualnih podataka u značajke koje se mogu koristiti za klasifikaciju, kao što su zadaci kategorizacije dokumenata ili analiza osjećaja. Kombinira se s tehnikama poput TF-IDF-a kako bi se dodatno poboljšala točnost modela i eliminirao utjecaj često korištenih riječi koje nisu informativne [19].

3.1.2 TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) je ključna tehnika u obradi prirodnog jezika koja pomaže u određivanju važnosti riječi u dokumentu unutar kolekcije dokumenata. Ova metoda kombinira dvije komponente: frekvenciju pojavljivanja riječi u dokumentu i inverznu frekvenciju pojavljivanja te riječi u skupu dokumenata kako bi izračunala težinu svake riječi, ističući riječi koje su značajne za određeni dokument, ali nisu uobičajene u drugim dokumentima [20].

Term Frequency (TF) odnosi se na broj pojavljivanja riječi unutar jednog dokumenta. Frekvencija riječi računa se pomoću formule:

$$TF(t,d) = \frac{\text{Broj pojavljivanja riječi } t \text{ u dokumentu } d}{\text{Ukupan broj riječi u dokumentu } d}$$

Međutim, frekvencija nije uvijek idealna jer može prenaglasiti uobičajene riječi. Stoga se često koristi logaritamska skala kako bi se umanjio utjecaj često ponavljanih riječi, što rezultira preciznijim vrednovanjem važnosti riječi unutar dokumenta [2].

Inverzna frekvencija dokumenata (IDF) procjenjuje koliko je riječ rijetka ili specifična za određeni dokument u odnosu na cijelu kolekciju dokumenata. Računa se pomoću formule:

$$IDF(t,D) = \log\left(\frac{\text{Ukupan broj dokumenata u korpusu } N}{\text{Broj dokumenata koji sadrže riječ } t}\right)$$

Riječi koje se pojavljuju u manjem broju dokumenata dobivaju višu vrijednost IDF-a jer su one značajnije za razlikovanje tih dokumenata od ostalih [20].

Na primjer, ako imamo dva dokumenta:

- "The cat sat on the mat."
- "The dog barked at the cat."

Nakon primjene TF-IDF vektorizacije, rječnik može izgledati ovako: ['barked', 'cat', 'dog', 'mat', 'sat', 'the'], a rezultirajuće TF-IDF vrijednosti mogu se prikazati u Tablici 2.

	barked	cat	dog	mat	sat	the
Dokument 1	0.000	0.430	0.000	0.430	0.430	0.561
Dokument 2	0.523	0.311	0.523	0.000	0.000	0.627

Tablica 2 Rječnik i TF-IDF vrijednosti nakon primjene TF-IDF-a

U Tablici 2 TF-IDF daje različite težine riječima na temelju njihove učestalosti u svakom dokumentu, ali i rijetkosti u cijelom skupu dokumenata. Kao rezultat, češće riječi poput "the" dobivaju nižu vrijednost, dok riječi poput "barked", koje su rjeđe, dobivaju veću važnost.

TF-IDF se koristi u raznim zadacima, uključujući pretraživanje informacija, analizu teksta i klasifikaciju dokumenata. Koristi se za određivanje ključnih riječi unutar dokumenta, omogućujući lakše prepoznavanje i rangiranje relevantnih dokumenata u pretraživanjima. Na primjer, u pretraživanju informacija, TF-IDF pomaže tražilicama da identificiraju dokumente s relevantnim informacijama na temelju upita korisnika, ističući ključne termine i ignorirajući uobičajene riječi poput "the" ili "and" [2].

Jedna od glavnih prednosti TF-IDF-a je njegova sposobnost mjerenja relevantnosti pojma unutar dokumenta na temelju učestalosti riječi i inverzne učestalosti dokumenata, što omogućuje precizno identificiranje ključnih riječi koje su najrelevantnije za određeni dokument. Također, TF-IDF se može uspješno primijeniti na velikim korpusima teksta jer je skalabilan i efikasan u obradi velikih količina podataka. Dodatno, TF-IDF automatski smanjuje težinu zaustavnih riječi koje su česte, ali nisu relevantne za analizu, čime se poboljšava preciznost rezultata [21]. Njegova svestranost omogućuje primjenu u raznim NLP zadacima, kao što su klasifikacija teksta, pronalaženje informacija i grupiranje dokumenata. Ocjene koje generira TF-IDF lako su interpretativne, što olakšava analizu rezultata i donošenje zaključaka [2]. Još jedna prednost je njegova kompatibilnost s različitim jezicima i kodiranjima, čineći ga pogodnim za višejezične tekstualne podatke [20].

Međutim, TF-IDF ima i nekoliko ograničenja. Primarno, zanemaruje kontekst u kojem se riječi pojavljuju te se oslanja isključivo na učestalost pojmova, što može dovesti do netočnog

tumačenja značenja dokumenta. Nadalje, pretpostavlja neovisnost pojmova, iako su riječi u prirodnom jeziku često međusobno povezane [21]. Veličina rječnika može postati vrlo velika pri radu s velikim skupovima podataka, što povećava složenost obrade. TF-IDF ne uzima u obzir redoslijed riječi, što može biti problematično u zadacima gdje je taj redoslijed ključan, poput analize osjećaja. Također, ograničen je na mjerenje učestalosti pojmova, ignorirajući druge važne značajke poput duljine dokumenta ili položaja riječi [20]. Na kraju, iako smanjuje težinu zaustavnih riječi, TF-IDF može ostati osjetljiv na njih ukoliko nisu ispravno uklonjene prije obrade [2].

3.1.3 Word2Vec

Word2Vec je popularna tehnika u obradi prirodnog jezika koja se koristi za stvaranje vektorskih reprezentacija riječi. Ova metoda, razvijena od strane istraživača u Googleu 2013. godine, omogućava da se riječi iz tekstualnih podataka pretvore u numeričke vektore koji zadržavaju semantičke odnose između riječi [22]. Word2Vec je revolucionirao način na koji NLP sustavi razumiju i procesiraju jezik, jer omogućava da slične riječi imaju slične vektorske reprezentacije.

Word2Vec model radi tako da uzima ogromne korpuse teksta i uči vektorske reprezentacije riječi na temelju njihovog konteksta. To znači da su riječi koje se pojavljuju u sličnim kontekstima unutar teksta prikazane vektorima koji su blizu jedni drugima u višedimenzionalnom prostoru. Postoje dva glavna pristupa za treniranje Word2Vec modela: *Continuous Bag of Words* (CBOW) i *Skip-Gram*.

- CBOW (Continuous Bag of Words): Ovaj model predviđa ciljnu riječ koristeći riječi u njenoj okolini (kontekstu). Primjerice, u rečenici "The cat sat on the mat," CBOW model bi pokušao predvidjeti riječ "sat" koristeći riječi "The", "cat", "on", i "the". CBOW je učinkovitiji kada se radi s velikim korpusima podataka, jer koristi prosječnu vektorsku reprezentaciju svih riječi u kontekstu za predikciju ciljane riječi [22].
- Skip-Gram: Za razliku od CBOW-a, Skip-Gram model koristi ciljnu riječ kako bi predvidio riječi u njenom kontekstu. Na primjer, koristeći riječ "sat", model bi pokušao predvidjeti riječi "The", "cat", "on", i "the". Skip-Gram je učinkovitiji kada se radi s rjeđim riječima i manjim korpusima, jer fokusira na učenje preciznih reprezentacija za riječi iz konteksta [2].

Word2Vec model koristi neuronske mreže za učenje vektorskih reprezentacija riječi. Proces treniranja modela uključuje optimizaciju vektora riječi kako bi se minimizirala razlika između predviđenih i stvarnih riječi u kontekstu. Ovaj proces se ponavlja kroz mnoge iteracije nad tekstualnim podacima sve dok model ne nauči optimalne vektorske reprezentacije za sve riječi u korpusu. Jedan od ključnih aspekata Word2Vec modela je koncept negativnog uzorkovanja, koji pomaže u ubrzavanju treniranja modela. Negativno uzorkovanje uključuje nasumično odabiranje nekoliko riječi koje nisu povezane s ciljanom riječi i njihovim kontekstom, te njihovo korištenje u treniranju modela kako bi se smanjila greška modela [23].

Nakon što je model treniran, vektori koje stvara Word2Vec mogu se koristiti za razne NLP zadatke. Primjerice, vektori mogu biti korišteni za mjerenje sličnosti između riječi, grupiranje sličnih riječi, klasifikaciju teksta, generiranje teksta, i mnoge druge aplikacije. Jedna od najsnažnijih karakteristika Word2Vec modela je sposobnost da uhvati semantičke odnose između riječi, kao što su sinonimija, antonimija, ili relacije [24].

Prednosti Word2Vec modela u odnosu na druge metode vektorizacije su:

- Word2Vec model stvara guste vektorske reprezentacije koje su kompaktne i efikasne za korištenje u mnogim NLP zadacima.
- Model može uhvatiti semantičke i sintaktičke odnose između riječi, što omogućava preciznije analize teksta.
- Word2Vec je fleksibilan i može se primijeniti na različite jezike i domene [22].

Ograničenja Word2Vec modela u odnosu na novije tehnike su:

- Word2Vec model zahtijeva veliki korpus podataka za treniranje kako bi proizveo kvalitetne vektore.
- Model ne uzima u obzir redoslijed riječi u rečenici, što može biti ograničavajuće u nekim NLP zadacima, poput analize sentimenta.
- Word2Vec može biti osjetljiv na odabir hiperparametara kao što su veličina vektora i veličina prozora (context window), što može zahtijevati puno vremena za fino podešavanje [24].

3.1.4 Doc2Vec

Doc2Vec je proširenje popularne tehnike Word2Vec koje omogućava kreiranje vektorskih reprezentacija za cijele dokumente, umjesto pojedinačnih riječi. Ova metoda omogućava da dokumenti budu predstavljeni kao gusti, numerički vektori koji zadržavaju semantičke informacije iz teksta. To je korisno u zadacima kao što su klasifikacija dokumenata, grupiranje i preporuke.

Doc2Vec koristi sličan pristup kao Word2Vec, ali proširuje model dodavanjem jedinstvenog identifikatora (vektora) za svaki dokument u korpusu. Cilj Doc2Vec-a je treniranje modela koji može generirati vektorsku reprezentaciju dokumenta na temelju njegovog sadržaja. Postoje dva glavna pristupa u Doc2Vec-u: *Distributed Memory* (DM) i *Distributed Bag of Words* (DBOW).

- Distributed Memory (DM): Ovaj pristup je sličan CBOW modelu iz Word2Vec-a. DM model trenira reprezentaciju dokumenta koristeći riječi u njegovom kontekstu. Tijekom treniranja, model pokušava predvidjeti ciljnu riječ koristeći okolne riječi u kontekstu, kao i identifikator dokumenta. Reprezentacija dokumenta se tako ažurira zajedno s reprezentacijama riječi [25].
- Distributed Bag of Words (DBOW): Ovaj pristup je sličan Skip-Gram modelu. Umjesto predviđanja ciljnih riječi pomoću konteksta, DBOW model koristi dokument kao kontekst za predviđanje riječi u dokumentu. Ovaj pristup ne koristi eksplicitne vektore za riječi, već samo za dokumente, čineći ga jednostavnijim, ali također učinkovitim [26].

Doc2Vec je posebno koristan u zadacima klasifikacije i grupiranja tekstualnih podataka. Njegova sposobnost da generira vektore koji predstavljaju cijele dokumente omogućava modelima strojnog učenja da lakše klasificiraju i analiziraju tekst na temelju njegovog sadržaja. Na primjer, u višeklasnoj klasifikaciji tekstualnih podataka, Doc2Vec može generirati reprezentacije za različite dokumente i omogućiti preciznije klasifikacije pomoću metoda poput logističke regresije ili SVM-a [27].

Proces treniranja Doc2Vec modela uključuje dva glavna koraka:

1. Priprema podataka: Svaki dokument se označava jedinstvenim identifikatorom (tagged document) i tekst se dijeli na riječi.
2. Treniranje modela: Model koristi vektore riječi i dokumenta kako bi naučio semantičke odnose između riječi unutar dokumenta i dokumenta unutar korpusa.

Prednosti Doc2Vec modela u odnosu na druge metode vektorizacije su:

- Semantička reprezentacija dokumenata: Doc2Vec omogućava treniranje vektora koji zadržavaju semantičke informacije o cijelom dokumentu, što omogućava bolje razumijevanje konteksta i značenja dokumenta.
- Sposobnost skaliranja: Doc2Vec može skalirati na velike korpuse dokumenata i učinkovito trenirati na velikim skupovima podataka.
- Generalizacija na nove dokumente: Doc2Vec može generirati vektore za nove, nepoznate dokumente, omogućujući da model bude primjenjiv na stvarne probleme gdje dolazi do dodavanja novih podataka.

Ograničenja Doc2Vec modela u odnosu na druge tehnike su:

- Gubitak sintaktičkih informacija: Budući da Doc2Vec fokusira na semantičke informacije, može izgubiti važne sintaktičke detalje koji su ključni za neke NLP zadatke.
- Trening zahtijeva puno podataka: Kao i kod mnogih tehnika dubokog učenja, treniranje Doc2Vec modela zahtijeva veliki korpus podataka kako bi proizveo kvalitetne vektorske reprezentacije [26].

Doc2Vec predstavlja moćnu tehniku za generiranje vektorskih reprezentacija cijelih dokumenata. Kroz svoje dvije glavne varijante, DM i DBOW, omogućava modelima da hvataju semantičke odnose unutar i između dokumenata. Doc2Vec je ključan alat u zadacima kao što su klasifikacija dokumenata, grupiranje i pretraživanje informacija. Korištenjem *Gensim* biblioteke, Doc2Vec se može lako integrirati u razne NLP projekte, pružajući duboke uvide u sadržaj teksta i dokumenta [28].

3.2 Korišteni algoritmi

U ovom radu korišteni su ključni algoritmi strojnog učenja za višestruku klasifikaciju oznaka, uključujući logičku regresiju, slučajnu šumu i klasifikator s glasanjem. Svaki od ovih algoritama ima specifične prednosti u klasifikaciji tekstualnih podataka, a zajedno omogućuju kombinaciju različitih pristupa za optimizaciju točnosti modela. Korištenje ovih algoritama omogućilo je balansiranje jednostavnosti, interpretabilnosti i robusnosti u klasifikaciji složenih podataka.

3.2.1 Logička regresija

Logička regresija je temeljni algoritam za klasifikaciju koji se koristi za modeliranje odnosa između značajki i vjerojatnosti pripadnosti klasi. Temelji se na pretpostavci linearnosti između ulaznih značajki i logaritma omjera vjerojatnosti (*logit*). Za razliku od linearnog regresijskog modela koji predviđa kontinuirane vrijednosti, logistička regresija koristi logističku funkciju kako bi ograničila izlazne vrijednosti između 0 i 1, omogućujući predviđanje vjerojatnosti pripadnosti određenoj klasi [2].

Logistička regresija modelira vjerojatnost pripadnosti klasi kao funkciju značajki koje opisujemo vektorom $\boldsymbol{x} = [x_1, x_2, \dots, x_n]$. Model računa linearnu kombinaciju značajki i težina $w_1, w_2, \dots, w_n$ te primjenjuje logističku funkciju kako bi predvidio vjerojatnost p da uzorak pripada klasi:

$$p(x) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}}$$

Gdje su w_0 pristranost (engl. bias) i $w_1, \dots, w_n$ težine za značajke.

Logistička regresija je najpoznatija po svojoj primjeni u binarnim klasifikacijama, ali se može proširiti na višestruku klasifikaciju oznaka kroz različite pristupe. U ovom radu korištena je OneVsRestClassifier tehnika, gdje se trenira jedan klasifikator za svaku oznaku, omogućujući klasifikaciju primjera u više klasa istovremeno. To znači da za svaku oznaku (žanr u slučaju ovog rada), model uči poseban klasifikator koji odlučuje pripada li dokument toj oznaci ili ne [2].

Logistička regresija nudi nekoliko prednosti:

- Jednostavnost i interpretabilnost: Model je lako razumljiv, a parametri se mogu interpretirati kao težine koje određuju doprinos svake značajke rezultatu.
- Efikasnost: Logistička regresija se brzo trenira čak i na velikim skupovima podataka, što je korisno za zadatke s velikim količinama tekstualnih podataka.

Međutim, model također ima ograničenja:

- Linearnost: Logistička regresija predviđa linearnu razdiobu podataka, što može biti problematično za složenije, nelinearne odnose među značajkama.

- Prekomjerna prilagodba (overfitting): Logistička regresija može imati tendenciju prekomjerne prilagodbe kada postoji veliki broj značajki ili maleni skup podataka, pa je potrebno primijeniti regularizaciju kako bi se spriječio ovaj problem [29].

U tekstualnoj klasifikaciji, logistička regresija često koristi metode vektorizacije kao što su TF-IDF ili CountVectorizer za pretvaranje tekstualnih podataka u numeričke značajke. Ovaj postupak omogućava modelu da uči na osnovi frekvencije riječi i odnosa među njima unutar dokumenata, čineći ga korisnim za zadatke poput klasifikacije dokumenata, sentiment analize i višestruke klasifikacije oznaka žanrova knjiga, kao što je slučaj u ovom radu [2].

Logistička regresija ostaje jedan od najpouzdanijih i najkorištenijih algoritama za klasifikaciju teksta, zahvaljujući svojoj jednostavnosti, brzini i efikasnosti. U kombinaciji s tehnikama višestruke oznake, ovaj algoritam omogućuje točnu i efikasnu klasifikaciju više kategorija unutar velikih i složenih skupova podataka.

3.2.2 Slučajna šuma

Slučajna šuma (engl. RandomForestClassifier*)* je popularan "asambl" algoritam koji koristi više stabala odlučivanja kako bi poboljšao točnost klasifikacije i smanjio rizik od prekomjerne prilagodbe. Ovaj model trenira niz stabala odlučivanja na različitim podskupovima podataka i kombinira njihove rezultate pomoću glasanja vjerojatnosti ili uzimanjem srednje vrijednosti, odnosno prosjeka vjerojatnosti koje svaki model predviđa za svaku klasu. Prednost ovog pristupa je njegova robusnost, otpornost na prekomjernu prilagodbu te sposobnost rukovanja velikim i složenim skupovima podataka [30].

Algoritam se sastoji od nekoliko stabala odlučivanja, gdje svako stablo daje predikciju za određeni uzorak, a konačna odluka donosi se glasanjem. Korištenjem slučajnih podskupova značajki i uzoraka podataka, model stvara raznolike stabla, čime smanjuje korelaciju između stabala i time povećava točnost predikcije. Svako stablo može se trenirati na različitim podskupovima značajki, što smanjuje varijancu i rizik od prekomjernog opremanja (engl. overfitting) [31].

U ovom radu, slučajna šuma se koristi za višestruku klasifikaciju oznaka. Algoritam generira predikcije za više žanrova, dopuštajući svakom stablu da doprinosi konačnoj klasifikaciji. Korištenjem glasanja ili prosjeka predikcija iz više stabala, model pruža stabilnije i preciznije rezultate. Zbog svoje sposobnosti rukovanja s velikim brojem oznaka i prilagodbe različitim

tipovima podataka, slučajna šuma je učinkovit izbor za ovaj problem [32]. Omogućava modelu da se bolje nosi s varijacijama u tekstualnim podacima i istovremeno klasificira primjere u više klasa. Ova metoda osigurava visoku točnost predikcija zahvaljujući kombiniranju predikcija više stabala.

Prednosti slučajne šume u odnosu na druge klasifikatore su:

- Robusnost: Model je vrlo otporan na prekomjernu prilagodbu zbog korištenja više stabala koja treniraju na različitim podskupovima podataka.
- Točnost: Zbog kombiniranja predikcija više stabala, model često nadmašuje pojedinačne klasifikatore.
- Fleksibilnost: slučajna šuma dobro funkcionira i s numeričkim i s kategorijskim podacima, čime je prikladan za različite tipove zadataka [32].

Nedostaci slučajne šume u odnosu na jednostavnije modele su:

- Složenost modela: Zbog velikog broja stabala, model može postati težak za interpretaciju.
- Računalna složenost: Treniranje velikog broja stabala može biti računalno zahtjevno, osobito na velikim datasetima [32].

3.2.3 Klasifikator s glasanjem

Klasifikator s glasanjem (engl. VotingClassifier) je "asambl" metoda koja kombinira predikcije više modela kako bi poboljšala ukupne rezultate klasifikacije. Ovaj model omogućuje da se koriste predikcije različitih klasifikatora te na temelju njih donosi konačnu odluku. Postoje dva glavna načina kombiniranja predikcija: tvrdo glasanje (engl. hard voting) i meko glasanje (engl. soft voting).

- Tvrdo glasanje: Kod tvrdog glasanja, svaki model daje svoju predikciju za klasu, a konačna odluka je ona koju je odabrala većina modela (većinsko glasanje). Ovo je korisno u slučajevima kada želite kombinirati različite modele i odabrati klasu koju je većina modela predvidjela. Tvrdo glasanje je posebno učinkovit kada svi modeli imaju slične performanse, ali različite točke snage.
- Meko glasanje: Kod mekog glasanja, koristi se prosjek vjerojatnosti koje svaki model predviđa za svaku klasu. Konačna klasa je ona s najvećom prosječnom vjerojatnošću.

Meko glasanje je koristan kada modeli mogu proizvesti vjerojatnosti, a neki modeli mogu biti precizniji od drugih. Ova metoda daje veću fleksibilnost u tome koliko svaki model doprinosi konačnoj odluci, ovisno o njegovoj sigurnosti u predikciji [33].

Klasifikator s glasanjem je posebno koristan za kombiniranje različitih tipova modela, kao što su logička regresija i slučajna šuma, što omogućava da se iskoriste njihove različite prednosti. U ovom radu, klasifikator s glasanjem kombinira rezultate logičke regresije i slučajne šume, koristeći meko glasanje kako bi se postigla veća preciznost u višestrukoj klasifikaciji oznaka (žanrova knjiga).

Prednosti klasifikatora s glasanjem su:

- Poboljšana točnost: Kombiniranjem više modela, klasifikator s glasanjem može nadmašiti performanse pojedinačnih klasifikatora, čineći model robusnijim.
- Fleksibilnost: Moguće je kombinirati modele s različitim karakteristikama, kao što su linearnost i nelinearnost, kako bi se bolje uhvatili svi aspekti podataka [34].

Nedostaci klasifikatora s glasanjem su:

- Računalna složenost: Korištenje više modela može povećati vrijeme treniranja i predikcije.
- Teža interpretacija: Kada se kombiniraju različiti modeli, postaje teže interpretirati zašto je model donio određenu odluku [34].

Ovaj algoritam pokazao se učinkovitim u radu jer kombinira različite modele kako bi se bolje uhvatile složenosti u tekstualnim podacima i osiguralo preciznije predikcije.

3.3 Balansiranje podataka

Balansiranje podataka u klasifikaciji je ključno za postizanje preciznijih i pouzdanijih modela, osobito kada se suočavamo s problemom neuravnoteženih podataka, gdje su neke oznake (klase) značajno manje zastupljene od drugih. U višestrukoj klasifikaciji oznaka, neuravnoteženost klasa može dovesti do toga da modeli favoriziraju češće klase, zanemarujući rijetke.

3.3.1 Stratificirana K-struka unakrsna validacija

Stratificirana K-struka unakrsna validacija (engl. Stratified K-Fold Cross-Validation) je tehnika koja se koristi za balansiranje podataka prilikom podjele u trening i test sklopove. Za razliku od klasične K-struke unakrsne validacije, gdje se podaci dijele nasumično, stratificirana metoda osigurava da su razmjeri svake klase konzistentne u svakom skupu. Ova tehnika podjele osobito je korisna u neuravnoteženim skupovima podataka, gdje može pomoći u postizanju boljih rezultata jer svaki sklop zadržava proporcionalan broj uzoraka svake klase [35].

U višestrukoj klasifikaciji oznaka, uzorci mogu pripadati više klasa istovremeno. Primjerice, jedna knjiga može pripadati žanrovima "fantazija", "drama" i "avantura". Standardna K-struka validacija ne vodi računa o očuvanju ravnoteže klasa tijekom podjele, što može rezultirati neujednačenom raspodjelom oznaka u različitim sklopovima. Multilabel Stratified KFold rješava ovaj problem tako što stratificira podatke prema svakoj oznaci (ili klasi), osiguravajući da proporcije svake oznake ostanu konstantne unutar svakog skupa podataka. To znači da, čak i kada imamo neuravnotežene klase, svaki skup podataka (bilo trening ili test) sadržavat će približno isti postotak svake oznake kao i izvorni skup podataka.

Prednosti stratificirane K-struke unakrsne validacije u odnosu na klasičnu K-struku validaciju su:

- Očuvanje razmjera klasa: Korištenjem stratificirane metode sprječava se dominacija češćih oznaka nad rjeđima, što je ključno u višestrukoj klasifikaciji oznaka, u odnosu na klasičnu K-struku validaciju koja može dovesti do neuravnotežene raspodjele klasa među sklopovima.
- Poboljšana generalizacija: Modeli trenirani uz pomoć stratificiranih sklopova bolje generaliziraju na nove podatke, jer su u trening i test setovima zadržane iste razmjere oznaka kao i u originalnim podacima, u odnosu na klasičnu metodu koja može rezultirati lošijim performansama zbog neuravnoteženih sklopova [35].

Nedostaci Stratificirane K-struke unakrsne validacije u odnosu na klasičnu K-struku validaciju mogu uključivati sljedeće:

- Računalna složenost: Stratificirana K-struka unakrsna validacija može biti računalno složenija i sporija od klasične K-struke validacije, osobito kod velikih skupova podataka, jer je potrebno dodatno osigurati proporcionalnu distribuciju oznaka u svakom sklopu.

- Složenost implementacije: U višestrukoj klasifikaciji oznaka, osiguravanje ravnoteže za svaku oznaku može biti tehnički složenije, što zahtijeva specifične alate i algoritme poput Multilabel Stratified KFold, dok klasična K-struka validacija ne zahtijeva takve prilagodbe.
- Ograničena primjena kod vrlo neuravnoteženih podataka: Ako su podaci izrazito neuravnoteženi (npr. neke oznake su vrlo rijetke), čak i stratificirana validacija možda neće uspjeti potpuno uravnotežiti sklopove, što može negativno utjecati na performanse modela.

U ovom radu, Multilabel Stratified KFold koristi se za podjelu podataka na trening i test setove. Korištenje ove metode osigurava balansirane skupove podataka u svakom dijelu unakrsne validacije, čime se postiže veća točnost i robusnost modela, posebno kod rijetkih oznaka.

4 Skup podataka s opisima knjiga i žanrovima

U ovom poglavlju detaljno ćemo opisati skup podataka korišten za klasifikaciju žanrova knjiga primjenom tehnika obrade prirodnog jezika. Skup podataka naziva "books", u CSV formatu, sadrži informacije o 52,478 knjiga, uključujući atribute poput naslova, autora, ocjena, jezika, broja stranica, datuma objavljivanja, te najvažnije za ovaj rad opise knjiga i žanrove. Ovaj skup podataka pruža bogat izvor informacija za statističku analizu i treniranje modela strojnog učenja.

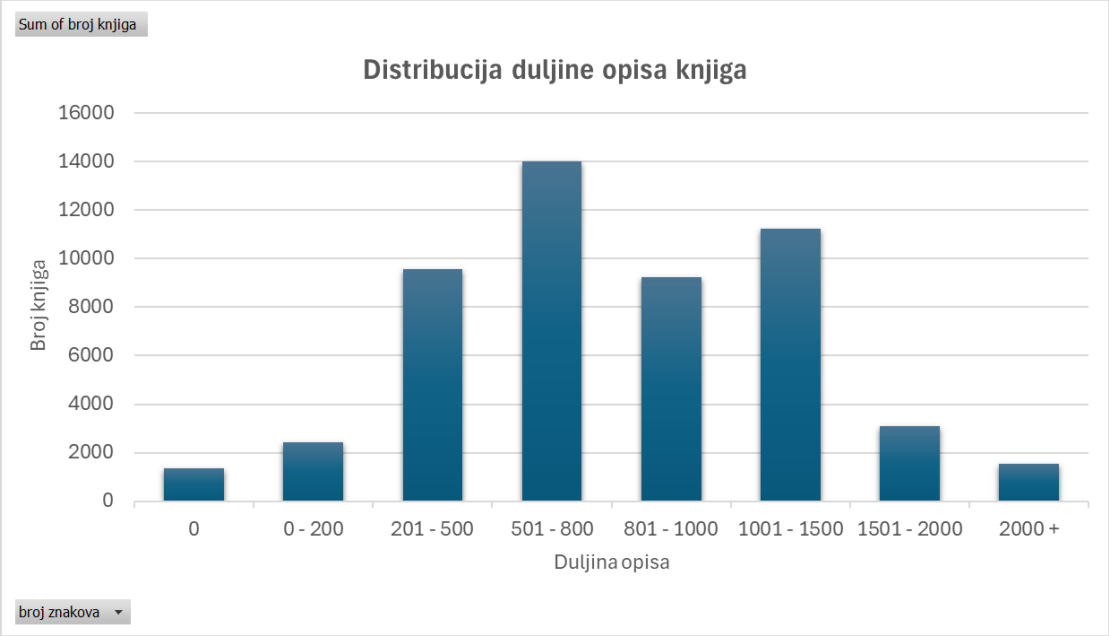
Podaci su preuzeti sa platforme Kaggle[1], i sadrže opise, autore, ocjene, brojeve stranica, žanrove za knjige i druge informacije o knjigama. Međutim, za ovu analizu fokus je stavljen na tekstualne opise knjiga i pripadajuće žanrove.

Skup podataka se sastoji od 23 stupca, od kojih su najvažniji za pojedinu knjigu:

- title: Naslov knjige (52,478 vrijednosti, bez praznih podataka)
- author: Autor knjige (52,478 vrijednosti)
- rating: Ocjena knjige (prosječna ocjena korisnika) u rasponu od 0 do 5 (52,478 vrijednosti)
- description: Opis knjige, ključan za analizu tekstualnih podataka (51,140 vrijednosti, s 1,338 praznih)
- genres: Žanrovi knjige u obliku liste (52,478 vrijednosti, bez praznih podataka)
- language: Jezik knjige (48,672 vrijednosti)
- numRatings: Ukupan broj ocjena za knjigu (52,478 vrijednosti)
- publishDate: Datum objavljivanja knjige (51,598 vrijednosti)
- pages: Broj stranica knjige (50,131 vrijednost)

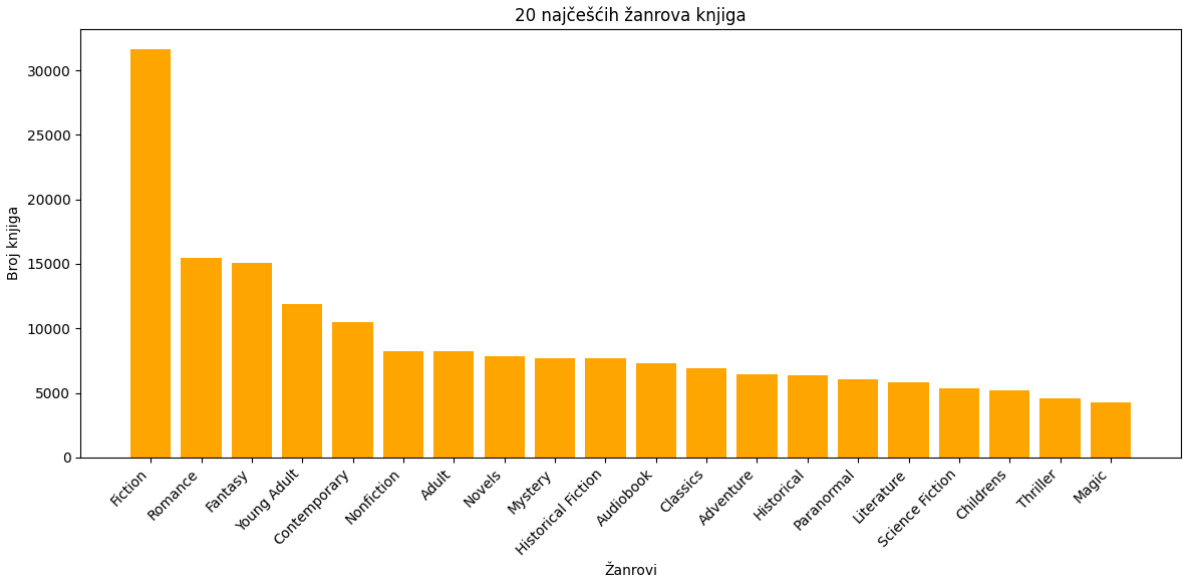
Opis knjiga je jedan od ključnih atributa skupa podataka jer predstavlja neobrađeni tekst koji sadrži informacije o radnji, likovima, stilu pisanja i temama koje su relevantne za klasifikaciju žanrova. U skupu podataka, stupac description sadrži 51,140 opisa knjiga, dok je za 1,338 knjiga taj podatak nedostupan.

[1] Skup podataka dostupan je na platformi Kaggle: https://www.kaggle.com/datasets/meetnaren/goodreads-best-books



Slika 3 Distribucija duljine opisa knjiga

Duljina opisa knjiga varira značajno. Neki opisi sadrže samo nekoliko rečenica, dok drugi uključuju detaljne sinopsise ili čak odlomke iz knjige. Prosječna duljina opisa je oko 600 riječi, dok najduži opisi prelaze i 10 000 riječi. Graf na Slici 3 prikazuje distribuciju duljine opisa knjiga. Na grafu, x-os prikazuje duljinu opisa knjiga u različitim rasponima znakova, dok y-os prikazuje broj knjiga koje pripadaju svakom rasponu duljine opisa. Najveći broj knjiga ima opise duljine između 501 i 800 znakova, dok su knjige s kraćim (do 200 znakova) i dužim opisima (preko 2000 znakova) manje zastupljene.



Slika 4 Distribucija dvadeset najčešćih žanrova knjiga

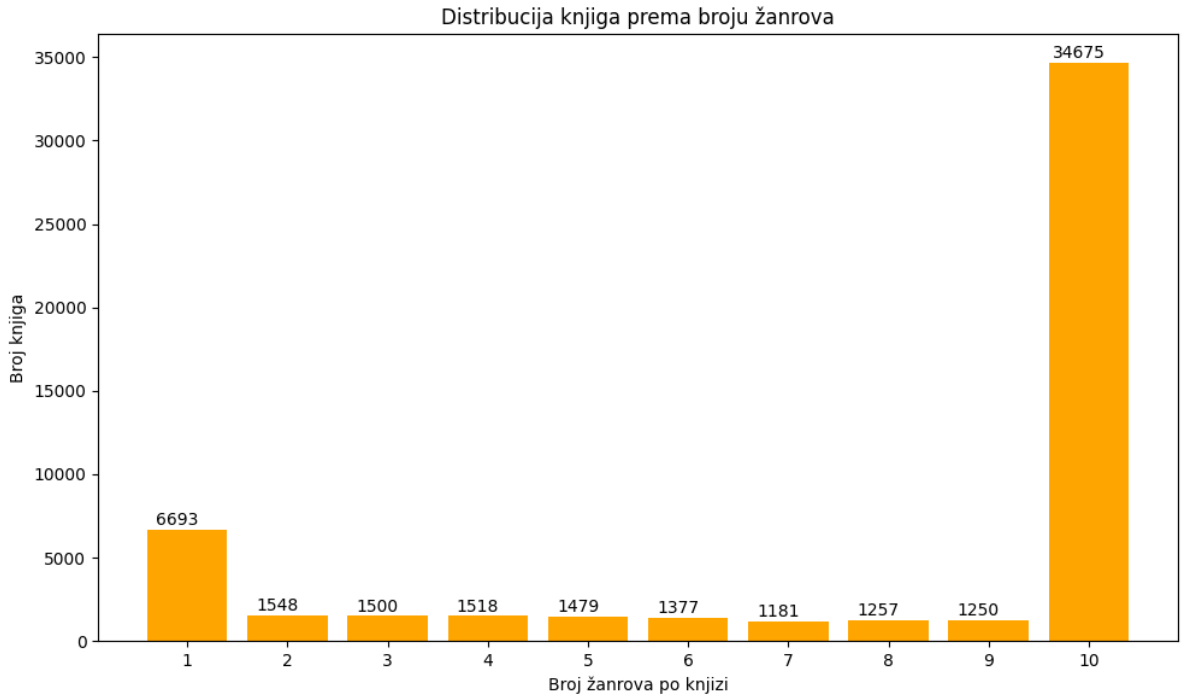
Graf na Slici 4 prikazuje distribuciju 20 najčešćih žanrova knjiga. Najzastupljeniji žanr je Fiction s više od 30.000 knjiga, dok Romance, Fantasy i Young Adult također imaju značajan broj knjiga, između 10.000 i 20.000. Ostali žanrovi kao što su Contemporary, Nonfiction, Adult, i Mystery su manje zastupljeni, ali i dalje čine velik dio skupa podataka. Najmanje zastupljeni žanrovi u ovoj grupi su Thriller i Magic, svaki s manje od 5.000 knjiga.

Žanrovi su kategorizirani atributi koji služe kao oznake za višestruku klasifikaciju knjiga. U skupu podataka postoje 983 različitih žanrova knjiga. To pokazuje raznolikost žanrova koji su prisutni u opisu knjiga, a mnogi od tih žanrova uključuju one kategorije koje se manje pojavljuju kao što su "Comics Manga", "Reverse Harem", "Virtual Reality", "Ancient History", "Hard Science Fiction", "Food Writing", i mnogi drugi. Postoje i parovi žanrova koji se pojavljuju isključivo zajedno. Primjer takvih žanrova su "Fat Acceptance" i "Fat Studies". To znači da se u svakom zapisu gdje se pojavljuje jedan od ovih žanrova, uvijek pojavljuje i drugi. Takva vrsta odnosa je rijetka, jer većina žanrova može postojati u kombinaciji s različitim drugim žanrovima.

Svaka knjiga može pripadati jednom ili više žanrova, ovisno o sadržaju i temi. U skupu podataka žanrovi nisu ravnomjerno raspodijeljeni. Neki žanrovi, poput Fiction i Fantasy, pojavljuju se vrlo često, dok su drugi, poput Science Fiction ili Nonfiction, manje zastupljeni. Najzastupljeniji žanrovi u skupu podataka su:

- Fiction: Pojavljuje se u više od 50% knjiga. Ovaj žanr obuhvaća širok spektar fikcionalnih djela, što ga čini vrlo svestranim, ali i teškim za klasifikaciju.
- Fantasy: Drugi najzastupljeniji žanr, prisutan u više od 25% knjiga. Fantasy žanr često uključuje elemente magije, mitologije i nadnaravnih elemenata.
- Romance: Ovaj žanr je posebno popularan u knjigama namijenjenima širokoj publici, a pojavljuje se u oko 20% knjiga.
- Young Adult: Fokusira se na mladu publiku i obuhvaća teme odrastanja, školskih problema, ali i fantazije i ljubavi. Ovaj žanr je prisutan u oko 15% knjiga.
- Nonfiction: Knjige iz ovog žanra predstavljaju stvarne događaje, biografije, povijesne knjige i priručnike.

Mnoge knjige pripadaju više od jednog žanra. Na primjer, knjiga može istovremeno biti označena kao Fantasy, Young Adult, i Romance. Ova višestruka klasifikacija dodatno komplicira zadatak strojnog učenja jer model mora pravilno prepoznati sve relevantne žanrove za pojedinu knjigu.



Slika 5 Distribucija knjiga prema broju žanrova

Grafikon koji je prikazan na Slici 5 predstavlja distribuciju knjiga prema broju žanrova. Brojevi žanrova koje knjige mogu imati (od 1 do 10) se uspoređuju s brojem knjiga koje imaju odgovarajući broj žanrova. Najveći broj knjiga (34,675) ima 10 žanrova, što je značajan udio. Knjige s jednim žanrom također čine značajan dio, s 6,693 knjiga. Knjige s dva do devet žanrova javljaju se u mnogo manjem broju.

Statistička analiza žanrova

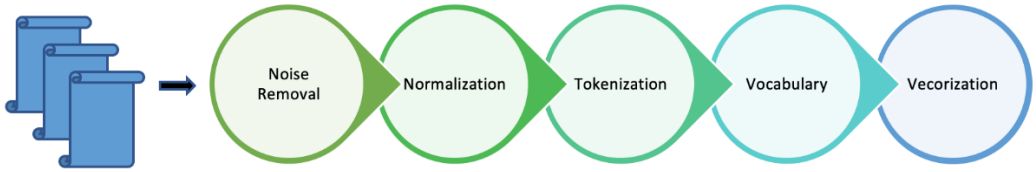
- Frekvencija pojavljivanja žanrova: Fiction je daleko najzastupljeniji žanr, pojavljujući se u više od 60% knjiga u skupu podataka. Nasuprot tome, žanrovi poput Poetry i Graphic Novels pojavljuju se rjeđe, u manje od 5% knjiga.
- Distribucija žanrova: Većina knjiga pripada jednom do tri žanra. Knjige s četiri ili više žanrova rijetke su i obično pripadaju kompleksnijim pričama koje spajaju elemente više različitih stilova pisanja (npr. Historical Fiction + Romance + Mystery).

5 Implementacija modela

U ovoj implementaciji, korišteno je nekoliko tehnika vektorizacije i algoritama strojnog učenja kako bi se klasificirali tekstualni podaci prema višestrukim žanrovima. Podaci su pripremljeni kroz procesiranje i vektorizaciju, a korišteni modeli uključuju logičku regresiju, slučajnu šumu i klasifikator s glasanjem. Vektorizacija se provodi putem TF-IDF i Word2Vec, dok su rezultati optimizirani uz pomoć GridSearchCV. Cilj je bio postići točne predikcije za višestruke klasifikacije oznaka na temelju tekstualnih opisa knjiga.

5.1 Pretprocesiranje podataka

Pretprocesiranje podataka u obradi prirodnog jezika (NLP) ključan je korak za osiguravanje da su podaci u formatu pogodnom za modele strojnog učenja. U ovom radu, pretprocesiranje podataka uključuje niz koraka koji pripremaju tekstualne podatke za kasnije faze vektorizacije i treniranja modela.



Slika 6 Koraci u pretprocesiranju teksta [7]

Niz koraka koji su ključni u procesu predprocesiranja teksta u kontekstu obrade prirodnog jezika (NLP) prikazani su Slikom 6. Predprocesiranje teksta je važan korak koji priprema sirove tekstualne podatke za analizu i modeliranje. Svaki od koraka prikazanih na slici odvija se u određenom redoslijedu kako bi se osiguralo da su podaci u što čišćem, standardiziranom i reprezentativnom obliku prije nego što se primijene složeniji algoritmi za obradu ili strojno učenje.

Koraci koji se mogu koristiti u pretprocesuiranju teksta su:

1. Uklanjanje šuma podrazumijeva eliminaciju nepotrebnih ili neželjenih elemenata iz teksta koji ne doprinose njegovom semantičkom značenju. To mogu biti brojevi, interpunkcijski znakovi, posebni znakovi, HTML oznake ili drugi elementi koji ometaju analizu teksta.
2. Normalizacija je proces standardizacije teksta. To uključuje pretvaranje svih slova u mala slova kako bi se izbjegle razlike između velikih i malih slova, uklanjanje

naglasaka i diakritičkih znakova, te standardizacija oblika riječi (npr. konverzija svih različitih oblika riječi u njihov osnovni oblik).

3. Tokenizacija je proces razbijanja teksta na manje dijelove, poznate kao tokeni, gdje svaki token može predstavljati riječ, frazu ili čak pojedinačne simbole, ovisno o specifičnom zadatku. Tokenizacija omogućuje daljnju analizu teksta jer pojednostavljuje rad s riječima ili frazama.
4. Kreiranje vokabulara uključuje identifikaciju i organizaciju svih jedinstvenih riječi ili tokena u skupu podataka. Vokabular predstavlja skup svih riječi koje model treba "naučiti" ili obraditi tijekom analize.
5. Vektorizacija je proces pretvaranja riječi ili tokena u numeričke vrijednosti (vektore) koje modeli strojnog učenja mogu koristiti za analizu. Vektorizacija omogućuje da tekstualni podaci budu predstavljeni u formatu koji je razumljiv računalnim algoritmima. Primjeri vektorizacije uključuju metode kao što su Bag-of-Words, TF-IDF, Word2Vec, itd.

Prvi korak je učitati podatke. Učitavanje podataka vrši se iz CSV datoteke koja sadrži informacije o knjigama kao što su naslovi, autori, opisi, ocjene i žanrovi:

```
import pandas as pd

file_path = 'books.csv'
books_df = pd.read_csv(file_path)
```

Kod 1 Učitavanje podataka

Biblioteka *Pandas* (pd) je ključna za učitavanje i manipulaciju podataka u obliku DataFrame-a. Omogućuje jednostavan rad s velikim skupovima podataka putem različitih metoda. Na primjer, funkcija pd.read_csv() koristi se za učitavanje CSV datoteka u pandas DataFrame, čime se podaci pretvaraju u strukturu pogodnu za analizu.

U sljedećem prikazanom kodu (Kod 2) je napravljen sljedeći korak, a to je uklanjanje praznih opisa i obrada teksta. Podaci koji sadrže nedostajuće vrijednosti u stupcu 'description' (opis) su uklonjeni kako bi se osiguralo da svi korišteni primjeri imaju potpune tekstualne podatke, to je obavljeno korištenjem funkcije *dropna()*.

```
from sklearn.feature_extraction.text import ENGLISH_STOP_WORDS as
sklearn_stop_words

books_df = books_df.dropna(subset=['description'])
import re
```

```
def preprocess_text(text):
    text = re.sub(r'\d+', '', text)
    text = re.sub(r'[^\w\s]', '', text)
    text = text.lower()
    words = text.split()
    filtered_words = [word for word in words if word not in
sklearn_stop_words]
    processed_text = ' '.join(filtered_words)
    return processed_text
```

Kod 2 Obrada teksta

Obrada tekstualnih podataka neophodna je kako bi se uklonili nepotrebni elementi i pripremio čisti tekst za modele strojnog učenja. U ovom slučaju kao što je prikazano u Kodu 2, kreirana je funkcija *preprocess_text* koja obavlja sljedeće operacije:

- Uklanjanje brojeva: Svi numerički znakovi uklanjaju se iz opisa knjiga korištenjem regularnog izraza r'\d+'.
- Uklanjanje interpunkcije: Koristeći regularni izraz r'[^\w\s]', uklanjaju se svi znakovi osim riječi i razmaka.
- Transformacija u mala slova: Tekst se konvertira u mala slova kako bi se osiguralo da je tekst standardiziran.
- Uklanjanje stop riječi: Iz opisa knjiga uklanjaju se uobičajene riječi koje ne doprinose semantici teksta, poput "and", "the", "is", koristeći definirane stop riječi iz biblioteke sklearn.

Korištena je Pythonova biblioteka 're' za rad s regularnim izrazima (enlg. *regular expressions)*, koja omogućava pretraživanje, zamjenu i manipulaciju tekstom na temelju uzorka. Iz ove biblioteke je korišten funkcija *re.sub()* koja se koristi za zamjenu uzorka u tekstu. Sintaksa je *re.sub(pattern, replacement, string)*, gdje:

- *pattern* označava regularni izraz koji treba pronaći,
- *replacement* je tekst koji će zamijeniti pronađeni uzorak,
- *string* je tekst u kojem se vrši zamjena.

Izraz *r' '* označava "raw" string, gdje svi znakovi unutar stringa interpretiraju se doslovno. Ovo je važno za regularne izraze jer izbjegava specijalno tretiranje escape sekvenci (npr. \n ili \t). Na primjer, r'\d+' znači da se traže svi brojevi (\d označava znamenke) u tekstu.

Za konvertiranje znakova u stringu u mala slova korištena je metoda *lower()*, a za razdvajanjje stringa u listu riječi na temelju razmaka korištena je metoda *split()*. Nakon toga je korištena i

metoda *join()* za ponovno spajanje liste riječi u string, koristeći razmak kao separator između riječi.

Ova funkcija primijenjena je na opis knjiga kako si se svaki tekstualni opis pretvorio u format spreman za analizu. Rezultat je tekst bez brojeva, interpunkcije, velikih slova i stop riječi, što omogućuje učinkovitiju vektorizaciju i analizu. Korištena je funkcija *apply()* koja primjenjuje funkciju *preprocess_text* na svaki redak u stupcu *description*. Rezultati obrade se spremaju u novi stupac *processed_description*, koji sadrži pročišćene opise knjiga spremne za daljnje analize.

```
books_df['processed_description'] =
books_df['description'].apply(preprocess_text)
```

Kod 3 Primjena funkcije za obradu teksta

Na primjer ako primijenimo metodu *preprocess_text* na opis knjige *Zločin i kazna* autora Fjodor Mihajlovič Dostojevski koji je sljedeći:

> *Raskolnikov, a destitute and desperate former student, wanders through the slums of St Petersburg and commits a random murder without remorse or regret. He imagines himself to be a great man, a Napoleon: acting for a higher purpose beyond conventional moral law. But as he embarks on a dangerous game of cat and mouse with a suspicious police investigator, Raskolnikov is pursued by the growing voice of his conscience and finds the noose of his own guilt tightening around his neck. Only Sonya, a downtrodden prostitute, can offer the chance of redemption*

dobit ćemo sljedeći rezultat:

> *raskolnikov destitute desperate former student wanders slums st petersburg commits random murder remorse regret imagines great man napoleon acting higher purpose beyond conventional moral law embarks dangerous game cat mouse suspicious police investigator raskolnikov pursued growing voice conscience finds noose guilt tightening around neck sonya downtrodden prostitute offer chance redemption*

Kao što vidimo, uklonjeni su brojevi, interpunkcija, velika slova i stop riječi, ostavljajući samo ključne riječi koje zadržavaju osnovni smisao teksta.

Osim obrade tekstualnih opisa, potrebno je obraditi i informacije o žanrovima knjiga. U tu svrhu kreirana je funkcija *clean_genres*, koja čisti i normalizira podatke o žanrovima. Ova funkcija uklanja nepotrebne znakove poput zagrada i apostrofa te dijeli žanrove u liste.

```
def clean_genres(genres):
    cleaned_genres = []
    for genre in genres:
        genre = genre.strip()
        genre = genre.replace("'", "").replace("[", "").replace("]", "")
        cleaned_genres.append(genre)
    return cleaned_genres

books_df['genres'] = books_df['genres'].apply(lambda x:
clean_genres(x.split(',')))
```

Kod 4 Funkcija za čišćenje i normalizaciju žanrova

Funkcija *clean_genres(genres)* koristi se za čišćenje i normalizaciju popisa žanrova iz podataka koja je prikazana u Kodu 4. Postupak funkcionira tako da iterira kroz svaki element (žanr) unutar liste *genres*, uklanja nepotrebne znakove i praznine koristeći metode *strip()* i replace(), te dodaje pročišćene žanrove u novu listu *cleaned_genres*. Konkretno, uklanjaju se vodeći i prateći razmaci oko svake riječi te znakovi poput apostrofa, zareza i uglatih zagrada. Na kraju, vraća se popis očišćenih žanrova spremnih za daljnju obradu. Ova metoda je primijenjena na stupac u kojem su zapisani žanrovi knjige. Koristi se metoda *apply()* koja prolazi kroz svaki redak u stupcu *genres*. Unutar *apply()* funkcije koristi se lambda izraz *lambda x: clean_genres(x.split(','))*, koji prvo dijeli žanrove u listu na temelju zareza pomoću *split(',')*, a zatim na tu listu primjenjuje funkciju *clean_genres*, čime se uklanjaju neželjeni znakovi i razmaci. Rezultat je očišćena lista žanrova za svaki redak u *books_df*.

Pretprocesiranje podataka je osnova svakog NLP zadatka jer osigurava da su podaci očišćeni i strukturirani za daljnju analizu. Kroz uklanjanje brojeva, interpunkcije, normalizaciju teksta te čišćenje žanrova, omogućili smo konzistentan i strukturiran skup podataka koji je spreman za vektorizaciju i treniranje modela.

5.2 Vektorizacija tekstualnih opisa

Za pretvaranje tekstualnih opisa knjiga u numeričke vektore korištene su metode *TF-IDF* i *CountVectorizer* iz *Scikit-learn* biblioteke, te *Word2Vec* i *Doc2Vec* iz *gensim* biblioteke. Ove metode su u prethodnom poglavlju detaljno objašnjene.

Korištenje TfidfVectorizer iz Scikit-learn biblioteke omogućava nam da tekstualne podatke (opise knjiga) pretvorimo u numeričke vektore koji se mogu koristiti za treniranje modela strojnog učenja.

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer

tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = tfidf_vectorizer.fit_transform(books_df['processed_description'])
```

Kod 5 Korištenje metode TF-IDF

TfidfVectorizer vektorizira tekstualne podatke koristeći Term Frequency-Inverse Document Frequency (TF-IDF). Ova metoda dodjeljuje težine riječima temeljem njihove učestalosti unutar dokumenta i cijelog skupa dokumenata. TF-IDF smanjuje važnost riječi koje su vrlo česte u svim dokumentima (npr. "and", "the") te dodaje težinu riječima koje su specifične za određene dokumente, a rjeđe se pojavljuju u ostatku skupa podataka.

Parametar *max_features = 5000* ograničava broj značajki na 5000, što znači da onaj žanr koji ima više ili jednako 5000 primjeraka će biti uključen, što pomaže u smanjenju dimenzionalnosti i zadržavanju najvažnijih informacija. Ovaj prag od 5000 uzoraka odabran je iz nekoliko razloga koje ćemo navesti u nastavku.

U mnogim slučajima kada dođe do problema kod obrade prirodnog jezika, poput klasifikacije žanrova knjiga, podaci su često neuravnoteženi, što znači da se neki žanrovi pojavljuju vrlo često, dok su drugi vrlo rijetki. Žanrovi koji se pojavljuju u manje od 5000 primjera mogli bi donijeti probleme modelu jer rijetki uzorci mogu uzrokovati:

- Pretreniranje: Model se može previše prilagoditi malom broju rijetkih uzoraka, umjesto da generalizira.
- Neuspjeh u učenju: U žanrovima s premalo uzoraka, model možda neće moći naučiti značajne obrasce zbog nedovoljne količine podataka.
- Neadekvatna evaluacija: Metrike modela kao što su preciznost, osjetljivost i F1-mjera mogu biti pristrane ako se rijetki žanrovi ne pojavljuju dovoljno često u testnom skupu.

Prag od 5000 pomaže u eliminiranju rijetkih žanrova koji bi mogli negativno utjecati na performanse modela ili otežati evaluaciju rezultata.

Odabir broja 5000 također je vođen karakteristikama skupa podataka. Taj broj je dovoljno visok da eliminira rijetke žanrove, ali i dovoljno nizak da zadrži dovoljno različitih žanrova kako bi klasifikacija bila korisna i sveobuhvatna.

TfidfVectorizer dolazi iz biblioteke scikit-learn, koja se koristi za strojno učenje u Pythonu. Funkcija fit_transform kombinira dva koraka:

- *fit()*: Analizira podatke kako bi naučila značajke iz skupa podataka. U ovom slučaju, identificira riječi u tekstovima i računa njihove TF-IDF vrijednosti.
- *transform()*: Nakon učenja, transformira ulazni tekst u numeričke vektore koristeći naučene značajke.

Dakle, *fit_transform* najprije uči model TF-IDF na tekstovima, a zatim ih pretvara u numeričke vektore. Ovaj proces transformira tekstualne opise knjiga iz stupca *processed_description* u matricu brojeva *X_tfidf*, gdje svaki redak predstavlja knjigu, a svaki stupac predstavlja jednu od najrelevantnijih 5000 riječi. Vrijednosti u matrici predstavljaju težinu riječi prema TF-IDF mjerama.

U ovom radu koristi se i *CountVectorizer* iz biblioteke *scikit-learn* za pretvaranje teksta u brojčane značajke na temelju učestalosti riječi. Primjena CountVectorizer-a omogućava stvaranje matrice, gdje svaka kolona predstavlja jedinstvenu riječ iz skupa dokumenata, a svaki redak predstavlja dokument. Vrijednosti unutar matrice predstavljaju broj pojavljivanja pojedine riječi unutar određenog dokumenta.

```
count_vectorizer = CountVectorizer(max_features=5000)
X_count = count_vectorizer.fit_transform(books_df['processed_description'])
```

Kod 5 Korištenje metode CountVectorizer

U Kodu 5 korištena je opcija *max_features=5000*, koja ograničava broj značajki na 5000 najrelevantnijih riječi. Ova postavka je korisna kod velikih korpusa teksta, gdje previše značajki može dovesti do preopterećenja modela. Nakon definiranja *CountVectorizer*, metoda *fit_transform()* se koristi kako bi se analizirale tekstualne podatke i stvorila matrica značajki. Za skup opisa knjiga pohranjenih u stupcu *processed_description*, metoda *fit_transform* generira numeričke prikaze tih opisa, koji se zatim koriste u procesu treniranja modela.

Sljedeća metoda za vektorizaciju teksta koja je korištena u radu je Word2Vec. U Kodu 6 prvo smo podijelili opise knjiga na liste riječi i spremamo to u listu *sentences*. Svaki opis postaje lista riječi (tokenizacija). Word2Vec model trenira se na ovim tokeniziranim opisima, gdje svaka riječ se predstavlja vektorom od 100 dimenzija *vector_size=100*. Vektor dimenzije 100 je općenito dovoljno velik da uhvati osnovne semantičke i sintaktičke odnose među riječima, ali ne previše velik da bi doveo do prekomjernog treniranja ili "šuma" u reprezentaciji riječi. To omogućava generalizaciju nad podacima bez gubitka previše informacija. Postavljanje vektora od 100 dimenzija za svaku riječ u vokabularu zahtijeva manje memorije u usporedbi s vektorima većih dimenzija. U kontekstu rada s više modela (kao što su TF-IDF, Word2Vec, i Doc2Vec), smanjenje memorijskog opterećenja može biti ključno za performanse. Model koristi kontekst pet riječi s obje strane ciljne riječi *window=5* te svaka riječ koja se pojavljuje barem jednom ulazi u mode *min_count=1* i koristi se paralelna obrada s četiri jezgre procesora *workers=4*. Zatim se model trenira kroz 10 epoha i koristi se broj primjera iz korpusa *corpus_count* za definiranje veličine treninga.

```
from gensim.models import Word2Vec, Doc2Vec
from sklearn.preprocessing import StandardScaler

sentences = [desc.split() for desc in books_df['processed_description']]
word2vec_model = Word2Vec(sentences, vector_size=100, window=5, min_count=1,
workers=4)
word2vec_model.train(sentences, total_examples=word2vec_model.corpus_count,
epochs=10)

def get_avg_word2vec(text, model):
    words = text.split()
    word_vectors = [model.wv[word] for word in words if word in model.wv]
    if not word_vectors:
        return np.zeros(model.vector_size)
    avg_vector = np.mean(word_vectors, axis=0)
    return avg_vector

X_word2vec = np.array([get_avg_word2vec(desc, word2vec_model) for desc in
books_df['processed_description']])

scaler = StandardScaler()
X_word2vec_scaled = scaler.fit_transform(X_word2vec)
```

Kod 6 Korištenje metode Word2Vec

U Kodu 6 smo definirali i funkciju koja izračunava prosječni vektor za zadani tekst *get_avg_word2vec*. Pomoću metode *split()* dijelimo tekst u riječi, zatim uzimamo vektore samo za riječi koje su prisutne u *Word2Vec* modelu i spremamo to u listu *word_vectors*. Ako u listi

nema riječi vraćamo nulti vektor, inače izračunavamo prosjek svih vektora pomoću metode *mean()* i spremamo to u varijablu *avg_vector*. Sljedeće je primjenjivanje metode *get_avg_word2vec()* na sve opise knjiga i dobijemo prosječne vektore za svaki opis. Nakon što su vektori kreirani za svaki opis, standardiziraju se korištenjem *StandardScaler* funkcije iz *sklearn* biblioteke. Ovaj korak osigurava da svi vektori imaju srednju vrijednost nula i standardnu devijaciju jedan, što poboljšava performanse modela.

Word2Vec model omogućuje hvatanje semantičkih odnosa između riječi u tekstu, čime se poboljšava reprezentacija teksta u obliku vektora. Takva vektorizacija koristi se kao ulaz u modele strojnog učenja za bolje klasifikacijske rezultate.

```
from gensim.models.doc2vec import TaggedDocument

tagged_data = [TaggedDocument(words=desc.split(), tags=[str(i)]) for i, desc
in enumerate(books_df['processed_description'])]
doc2vec_model = Doc2Vec(tagged_data, vector_size=100, window=5, min_count=1,
workers=4, epochs=10)

X_doc2vec = np.array([doc2vec_model.infer_vector(desc.split()) for desc in
books_df['processed_description']])
```

Kod 7 Korištenje metode Doc2Vec

Slijedeće što koristimo je *Doc2Vec* model, koji je proširenje Word2Vec modela, za vektorizaciju cijelih dokumenata, u ovom slučaju opisa knjiga. U Kodu 7 prvo se svaki opis transformira u "označeni dokument" koristeći funkciju *TaggedDocument*, gdje *words* predstavlja riječi u tekstu, a *tags* dodjeljuje jedinstveni ID svakom dokumentu. Nakon toga, instanciramo *Doc2Vec* model sa definiranom dimenzijom vektora *vector_size=100*, postavlja se veličina prozora unutar kojeg model gleda kontekst riječi *window=5*, ignoriraju se riječi koje se pojavljuju manje od jedanput *min_count=1*, postavljen je broj procesorskih jezgara koje će se koristiti prilikom treniranja *workers=4* i broj iteracija kroz cijeli korpus podataka *epochs=10*.

Model se trenira na podacima *tagged_data*, gdje svaki dokument (opis knjige) postaje dio procesa treniranja kako bi se generirali semantički reprezentativni vektori za cijele rečenice. Nakon treniranja modela, koristimo metodu *infer_vector()* za dobivanje vektorske reprezentacije svakog opisa knjige. Ovaj proces omogućava konverziju cijelog dokumenta u numerički vektor koji može biti korišten u modelima strojnog učenja. Kreirani vektori (X_doc2vec) se zatim pretvaraju u *NumPy* niz *(np.array)* za lakšu obradu i dalje mogu biti korišteni u treniranju modela.

5.3 Kodiranje oznaka (žanrova)

Kodiranje žanrova ili oznaka u numerički format nužno je za rad s modelima strojnog učenja, posebno u kontekstu višestruke klasifikacije oznaka. U ovom kodu koristi se *MultiLabelBinarizer* iz biblioteke *scikit-learn* kako bi se pretvorili tekstualni žanrovi knjiga u binarne vektore. Svaki vektor sadrži 1 za prisutne žanrove i 0 za one koji nisu prisutni.

Primjerice, ako su za knjigu dodijeljeni žanrovi *['Fiction', 'Fantasy']*, binarni vektor će imati 1 na pozicijama za te žanrove, a 0 za sve ostale.

Kod 8 prikazuje primjenu *MultiLabelBinarizer*, gdje prvi korak uključuje pretvaranje žanrova u binarni format, gdje svaki žanr postaje vlastita kolona, a za svaku knjigu bilježi se prisutnost ili odsutnost tog žanra.

```
from sklearn.preprocessing import MultiLabelBinarizer

mlb = MultiLabelBinarizer()
y = mlb.fit_transform(books_df['genres'])

min_samples = 10000

genre_counts = np.sum(y, axis=0)

genres_to_keep = np.where(genre_counts >= min_samples)[0]

y_filtered = y[:, genres_to_keep]
X_filtered = X_tfidf

mlb_filtered = MultiLabelBinarizer(classes=mlb.classes_[genres_to_keep])
y_filtered = mlb_filtered.fit_transform(books_df['genres'])
```

Kod 8 Kodiranje oznaka – MultiLabelBinarizer

Definiran je minimalan broj pojavljivanja za svaki žanr (*min_samples = n*) kako bi se eliminirali žanrovi s premalim brojem uzoraka. Ovo je važno kako bi se model usredotočio na žanrove s dovoljno podataka za učinkovito učenje. Nakon što su odabrani relevantni žanrovi, podaci se filtriraju kako bi se zadržali samo žanrovi s dovoljnim brojem uzoraka. Time se smanjuje kompleksnost problema i fokusira na najzastupljenije oznake. Na kraju, stvara se novi *MultiLabelBinarizer* koji koristi filtrirane žanrove, čime se osigurava konzistentnost između trening i test skupova. Ovaj proces omogućuje učinkovito upravljanje velikim brojem žanrova, smanjuje rizik od overfittinga na rijetke oznake i poboljšava performanse modela strojnog učenja.

5.4 Balansirana podjela podataka na podatke za treniranje i za testiranje

Balansiranje podataka ključno je u klasifikacijskim problemima, osobito u višestrukim oznakama, gdje neki žanrovi mogu biti znatno manje zastupljeni od drugih. Neravnoteža u podacima može dovesti do toga da model favorizira češće oznake, što rezultira lošijom točnošću na rjeđim klasama. Kako bi se to izbjeglo, ovaj rad koristi *Multilabel Stratified KFold*, tehniku koja osigurava da su proporcije svake oznake konzistentne kroz sve sklopove podataka, zadržavajući ravnotežu između trening i test setova.

Umjesto nasumične podjele podataka, ova tehnika dijeli podatke na način da svaka oznaka (ili kombinacija oznaka) ima približno istu distribuciju u svakom od sklopova. Ova strategija sprječava situacije u kojima bi neki sklopovi podataka imali nedovoljno primjera za određene oznake, što bi moglo negativno utjecati na performanse modela.

U Kodu 9 na modelu koji koristi TF-IDF se koristi *MultilabelStratifiedKFold* iz biblioteke *iterstrat*, koji omogućuje stratifikaciju više oznaka. Primjerice, podaci o knjigama i njihovim žanrovima podijeljeni su na pet sklopova koristeći ovu tehniku. Korištenje 5 sklopova u unakrsnoj validaciji je standardni pristup koji pruža dobru ravnotežu između računalne efikasnosti, preciznosti procjene performansi modela i stabilnosti rezultata. Ova metoda je dovoljno pouzdana za većinu zadataka strojnog učenja, a pritom ne zahtijeva previše računalnih resursa, čineći je idealnim izborom za mnoge praktične primjene.

```
from iterstrat.ml_stratifiers import MultilabelStratifiedKFold

mskf = MultilabelStratifiedKFold(n_splits=5, shuffle=True, random_state=42)
train_indices, test_indices = next(mskf.split(X_filtered, y_filtered))

X_train = X_filtered[train_indices]
X_test = X_filtered[test_indices]
y_train = y_filtered[train_indices]
y_test = y_filtered[test_indices]
```

Kod 9 Balansirana podjela podataka na trening i test set za TF-IDF

Zatim se kreiraju trening i test skupovi na temelju prethodno generiranih indeksa kroz proces višestruke stratificirane K-struke validacije. Tada *train_indices* i *test_indices* su indeksi koji su generirani pomoću funkcije MultilabelStratifiedKFold, a služe za podjelu podataka na dva skupa:

- *X_train* i *y_train* predstavljaju podatke za trening, gdje se koristi većina podataka kako bi se model trenirao.

- *X_test* i *y_test* predstavljaju podatke za testiranje, gdje se koristi manji dio podataka za evaluaciju performansi modela.

Ova metoda balansirane podijele podataka na trening i test se korištena je i na modelu koji koristi *CountVectorizer* (Kod 10), model koji koristi *Word2Vec* (Kod 11) i model koji koristi *Doc2Vec* (Kod 12).

```
train_indices, test_indices = next(mskf.split(X_count, y_filtered))

X_train_count = X_count[train_indices]
X_test_count = X_count[test_indices]
y_train_count = y_filtered[train_indices]
y_test_count = y_filtered[test_indices]
```

Kod 10 Balansirana podjela podataka na trening i test set za CountVectorizer

```
train_indices, test_indices = next(mskf.split(X_word2vec_scaled, y_filtered))

X_train_w2v = X_word2vec_scaled[train_indices]
X_test_w2v = X_word2vec_scaled[test_indices]
y_train_w2v = y_filtered[train_indices]
y_test_w2v = y_filtered[test_indices]
```

Kod 11 Balansirana podjela podataka na trening i test set za Word2Vec

```
train_indices, test_indices = next(mskf.split(X_doc2vec, y_filtered))

X_train_d2v = X_doc2vec[train_indices]
X_test_d2v = X_doc2vec[test_indices]
y_train_d2v = y_filtered[train_indices]
y_test_d2v = y_filtered[test_indices]
```

Kod 12 Balansirana podjela podataka na trening i test set za Doc2Vec

Ova metoda osigurava da su sklopovi podataka balansirani u smislu razmjera oznaka, što pomaže modelima da uče i testiraju s konzistentnim podacima. Balansiranje podataka na ovaj način značajno poboljšava generalizaciju modela, posebno kada se radi o klasifikaciji rjeđe zastupljenih žanrova. Time se omogućuje da model bude precizniji i otporniji na pogreške u klasifikaciji, čime se postižu bolje ukupne performanse.

5.5 Treniranje modela

Za treniranje modela, korišteno je nekoliko algoritama strojnog učenja, uključujući logiču regresiju, slučajnu šumu i klasifikator s glasanjem. Svi modeli implementirani su korištenjem strategije *OneVsRestClassifier*, koja omogućuje višestruku klasifikaciju oznaka.

OneVsRestClassifier je metoda za rješavanje problema klasifikacije s više oznaka (*multilabel classification*) ili više klasa *(multiclass classification*). Radi tako da trenira jedan binarni klasifikator za svaku oznaku ili klasu. Svaki klasifikator uči razlikovati jednu oznaku od svih ostalih (dakle, "jedan naspram ostalih"). U kontekstu višestruke oznake, model predviđa za svaku oznaku neovisno. *OneVsRestClassifier* dolazi iz biblioteke *scikit-learn*, popularne Python biblioteke za strojno učenje koja pruža različite modele i alate za klasifikaciju, regresiju, klasteriranje i druge zadatke strojnog učenja. Ovaj pristup je posebno učinkovit kada imamo više klasa ili oznaka koje nisu međusobno ekskluzivne, kao što su različiti žanrovi knjiga.

Logistička regresija je jednostavan, ali učinkovit model za klasifikaciju. Kod 7 prikazuje implementiranje modela koji koristi logičku regresiju. Unutar *OneVsRestClassifier* okvira, za svaku oznaku (žanr) trenira se poseban binarni klasifikator, čime se omogućava višestruka klasifikacija oznaka. Parametar *max_iter=1000* postavlja maksimalan broj iteracija koje algoritam može napraviti kako bi postigao konvergenciju. Metoda *fit()* koristi se za treniranje modela strojnog učenja na danim podacima. Konkretno, kada pozovete *fit(X_train, y_train)*, model uči o uzorcima i pravilima na temelju ulaznih podataka (*X_train*) i njihovih pripadajućih oznaka ili klasa (*y_train*). Tijekom ovog procesa model prilagođava svoje interne parametre kako bi najbolje predvidio izlazne vrijednosti na temelju ulaza. Na kraju, model je spreman za predikcije na novim podacima.

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.linear_model import LogisticRegression

model = OneVsRestClassifier(LogisticRegression(max_iter=1000))
model.fit(X_train, y_train)
```

Kod 13 Model treniran korištenjem logičke regresije koji koristi TF-IDF

Na isti način na koji je treniran model koji koristi TF-IDF vektorizaciju (Kod 13) treniran je i model koji koristi CountVectorizer tehniku za vektoriziranje teksta (Kod 14), model koji koristi Word2Vec (Kod 15) i model koji koristi Doc2Vec (Kod 16).

```
model_count = OneVsRestClassifier(LogisticRegression(max_iter=1000))
model_count.fit(X_train_count, y_train_count)
```

Kod 14 Model treniran korištenjem logičke regresije koji koristi CountVectorizer

```
model_w2v = OneVsRestClassifier(LogisticRegression(max_iter=1000))
model_w2v.fit(X_train_w2v, y_train_w2v)
```

Kod 15 Model treniran korištenjem logičke regresije koji koristi Word2Vec

```
model_d2v = OneVsRestClassifier(LogisticRegression(max_iter=1000))
model_d2v.fit(X_train_d2v, y_train_d2v)
```

Kod 16 Model treniran korištenjem logičke regresije koji koristi Doc2Vec

Sljedeći model treniran je koristeći slučajnu šumu. Slučajna šuma koristi "asambl" pristup treniranjem više stabala odlučivanja. Ovaj model trenira 100 stabala (*n_estimators=100*), od kojih svako doprinosi konačnoj odluci. Kombinirajući predikcije svakog stabla, ovaj pristup smanjuje rizik od prekomjerne prilagodbe (overfittinga) te povećava točnost klasifikacije. Svako stablo u modelu treba vrijeme za treniranje. Kako povećavate broj stabala, vrijeme treniranja se linearno povećava. Korištenje 100 stabala omogućava dobru točnost bez znatnog povećanja računalnih resursa i vremena treniranja.

```
from sklearn.ensemble import RandomForestClassifier, VotingClassifier

ensemble_model = OneVsRestClassifier(VotingClassifier(estimators=[
    ('lr', LogisticRegression(max_iter=1000)),
    ('rf', RandomForestClassifier(n_estimators=100, random_state=42)),
], voting='soft'))
```

Kod 17 Model kombiniranje logičke regresije i slučajne šume

U Kodu 17 se koristi *asambl* model kombiniranjem logističke regresije i slučajnih šuma za višestruke klasifikacije oznaka putem *OneVsRestClassifier*. Stvara ansambl modela kombiniranjem dvije različite metode klasifikacije:

- Klasifikator s glasanjem: Kombinira više različitih modela. U ovom slučaju, koristi logističku regresiju i slučajnu šumu.
- *OneVsRestClassifier*: Implementira višestruku klasifikaciju oznaka tako da trenira jedan model za svaku oznaku (žanr).

Model logističke regresije, koji koristi maksimalno 1000 iteracija (*max_iter=1000*) kako bi postigao konvergenciju. Slučajna šuma, koji koristi 100 stabala (*n_estimators=100*) i zadržava slučajnu početnu vrijednost (*random_state=42*) za replikabilnost rezultata.

Tehnika mekog glasanja kombinira predikcije oba modela tako da računa prosječne vjerojatnosti za svaku klasu. Klasa s najvišom prosječnom vjerojatnošću postaje konačna predikcija.

Kombiniranjem dva različita modela (logistička regresija i slučajna šuma), postiže se bolji ukupni rezultat modela, jer se kombiniraju njihove snage, a slabosti se međusobno kompenziraju.

5.6 Optimizacija hiperparametara

Optimizacija hiperparametara je ključna faza u razvoju modela strojnog učenja, posebno u kontekstu obrade prirodnog jezika (NLP). U ovom radu koristimo *GridSearchCV*, alat iz biblioteke *scikit-learn*, za fino podešavanje hiperparametara modela kako bi se postigle najbolje performanse.

GridSearchCV je alat iz biblioteke *scikit-learn* koji se koristi za automatizirano pretraživanje najboljih hiperparametara modela strojnog učenja. Radi tako što isprobava sve moguće kombinacije zadanih vrijednosti hiperparametara i koristi unakrsnu validaciju (*cross-validation*) kako bi procijenio performanse svake kombinacije. Na kraju, vraća model s najboljim performansama prema zadanom kriteriju, obično točnosti ili F1-mjeru. To omogućava optimizaciju modela bez potrebe za ručnim testiranjem svake kombinacije hiperparametara.

Kod 18 prikazuje implementaciju optimizacije hiperparametara za model strojnog učenja koristeći metodu *GridSearchCV*. Definirani su u kodu parametri:

- *estimator__C*: Ovaj parametar kontrolira organizaciju u modelu logističke regresije. Niže vrijednosti C impliciraju jaču organizaciju, dok veće vrijednosti omogućuju modelu da se bolje prilagodi podacima. Testiraju se četiri različite vrijednosti: 0.1, 1, 10, i 100.
- *estimator__penalty*: Ovaj parametar određuje vrstu organizacije. *l2* organizacija je standardna i koristi se za smanjenje varijabilnosti modela.
- *estimator__solver*: Ovdje se specificira algoritam koji se koristi za optimizaciju modela logističke regresije. *lbfgs* je algoritam koji se često koristi zbog svoje efikasnosti za srednje velike skupove podataka.

GridSearchCV omogućava pretraživanje hiperparametara kroz zadani raspon vrijednosti i pronalazi najbolju kombinaciju koristeći unakrsnu validaciju. *One-vs-Rest* klasifikator pretvara problem višeklasne klasifikacije u više binarnih klasifikacija, pri čemu se za svaki razred trenira jedan model. Logistička regresija je bazni klasifikator s maksimalnim brojem iteracija postavljenim na 1000. Prosljeđuju se hiperparametri koje treba testirati *param_grid=parameters*. Koristi se petostruka unakrsna validacija (cross-validation) za procjenu performansi modela *cv=5*. Parametar *verbose=1* omogućava prikazivanje poruka o napretku prilikom pretrage hiperparametara, a parametar *n_jobs=-1* omogućava korištenje svih dostupnih procesorskih jezgara za paralelnu obradu, čime se ubrzava proces optimizacije. Ova metoda trenira model na trening setu koristeći zadane hiperparametre. Svojstvo GridSearchCV-a *best_estimator_* vraća najbolji model prema kriteriju unakrsne validacije.

```
from sklearn.model_selection import GridSearchCV

parameters = {
    'estimator__C': [0.1, 1, 10, 100],
    'estimator__penalty': ['l2'],
    'estimator__solver': ['lbfgs']
}
grid_search =
GridSearchCV(OneVsRestClassifier(LogisticRegression(max_iter=1000)),
param_grid=parameters, cv=5, verbose=1, n_jobs=-1)
grid_search.fit(X_train, y_train)
best_model = grid_search.best_estimator_
```

Kod 18 Podešavanje hiperparametara modela koristeći GridSearchCV

Na kraju implementirani su različiti modeli strojnog učenja s naglaskom na višestruku klasifikaciju oznaka žanrova knjiga. Primijenjene su metode vektorizacije kao što su TF-IDF, CountVectorizer, Word2Vec i Doc2Vec, dok su trenirani modeli poput logističke regresije, slučajne šume i klasifikatora s glasanjem. Korištena je 5-struka stratificirana unakrsna validacija kako bi se postigla ravnoteža u podjeli podataka i osigurala točna evaluacija modela.

6 Evaluacija modela

U ovom poglavlju prikazana je evaluacija različitih modela korištenih za klasifikaciju žanrova knjiga. Prvo se daje pregled skupa podataka korištenih za evaluaciju, a zatim se opisuje metodologija evaluacije uz korištenje odgovarajućih metrika poput točnosti, preciznosti i F1 mjere. Slijedi detaljna analiza rezultata za svaki model u odnosu na različite minimalne brojeve uzoraka po žanru. Na kraju, obrađeni su najčešći izazovi i pogreške povezani s klasifikacijom višestrukih oznaka.

6.1 Opis skupa podataka za evaluaciju

U ovom odjeljku fokus je na detaljnom opisu podataka korištenih za evaluaciju modela, s posebnim naglaskom na podatke o knjigama i njihovim žanrovima. Skup podataka sastoji se od tekstualnih opisa knjiga, žanrova, te drugih atributa poput naslova, autora i ocjena. Podaci su preuzeti iz datoteke *books.csv*. Ovaj skup podataka prošao je kroz nekoliko faza pretprocesiranja kako bi bio spreman za evaluaciju modela strojnog učenja. Opisat ćemo ključne karakteristike evaluacijskog skupa podataka, postupke filtriranja i segmentaciju, te veličine trening i test setova.

Nakon početnog čišćenja, skup podataka je sadržavao ukupno 52,478 zapisa knjiga s različitim atributima, uključujući naslove, autore, opise i žanrove. Za svrhe evaluacije, podaci su podijeljeni u skupove podataka za treniranje i testiranje koristeći Višeklasnu stratificiranu K-struku metodu, kako bi se očuvala proporcionalna zastupljenost žanrova u oba skupa. Na ovaj način, podaci su podijeljeni u omjeru 80:20, pri čemu:

- Skup za treniranje sadrži približno 40 000 primjera (knjiga).
- Skup za testiranje sadrži oko 10 000 primjera.

Kao ključni korak u pripremi podataka za evaluaciju, provedeno je detaljno pretprocesiranje. Ovaj korak obuhvatio je:

- Uklanjanje nepotpunih i nedosljednih zapisa: Redovi s praznim ili nevažećim opisima knjiga uklonjeni su kako bi se izbjegli problemi prilikom treniranja modela. Konkretno, oko 1,338 knjiga koje nisu imale opis izbačeno je iz skupa podataka.
- Čišćenje žanrova: Prazni stringovi i rijetki žanrovi uklonjeni su iz skupa podataka. Oko stotine rijetkih žanrova koji nisu imali dovoljno uzoraka uklonjeni su zbog toga što bi

njihova niska zastupljenost mogla uzrokovati neuravnoteženost u skupu podataka. Također, neodgovarajući znakovi u žanrovima, poput navodnika, zagrada i sl., očišćeni su kako bi se osiguralo pravilno prepoznavanje oznaka.

- Tekstualno čišćenje opisa: Opisi knjiga pretvoreni su u mala slova, a uklonjeni su brojevi, interpunkcija, te su izbačene stop riječi (npr. "and", "the"). Time su opisi pretvoreni u čist tekst koji se koristi za vektorizaciju u kasnijim koracima.

Kao dio pretprocesiranja podataka, odlučeno je da se iz skupa podataka izbace žanrovi koji su bili rijetko zastupljeni ili su sadržavali nekonzistentne ili prazne vrijednosti. Ovi žanrovi predstavljali su izazov za modeliranje zbog malog broja uzoraka ili nepotpunih informacija, što bi moglo negativno utjecati na performanse modela. Korišten je prvo prag od 5 000 primjera po žanru kako bi se osigurala dovoljna reprezentacija svakog žanra u skupu podataka.

Tijekom analize, identificirani su žanrovi poput praznih stringova, te specifične kategorije poput "10th Century", "Academics", "Alternate Universe", "BDSM", "Aeroplanes", i mnogi drugi, koji su bili prisutni u vrlo malom broju knjiga. Ukupno, više stotina žanrova je uklonjeno zbog nedovoljne zastupljenosti (što je minimalni broj uzoraka postavljen na veću vrijednost to je više žanrova ignorirano). Rijetki žanrovi mogli su stvoriti neuravnoteženost i otežati procesu treniranja modela. Izbacivanjem ovih kategorija, model se fokusira na glavne i dominantne žanrove, čime se osigurava veća točnost klasifikacije i bolja generalizacija prilikom predikcije.

Također, objašnjena je primjena različitih vektorizacija (*TF-IDF*, *CountVectorizer*, *Word2Vec*, *Doc2Vec*) na tekstualne opise, koje su se koristile za pretvorbu tekstualnih podataka u numeričke vektore, spremne za daljnju obradu u modelima strojnog učenja.

Na ovaj način, ovaj odjeljak osigurava sveobuhvatan pregled skupa podataka i pripremu podataka korištenih za evaluaciju modela, osiguravajući da su svi relevantni aspekti uključeni kako bi se omogućila precizna i vjerodostojna evaluacija.

6.2 Metodologija evaluacije

U ovom odjeljku fokusiramo se na metrike korištene za evaluaciju performansi modela, s posebnim naglaskom na specifičnosti višestruke klasifikacije oznaka, gdje svaka knjiga može pripadati više žanrova. Evaluacija modela temelji se na sljedećim metrikama: točnost (engl. accuracy), preciznost (engl. precision), odziv (engl. recall), F1-mjera (engl. F1-score) i Hammingov gubitak (engl. Hamming Loss).

Točnost je osnovna metrika koja mjeri postotak ispravno predviđenih oznaka u odnosu na sve predikcije. U kontekstu višestruke klasifikacije oznaka, točnost može biti manje informativna zbog neravnomjerne distribucije oznaka, što znači da model može imati visoku točnost čak i ako ignorira rijetke žanrove. Ipak, točnost daje opći pregled performansi modela.

Preciznost mjeri postotak ispravno predviđenih pozitivnih oznaka od ukupnog broja predviđenih pozitivnih oznaka. U kontekstu višestruke klasifikacije oznaka, preciznost je ključna metrika jer se fokusira na to koliko su točne predikcije modela za žanrove koje je označio kao prisutne. Visoka preciznost znači da je model pouzdan u svojim pozitivnim predikcijama.

Odziv također poznat kao osjetljivost, mjeri postotak ispravno predviđenih pozitivnih oznaka od ukupnog broja stvarno prisutnih pozitivnih oznaka. U višestrukoj klasifikaciji, visoki odziv znači da model uspješno identificira većinu stvarno prisutnih žanrova, što je važno kada je cilj minimizirati propuštanje relevantnih oznaka.

F1-mjera je harmonijska sredina preciznosti i odziva, pružajući balansiranu procjenu performansi modela kada su preciznost i odziv jednako važni. U višestrukoj klasifikaciji, F1-mjera je posebno korisna jer omogućava modelu da održava ravnotežu između identificiranja svih relevantnih žanrova (odziv) i izbjegavanja lažno pozitivnih predikcija (preciznost).

Hammingov gubitak je mjera koja pokazuje udio netočnih predikcija po oznaci. U višestrukoj klasifikaciji oznaka, Hammingov gubitak mjeri koliko često model pogrešno predviđa bilo koju od žanrovskih oznaka za svaku knjigu. Manji Hammingov gubitak označava bolju točnost, jer pokazuje da model rijetko pogrešno klasificira oznake. Formula za Hammingov gubitak je:

$$\text{Hamming Loss} = \frac{1}{n \times L} \sum_{i=1}^{N} \sum_{l=1}^{L} I(\hat{y}_{il} \neq y_{il})$$

Gdje *n* označava broj primjera, a L broj žanrova. $I(\hat{y}_{il} \neq y_{il})$ je indikator funkcija koja vraća 1 kada je predikcija pogrešna, a 0 kada je točna.

Kada radimo s višestrukom klasifikacijom, evaluacija performansi se može dodatno proširiti na mikro i makro verzije preciznosti, odziva i F1-mjere.

- Mikro verzije: Mikro verzije metričkih mjera agregiraju ukupni broj točnih i netočnih predikcija za sve oznake i tretiraju ih kao jednu klasu. Ova metoda bolje reflektira globalnu točnost modela.
- Makro verzije: Makro verzije računski tretiraju svaku klasu zasebno i uzimaju aritmetički prosjek preciznosti, odziva i F1-mjere za svaku klasu. Ova metoda daje jednak značaj rijetkim i često prisutnim klasama, što može biti korisno kada postoji neuravnotežena distribucija oznaka.

U kontekstu višestruke klasifikacije oznaka, važno je koristiti više metrika za dobivanje cjelovite slike o performansama modela. Svaka knjiga može pripadati više žanrova, stoga jednostavna točnost može biti obmanjujuća, jer model može jednostavno predviđati najčešće žanrove i tako ostvariti visoku točnost bez stvarnog razumijevanja podataka. Kombinacija preciznosti, odziva i F1-mjere omogućava evaluaciju modela s naglaskom na njegove sposobnosti u prepoznavanju i ispravnom klasificiranju svih relevantnih žanrova, što je ključno u višestrukoj klasifikaciji.

Ove metrike također pružaju bolji uvid u balans između preciznosti i odziva, omogućujući prilagodbu modela prema specifičnim potrebama problema, bilo da je važnije identificirati sve relevantne oznake ili minimizirati lažno pozitivne predikcije. Sljedeći dijelovi koda prikazuju kako su predikcije i evaluacija izvršeni za svaki od modela.

```
from sklearn.metrics import classification_report, accuracy_score,
hamming_loss

y_pred_count = model_count.predict(X_test_count)

print("Accuracy with CountVectorizer:", accuracy_score(y_test_count,
y_pred_count))
print("Hamming Loss:", hamming_loss(y_test, y_pred))
print("Classification Report with CountVectorizer:\n",
classification_report(y_test_count, y_pred_count,
target_names=filtered_genres, zero_division=0))
```

Kod 20 Predikcija i evaluacija modela koji je treniran na TF-IDF reprezentaciji tekstova

Kod 20 izvršava predikciju žanrova za testne podatke koristeći osnovni model, koji je treniran na TF-IDF reprezentaciji tekstova. Nakon predikcije, koristi se funkcija *accuracy_score* za izračunavanje točnosti modela, dok funkcija *hamming_loss()* za Hammingov gubitak mjeri udio netočnih oznaka u višelabelnim klasifikacijama. Funkcija *classification_report* generira

detaljan izvještaj koji uključuje preciznost, odziv i F1-mjeru za svaki žanr posebno. Metrike su ključne za procjenu učinkovitosti modela u prepoznavanju različitih žanrova.

```
y_pred_count = model_count.predict(X_test_count)

print("Accuracy with CountVectorizer:", accuracy_score(y_test_count,
y_pred_count))
print("Hamming Loss with CountVectorizer:", hamming_loss(y_test_count,
y_pred_count))
print("Classification Report with CountVectorizer:\n",
classification_report(y_test_count, y_pred_count,
target_names=filtered_genres, zero_division=0))
```

Kod 21 Predikcija i evaluacija modela s CountVectorizer

Koristi se sličan postupak i kod ostalih modela koji koriste druge tehnike (*CountVectorizer*, *Word2Vec*, *Doc2Vec*), predikcije i evaluacije tih modela prikazana u kodovima (Kod 21, Kod 22, Kod 23).

```
y_pred_w2v = model_w2v.predict(X_test_w2v)

print("Accuracy with Word2Vec:", accuracy_score(y_test_w2v, y_pred_w2v))
print("Hamming Loss with Word2Vec:", hamming_loss(y_test_w2v, y_pred_w2v))
print("Classification Report with Word2Vec:\n",
classification_report(y_test_w2v, y_pred_w2v, target_names=filtered_genres,
zero_division=0))
```

Kod 22 Predikcija i evaluacija modela s Word2Vec

```
y_pred_d2v = model_d2v.predict(X_test_d2v)

print("Accuracy with Doc2Vec:", accuracy_score(y_test_d2v, y_pred_d2v))
print("Hamming Loss with Doc2Vec:", hamming_loss(y_test_d2v, y_pred_d2v))
print("Classification Report with Doc2Vec:\n",
classification_report(y_test_d2v, y_pred_d2v, target_names=filtered_genres,
zero_division=0))
```

Kod 23 Predikcija i evaluacija modela s Word2Vec

```
y_pred_ensemble = ensemble_model.predict(X_test)

print("Accuracy with Ensemble Model:", accuracy_score(y_test,
y_pred_ensemble))
```

```
print("Hamming Loss with Ensemble Model:", hamming_loss(y_test,
y_pred_ensemble))
print("Classification Report with Ensemble Model:\n",
classification_report(y_test, y_pred_ensemble, target_names=filtered_genres,
zero_division=0))
```

Kod 24 Predikcija i evaluacija asambl modela

Evaluacija i predikacija modela koja je prikazana u Kodu 24 koristi ansambl model, koji kombinira predikcije iz više modela kako bi dao konačnu odluku o žanrovima. Evaluacija uključuje izračunavanje metričkih vrijednosti za cijeli ansambl model, pružajući uvid u to kako kombinacija modela poboljšava performanse u odnosu na pojedinačne modele.

Ove metode omogućuju sveobuhvatnu evaluaciju performansi modela u kontekstu klasifikacije s više oznaka, što je ključno za razumijevanje sposobnosti modela da točno i učinkovito klasificira knjige u različite žanrove.

6.3 Analiza rezultata

U ovom poglavlju analizirat ćemo performanse različitih modela korištenih za klasifikaciju žanrova knjiga. U analizi će se koristiti različite metrike, uključujući točnost, preciznost, odziv, F1 mjeru, mikro i makro verzije tih mjera te Hammingov gubitak. Ova potpoglavlja će uključivati pojedinačne tablice za svaki model te konačnu tabelu koja sažima rezultate svih modela.

Poseban fokus posvećen je utjecaju odabira minimalnog broja uzoraka po žanru (*min_samples*) na konačan broj "preživjelih" žanrova, tj. onih koji ostaju u skupu podataka nakon filtriranja. Ovo filtriranje je od vitalnog značaja jer osigurava da svi uključeni žanrovi imaju dovoljan broj uzoraka za treniranje i evaluaciju modela. Različiti pragovi za *min_samples* rezultirali su različitim brojem preostalih žanrova:

- 5000 uzoraka: preostalo je 18 žanrova.
- 8000 uzoraka: preostalo je 7 žanrova.
- 11000 uzoraka: preostalo je 5 žanrova.

Odabir ovih pragova temelji se na kompromisu između raznolikosti žanrova i kvalitete modela. Manji prag omogućava uključivanje više žanrova, što povećava raznolikost skupa podataka, ali može rezultirati nedovoljnom zastupljenošću nekih žanrova u treniranju. Veći prag, poput

11000, osigurava da su svi uključeni žanrovi dovoljno zastupljeni u skupu podataka, ali značajno smanjuje broj žanrova. Prag od 5000 uzoraka izabran je kao balans između raznolikosti žanrova i pouzdanosti modela, budući da omogućava uključivanje većeg broja žanrova bez gubitka kvalitete u treniranju modela.

Svako potpoglavlje će prikazati pojedinačne tablice za svaki model s rezultatima ovih metrika. Na kraju će biti uključena i konačna tablica koja sažima rezultate svih modela, pružajući jasan pregled njihovih performansi i omogućavajući međusobnu usporedbu modela na temelju svih korištenih metrika.

6.3.1 Rezultati modela za *min_samples = 5000*

U Tablici 3, predstavljeni su rezultati klasifikacije teksta pomoću TF-IDF vektorizacije, s minimalnim brojem uzoraka po žanru (*min_samples*) postavljenim na 5000. To znači da su samo oni žanrovi koji imaju najmanje 5000 uzoraka bili uključeni u trening i testiranje modela.

Za ovaj prag, "preživjelo" je ukupno 18 žanrova, što znači da je model treniran i evaluiran na uzorcima iz tih žanrova, dok su rijetki žanrovi s manje od 5000 uzoraka isključeni iz analize. Ovaj postupak filtriranja omogućava bolju evaluaciju modela jer se fokusira na žanrove koji su dovoljno zastupljeni u skupu podataka, čime se smanjuje mogućnost modela da pogriješi zbog nedostatka podataka za određene žanrove.

		preciznost	**odziv**	**F1-mjera**	**podržava**
1	Adult	0.65	0.20	0.31	1646
2	Adventure	0.70	0.29	0.41	1282
3	Audiobook	0.60	0.06	0.11	1455
4	Childrens	0.83	0.30	0.44	1029
5	Classics	0.74	0.29	0.42	1350
6	Contemporary	0.76	0.45	0.56	2096
7	Fantasy	0.84	0.65	0.74	2988
8	Fiction	0.77	0.89	0.83	6265
9	Historical	0.79	0.31	0.45	1269

		preciznost	odziv	F1-mjera	podržava
10	Historical Fiction	0.76	0.34	0.47	1521
11	Literature	0.72	0.24	0.36	1143
12	Mystery	0.82	0.43	0.57	1534
13	Nonfiction	0.81	0.57	0.67	1624
14	Novels	0.68	0.19	0.30	1540
15	Paranormal	0.84	0.43	0.57	1202
16	Romance	0.81	0.65	0.72	3081
17	Science Fiction	0.77	0.28	0.41	1068
18	Young Adult	0.76	0.51	0.61	2359
	micro avg	0.78	0.49	0.61	34452
	macro avg	0.76	0.39	0.50	34452

Tablica 3 Rezultati TF-IDF modela (min_samples = 5000)

Model je postigao ukupnu točnost od 0.1419, što znači da je ispravno klasificirao oko 14.19% svih primjera. Hammingov gubitak iznosi 0.12, što znači da je model pogrešno klasificirao oko 12% žanrova. Ovo je solidan rezultat za problem višestrukih oznaka, jer niži Hammingov gubitak ukazuje na manju pogrešku u klasifikaciji žanrova. Performanse modela značajno variraju među različitim žanrovima. Na primjer, žanrovi poput "Fantasy" i "Fiction" pokazuju visoke vrijednosti preciznosti (0.84 i 0.77) i F1-mjere (0.74 i 0.83), što ukazuje na to da model vrlo dobro prepoznaje i klasificira ove žanrove. S druge strane, žanrovi kao što su "Audiobook" i "Novels" imaju znatno niže rezultate, s F1-mjerom od samo 0.11 i 0.30, što sugerira da model ima poteškoće s točnim prepoznavanjem i klasifikacijom ovih žanrova. Makro prosječna F1-mjera je 0.50, dok je težinska prosječna F1-mjera 0.57, što pokazuje da model ima bolje performanse za češće žanrove, ali da generalno postoji prostor za poboljšanje u prepoznavanju manje zastupljenih ili specifičnih žanrova.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.49	0.33	0.39	1646
2	Adventure	0.51	0.38	0.43	1282

		preciznost	odziv	F1-mjera	podržava
3	Audiobook	0.36	0.19	0.25	1455
4	Childrens	0.58	0.44	0.50	1029
5	Classics	0.55	0.43	0.48	1350
6	Contemporary	0.64	0.50	0.56	2096
7	Fantasy	0.76	0.66	0.71	2988
8	Fiction	0.77	0.84	0.80	6265
9	Historical	0.59	0.42	0.49	1269
10	Historical Fiction	0.59	0.46	0.52	1521
11	Literature	0.52	0.37	0.43	1143
12	Mystery	0.66	0.51	0.58	1534
13	Nonfiction	0.73	0.62	0.67	1624
14	Novels	0.47	0.30	0.37	1540
15	Paranormal	0.66	0.57	0.61	1202
16	Romance	0.75	0.65	0.70	3081
17	Science Fiction	0.56	0.44	0.49	1068
18	Young Adult	0.66	0.56	0.60	2359
	mikro prosjek	0.66	0.55	0.60	34452
	makro prosjek	0.60	0.48	0.53	34452

Tablica 4 Rezultati CountVectorizer modela (min_samples = 5000)

U Tablici 4 prikazani su rezultati klasifikacije korištenjem CountVectorizer modela za klasifikaciju, postignuta je točnost od 0.1076, što znači da je model ispravno klasificirao oko 10.76% svih primjera. Vrijednost Hammingovog gubitka iznosi 0.1369, što znači da je oko 13.7% predviđanja pogrešno. Ovaj rezultat je malo lošiji u usporedbi s TF-IDF modelom, što sugerira da CountVectorizer generira više pogrešaka u klasifikaciji žanrova.

Žanrovi poput "Fantasy" i "Fiction" pokazuju relativno visoke vrijednosti preciznosti (0.76 i 0.77) i F1-mjere (0.71 i 0.80), što sugerira da model dobro prepoznaje i klasificira ove kategorije. S druge strane, žanrovi poput "Audiobook" i "Novels" imaju znatno niže rezultate,

s F1-mjerom od 0.25 i 0.37, što ukazuje na poteškoće modela u preciznom prepoznavanju tih žanrova. Makro prosječna F1-mjera iznosi 0.53 pokazuje da je model generalno učinkovitiji na češćim žanrovima, dok se suočava s izazovima u prepoznavanju rjeđih ili specifičnih žanrova. Ukupno gledano, CountVectorizer model pruža solidne rezultate, ali postoji prostor za poboljšanje, osobito u slučajevima žanrova s nižom učestalošću i složenijim tekstualnim obrascima.

		preciznost	**odziv**	**F1-mjera**	**podržava**
1	Adult	0.58	0.12	0.20	1646
2	Adventure	0.61	0.30	0.40	1282
3	Audiobook	0.09	0.00	0.00	1455
4	Childrens	0.67	0.30	0.42	1029
5	Classics	0.64	0.23	0.34	1350
6	Contemporary	0.69	0.43	0.53	2096
7	Fantasy	0.80	0.64	0.71	2988
8	Fiction	0.75	0.88	0.81	6265
9	Historical	0.64	0.26	0.37	1269
10	Historical Fiction	0.65	0.29	0.40	1521
11	Literature	0.61	0.22	0.32	1143
12	Mystery	0.75	0.44	0.56	1534
13	Nonfiction	0.74	0.55	0.63	1624
14	Novels	0.51	0.12	0.19	1540
15	Paranormal	0.74	0.41	0.53	1202
16	Romance	0.77	0.63	0.69	3081
17	Science Fiction	0.64	0.26	0.37	1068
18	Young Adult	0.71	0.46	0.56	2359
	micro avg	0.72	0.47	0.57	34452

	preciznost	odziv	F1-mjera	podržava
macro avg	0.64	0.36	0.45	34452

Tablica 5 Rezultati Word2Vec modela (min_samples = 5000)

U ovom izvještaju koji je prikazan u Tablicu 5 predstavljeni su rezultati klasifikacije korištenjem Word2Vec modela. Ukupna točnost modela je 0.1151, što znači da je model ispravno klasificirao oko 11.51% svih primjera, dok Hammingov gubitak iznosi 0.133, što znači da je oko 13.3% žanrova pogrešno klasificirano. Iako je točnost relativno niska, model pokazuje solidne performanse za određene žanrove. Na primjer, žanrovi poput "Fantasy" i "Fiction" postižu visoke vrijednosti preciznosti (0.80 i 0.75) i F1-mjeru (0.71 i 0.81), što sugerira da Word2Vec model dobro prepoznaje i klasificira ove žanrove. S druge strane, žanrovi poput "Audiobook" i "Novels" imaju vrlo niske rezultate, s F1-mjerom od 0.00 i 0.19, što ukazuje na poteškoće modela u prepoznavanju i klasifikaciji ovih žanrova.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.61	0.13	0.21	1646
2	Adventure	0.58	0.20	0.30	1282
3	Audiobook	0.46	0.02	0.05	1455
4	Childrens	0.61	0.16	0.25	1029
5	Classics	0.54	0.13	0.20	1350
6	Contemporary	0.67	0.32	0.43	2096
7	Fantasy	0.79	0.58	0.67	2988
8	Fiction	0.72	0.88	0.79	6265
9	Historical	0.65	0.22	0.33	1269
10	Historical Fiction	0.66	0.24	0.35	1521
11	Literature	0.49	0.14	0.21	1143
12	Mystery	0.73	0.35	0.47	1534
13	Nonfiction	0.72	0.40	0.52	1624
14	Novels	0.45	0.07	0.12	1540

		preciznost	odziv	F1-mjera	podržava
15	Paranormal	0.73	0.36	0.48	1202
16	Romance	0.76	0.54	0.63	3081
17	Science Fiction	0.61	0.16	0.25	1068
18	Young Adult	0.66	0.35	0.45	2359
	micro avg	0.70	0.40	0.51	34452
	macro avg	0.64	0.29	0.37	34452

Tablica 6 Rezultati Doc2Vec modela (min_samples = 5000)

U Tablici 6 prikazani su rezultati Doc2Vec modela pokazuju nisku točnost od 8.53%, što je najniže među analiziranim modelima. Vrijednost Hammingovog gubittka je 0.1432, što znači da je oko 14.32% predikcija modela pogrešno. To je veći gubitak nego kod drugih modela.

Iako postiže solidne rezultate za žanrove poput "Fiction" i "Fantasy", model ima značajne poteškoće u prepoznavanju manje zastupljenih žanrova poput "Audiobook" i "Novels".

Doc2Vec model fokusira se na stvaranje vektorskih reprezentacija cijelih dokumenata (tj. opisa knjiga), uzimajući u obzir kontekstualne informacije i semantičke odnose između riječi u cijelom dokumentu. Ovaj model može prepoznati skrivene obrasce u tekstu koji su povezani sa specifičnim žanrovima poput "Paranormal" ili "Novels", čak i ako te riječi nisu toliko česte u tekstu. Na taj način, Doc2Vec može naglasiti može istaknuti specifične ili rjeđe žanrove koji su semantički povezani s kontekstom opisa.

Ukupno gledano, Doc2Vec, sa svojom sposobnošću prepoznavanja dubljih semantičkih odnosa u opisu knjiga, može dovesti do naglašavanja specifičnijih žanrova koji se možda ne pojavljuju često, ali su semantički značajni, dok TF-IDF i CountVectorizer mogu favorizirati učestalije, općenitije žanrove.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.65	0.20	0.31	1646
2	Adventure	0.70	0.29	0.41	1282
3	Audiobook	0.60	0.06	0.11	1455

		preciznost	odziv	F1-mjera	podržava
4	Childrens	0.83	0.30	0.44	1029
5	Classics	0.74	0.29	0.42	1350
6	Contemporary	0.76	0.45	0.56	2096
7	Fantasy	0.84	0.65	0.74	2988
8	Fiction	0.77	0.89	0.83	6265
9	Historical	0.79	0.31	0.45	1269
10	Historical Fiction	0.76	0.34	0.47	1521
11	Literature	0.72	0.24	0.36	1143
12	Mystery	0.82	0.43	0.57	1534
13	Nonfiction	0.81	0.57	0.67	1624
14	Novels	0.68	0.19	0.30	1540
15	Paranormal	0.84	0.43	0.57	1202
16	Romance	0.81	0.65	0.72	3081
17	Science Fiction	0.77	0.28	0.41	1068
18	Young Adult	0.76	0.51	0.61	2359
	micro avg	0.78	0.49	0.61	34452
	macro avg	0.76	0.39	0.50	34452

Tablica 7 Rezultati Grid Search modela (min_samples = 5000)

Rezultati s najboljim modelom Tablica 7, odabranim putem Grid Search optimizacije, pokazuju točnost od 14.19%, što je identično rezultatima postignutim s osnovnim TF-IDF modelom. Hammingov gubitak iznosi 0.1208, što znači da je oko 12.08% svih predikcija pogrešno. Ovo je isti Hammingov gubitak kao i kod TF-IDF modela, što pokazuje da Grid Search nije smanjio ukupan broj pogrešnih predikcija.

Model se najbolje ponaša na žanrovima poput "Fantasy" i "Fiction", s visokim vrijednostima F1-mjere (0.74 i 0.83). Međutim, žanrovi poput "Audiobook" i "Novels" i dalje pokazuju niske performanse, s F1-mjerom od 0.11 i 0.30, što sugerira da optimizacija hiperparametara nije značajno poboljšala prepoznavanje tih specifičnih kategorija. Ukupno, Grid Search nije donio

značajna poboljšanja u usporedbi s osnovnim modelom, ali model održava solidne performanse za češće žanrove.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.75	0.10	0.17	1646
2	Adventure	0.78	0.17	0.28	1282
3	Audiobook	0.67	0.01	0.02	1455
4	Childrens	0.87	0.18	0.29	1029
5	Classics	0.80	0.21	0.34	1350
6	Contemporary	0.82	0.34	0.48	2096
7	Fantasy	0.86	0.63	0.72	2988
8	Fiction	0.76	0.91	0.83	6265
9	Historical	0.87	0.20	0.33	1269
10	Historical Fiction	0.85	0.22	0.35	1521
11	Literature	0.75	0.18	0.29	1143
12	Mystery	0.85	0.36	0.51	1534
13	Nonfiction	0.84	0.53	0.65	1624
14	Novels	0.71	0.13	0.22	1540
15	Paranormal	0.89	0.37	0.52	1202
16	Romance	0.83	0.62	0.71	3081
17	Science Fiction	0.86	0.21	0.34	1068
18	Young Adult	0.82	0.42	0.55	2359
	micro avg	0.80	0.44	0.57	34452
	macro avg	0.81	0.32	0.42	34452

Tablica 8 Rezultati asambl modela (min_samples = 5000)

Rezultati asambl modela (Tablica 8) pokazuju točnost od 13.38%, što je nešto niže u usporedbi s najboljim modelom iz Grid Search optimizacije. Hammingov gubitak iznosi 0.1249, što znači

da je oko 12.49% predikcija pogrešno klasificirano. Iako je ovaj rezultat blizu ostalim modelima, ansambl model ne uspijeva značajno smanjiti broj pogrešnih predikcija.

Iako asambl model postiže visoke vrijednosti preciznosti za mnoge žanrove, kao što su "Fantasy" (0.86) i "Fiction" (0.76), njegova vrijednost odziva je niska za mnoge žanrove, što rezultira nižimom F1-mjerom za određene kategorije. Na primjer, žanrovi poput "Audiobook" i "Novels" imaju vrlo niske F1-mjere (0.02 i 0.22), što ukazuje na to da model često propušta prepoznati uzorke iz tih žanrova. Makro prosječna F1-mjera od 0.42 sugerira da, iako model može biti dosta učinkovit za češće žanrove, suočava se s izazovima u konzistentnom prepoznavanju rjeđih i složenijih žanrova. Ukupno gledano, asambl model pruža solidne, ali ne značajno bolje performanse u usporedbi s jednostavnijim modelima, posebno kada je riječ o manje zastupljenim žanrovima.

Model	**Preciznost**		**Odziv**		**F1-mjera**		**Točnost**	**Hammingov gubitak**
	makro	**mikro**	**makro**	**mikro**	**makro**	**mikro**		
TF-IDF	0.76	0.78	0.39	0.49	0.50	**0.61**	**0.141**	**0.1208**
Count Vectorizer	0.60	0.66	**0.48**	**0.55**	**0.53**	0.60	0.107	0.1369
Word2Vec	0.65	0.72	0.37	0.47	0.45	0.57	0.115	0.1326
Doc2Vec	0.64	0.70	0.29	0.40	0.37	0.51	0.087	0.1432
Grid Search	0.76	0.78	0.39	0.49	0.50	**0.61**	**0.141**	**0.1208**
Ansambl	**0.81**	**0.80**	0.32	0.44	0.42	0.57	0.133	0.1249

Tablica 9 Sažetak rezultata svih modela (min_samples = 5000)

U ovoj analizi performansi različitih modela klasifikacije žanrova knjiga, uočeno je da većina modela nisu dostigli zadovoljavajuće rezultate (Tablica 9). Najveće muke imale su pristupi poput Doc2Vec modela, koji su pokazivali slabiju ukupnu točnost. Iako su modeli koji su koristili TF-IDF i Grid Search pokazali nešto bolju izvedbu, ukupno gledano nijedan model nije pokazao sposobnost dobrog prepoznavanja žanrova, što sugerira da je potencijalno

potrebno dodatno istraživanje i testing (npr. poboljšanje pre-procesiranja podataka, pregled navodnih oznaka žanrova, ili korištenje složenijih modela).

6.3.2 Rezultati modela za min_samples = 8000

U ovom poglavlju analizirat ćemo rezultate različitih modela korištenih za klasifikaciju žanrova knjiga s novim postavkom *min_samples = 8000*. Ova promjena u minimalnom broju uzoraka može značajno utjecati na performanse modela, stoga ćemo ponovno pogledati rezultate svakog modela posebno.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.62	0.21	0.31	1646
2	Contemporary	0.76	0.46	0.57	2095
3	Fantasy	0.83	0.64	0.72	2987
4	Fiction	0.77	0.88	0.82	6266
5	Nonfiction	0.80	0.58	0.67	1624
6	Romance	0.79	0.64	0.71	3081
7	Young Adult	0.76	0.51	0.61	2358
	micro avg	0.78	0.64	0.70	20057
	macro avg	0.76	0.56	0.63	20057

Tablica 10 Rezultati TF-IDF modela (min_samples = 8000)

Model TF-IDF sada pokazuje znatno poboljšanje u točnosti (0.3342). Hammingov gubitak iznosi 0.15, što je nešto veći gubitak nego kod *min_samples = 5000*. Iako je točnost porasla, Hammingov gubitak ukazuje da model ipak čini više grešaka na nivou pojedinačnih žanrova. Preciznost i odziv za različite žanrove variraju, s najboljim performansama za žanr Fantazija, dok je izlaz za Adult i dalje slabiji.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.47	0.32	0.38	1646

		preciznost	odziv	F1-mjera	podržava
2	Contemporary	0.64	0.51	0.57	2095
3	Fantasy	0.74	0.66	0.70	2987
4	Fiction	0.77	0.84	0.80	6266
5	Nonfiction	0.73	0.63	0.68	1624
6	Romance	0.73	0.64	0.68	3081
7	Young Adult	0.67	0.57	0.61	2358
	micro avg	0.72	0.65	0.68	20057
	macro avg	0.68	0.59	0.63	20057

Tablica 11 Rezultati CountVectiorizer modela (min_samples = 8000)

Točnost za model s Count Vectorizerom (0.2953) pokazuje poboljšanje u odnosu na prethodnu analizu, iako su rezultati još uvijek ispod željenih. Vrijednost Hammingovog gubitka je 0.169 ukazuje na veći postotak pogrešnih žanrova u odnosu na TF-IDF model. Najbolji žanrovi poput Fantazije i Fikcije ponovno imaju dobre rezultate prema preciznosti i odzivu.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.54	0.13	0.21	1646
2	Contemporary	0.67	0.42	0.52	2095
3	Fantasy	0.79	0.63	0.70	2987
4	Fiction	0.75	0.88	0.81	6266
5	Nonfiction	0.74	0.55	0.63	1624
6	Romance	0.76	0.62	0.68	3081
7	Young Adult	0.69	0.47	0.56	2358
	micro avg	0.74	0.62	0.67	20057
	macro avg	0.70	0.53	0.59	20057

Tablica 12 Rezultati Word2Vec modela (min_samples = 8000)

Word2Vec model zadržava sličnu razinu točnosti (0.2926) kao i u prethodnoj analizi. Hammingov gubitak je 0.168, što sugerira da model griješi nešto manje u predikcijama žanrova u odnosu na CountVectorizer, ali je i dalje ispod rezultata TF-IDF metode. Unatoč poboljšanoj točnosti, i dalje se suočava s izazovima prepoznavanja žanrova kao što su Adult i Young Adult.

		preciznost	**odziv**	**F1-mjera**	**podržava**
1	Adult	0.54	0.12	0.20	1646
2	Contemporary	0.66	0.30	0.42	2095
3	Fantasy	0.78	0.58	0.66	2987
4	Fiction	0.71	0.89	0.79	6266
5	Nonfiction	0.73	0.41	0.52	1624
6	Romance	0.74	0.53	0.62	3081
7	Young Adult	0.65	0.36	0.46	2358
	micro avg	0.71	0.56	0.63	20057
	macro avg	0.69	0.46	0.53	20057

Tablica 13 Rezultati Doc2Vec modela (min_samples = 8000)

Doc2Vec model doživljava smanjenje točnosti (0.2501) u usporedbi s drugim modelima i vrijednost Hammingovog gubitka je 0.186, što je najveći među svim modelima za ovaj skup podataka. Mnogi žanrovi i dalje nisu dobro prepoznati, a koristi se održava samo za Fikciju i Fantaziju.

		preciznost	**odziv**	**F1-mjera**	**podržava**
1	Adult	0.62	0.21	0.31	1646
2	Contemporary	0.76	0.46	0.57	2095
3	Fantasy	0.83	0.64	0.72	2987
4	Fiction	0.77	0.88	0.82	6266
5	Nonfiction	0.80	0.58	0.67	1624
6	Romance	0.79	0.64	0.71	3081

		preciznost	odziv	F1-mjera	podržava
7	Young Adult	0.76	0.51	0.61	2358
	micro avg	0.78	0.64	0.70	20057
	macro avg	0.76	0.56	0.63	20057

Tablica 14 Rezultati Grid Search modela (min_samples = 8000)

Najbolji model iz Grid Searcha pokazuje slične rezultate kao TF-IDF model, s istom točnošću 0.3342. Hammingov gubitak iznosi 0.1529, što je nešto niže u odnosu na druge modele poput CountVectorizera i Word2Vec-a, što znači da ovaj model ima manji udio pogrešnih predikcija u odnosu na ukupan broj oznaka. Ponovno se potvrđuje da su žanrovi poput Fantazije i Fikcije najsnažniji.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.62	0.21	0.31	1646
2	Adult	0.69	0.09	0.16	1646
3	Contemporary	0.80	0.34	0.47	2095
4	Fantasy	0.84	0.62	0.71	2987
5	Fiction	0.76	0.91	0.83	6266
6	Nonfiction	0.83	0.53	0.65	1624
7	Romance	0.82	0.62	0.70	3081
	micro avg	0.79	0.60	0.769	20057
	macro avg	0.79	0.50	0.58	20057

Tablica 15 Rezultati asambl modela (min_samples = 8000)

Ansambl model također pokazuje porast performansi s točnošću od 0.3180, a Hammingov gubitak iznosi 0.155, što znači da model ima 15.5% pogrešnih predikcija. Kao i prethodni modeli, najbolje se snalazi kod žanrova kao što su Fantazija i Fikcija.

Model	Preciznost		Odziv		F1-mjera		Točnost	Hammingov gubitak
	makro	mikro	makro	mikro	makro	mikro		
TF-IDF	0.76	0.78	0.56	0.64	**0.63**	**0.70**	**0.334**	**0.1529**
Count Vectorizer	0.68	0.72	**0.59**	**0.65**	**0.63**	0.68	0.295	0.1699
Word2Vec	0.71	0.74	0.53	0.62	0.59	0.67	0.293	0.1684
Doc2Vec	0.69	0.71	0.45	0.56	0.52	0.63	0.245	0.1861
Grid Search	0.76	0.78	0.56	0.64	**0.63**	**0.70**	**0.334**	**0.1529**
Asambl	**0.79**	**0.79**	0.50	0.60	0.58	0.69	0.318	0.1552

Tablica 16 Sažetak rezultata svih modela (min_samples = 8000)

Promjena parametra min_samples na 8000 dovela je do značajnog poboljšanja u performansama nekih modela (Tablica 16), osobito TF-IDF i Grid Search modela s točnošću od 0.3342. Ovo ukazuje na to da veći broj uzoraka može poboljšati prepoznavanje žanrova, jer neki žanrovi, poput Fantazije i Fikcije, pokazuju visoke razine preciznosti i odziva.

S druge strane, modeli kao što su Doc2Vec i Word2Vec nisu znatno poboljšali svoje performanse i ostali su ispod željenih vrijednosti. Ovo sugerira potrebu za dodatnim istraživanjem i eksperimentiranjem s različitim parametrima i pristupima u budućim analizama kako bi se poboljšala ukupna točnost modela.

6.3.3 Rezultati modela za min_samples = 11000

U ovoj analizi, obradit ćemo koji su modeli postigli kakvu preciznost u klasifikaciji žanrova knjiga s postavkom min_samples = 11000. Kroz rezultate različitih modela možemo primijetiti kako se performanse mijenjaju s povećanjem broja uzoraka.

		preciznost	odziv	F1-mjera	podržava
1	Fantasy	0.83	0.62	0.71	2987
2	Fiction	0.77	0.88	0.82	6266
3	Romance	0.80	0.65	0.72	3081
4	Young Adult	0.74	0.50	0.59	2358
	micro avg	0.79	0.72	0.75	14692
	macro avg	0.79	0.66	0.71	14692

Tablica 17 Rezultati TF-IDF modela (min_samples = 11000)

Model TF-IDF održava dobru točnost (0.4932). Vrijednost Hammingovog gubitka iznosi 0.1721. Manja vrijednost Hammingovog gubitka znači da je manji udio pogrešnih predikcija. Najbolje performanse ima za žanr Fantazija i Fikcija, dok Young Adult pati od niže preciznosti i odziva.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.62	0.21	0.31	1646
2	Contemporary	0.76	0.46	0.57	2095
3	Fantasy	0.83	0.64	0.72	2987
4	Fiction	0.77	0.88	0.82	6266
	micro avg	0.74	0.71	0.72	14692
	macro avg	0.72	0.67	0.69	14692

Tablica 18 Rezultati CountVectorizer modela (min_samples = 11000)

Model s CountVectorizer-om ima točnost od 0.4474. Hammingov gubitak od 19.4% označava udio pogrešno predviđenih žanrova u odnosu na sve predikcije. Manja vrijednost Hammingovog gubitka znači bolju točnost modela. U ovom slučaju, 19% pogrešnih predikcija je prihvatljivo, ali model ima prostora za poboljšanje. Iako su rezultati bolji nego u nekim drugim modelima, još uvijek su ispod željenog, posebno za žanr Young Adult.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.62	0.21	0.31	1646
2	Contemporary	0.76	0.46	0.57	2095
3	Fantasy	0.83	0.64	0.72	2987
4	Fiction	0.77	0.88	0.82	6266
	micro avg	0.75	0.71	0.73	14692
	macro avg	0.75	0.65	0.69	14692

Tablica 19 Rezultati Word2Vec modela (min_samples = 11000)

Word2Vec model postiže točnost od 0.4629. Zapaženo je poboljšanje, posebno u preciznosti za žanrove Fikcija i Fantazija, ali se i dalje suočava s izazovima u prepoznavanju Young Adult žanra. Model je pogriješio u približno 18.94% slučajeva, što ukazuje na umjerenu točnost, ali postoji potreba za smanjenjem ovog gubitka kako bi se poboljšala preciznost modela.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.62	0.21	0.31	1646
2	Contemporary	0.76	0.46	0.57	2095
3	Fantasy	0.83	0.64	0.72	2987
4	Fiction	0.77	0.88	0.82	6266
	micro avg	0.72	0.67	0.69	14692
	macro avg	0.72	0.59	0.64	14692

Tablica 20 Rezultati Doc2Vec modela (min_samples = 11000)

Doc2Vec model ima točnost od 0.4015, sličnih slabih performansi kao što su viđene ranije. Hammingov gubitak od 21% znači da model u prosjeku griješi u 21% predviđenih žanrova po uzorku. Ovaj rezultat pokazuje da postoji značajan broj netočnih predikcija i da model ne prepoznaje sve relevantne žanrove precizno. Najveća preciznost viđena je za Fikciju, dok Young Adult žanr ispod očekivanja.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.62	0.21	0.31	1646
2	Contemporary	0.76	0.46	0.57	2095
3	Fantasy	0.83	0.64	0.72	2987
4	Fiction	0.77	0.88	0.82	6266
	micro avg	0.79	0.72	0.75	14692
	macro avg	0.79	0.66	0.71	14692

Tablica 21 Rezultati Grid Search modela (min_samples = 11000)

Najbolji model iz Grid Searcha ponavlja istu točnost kao TF-IDF model (0.4932). Ovdje su rezultati slični, pružajući dobru predikciju za Fantasy i Fiction. Vrijednost Hammingovog gubitka iznosi 0.1721.

		preciznost	odziv	F1-mjera	podržava
1	Adult	0.62	0.21	0.31	1646
2	Contemporary	0.76	0.46	0.57	2095
3	Fantasy	0.83	0.64	0.72	2987
4	Fiction	0.77	0.88	0.82	6266
	micro avg	0.79	0.71	0.75	14692
	macro avg	0.81	0.63	0.70	14692

Tablica 22 Rezultati asambl modela (min_samples = 11000)

Ensemble model postiže točnost od 0.4915, s naglaskom na visoku preciznost za Fantaziju i Fikciju, ali izazovi sa Young Adult žanrom ostaju. Hammingov gubitak od 17.25% ukazuje na relativno nisku stopu grešaka.

Model	Preciznost		Odziv		F1-mjera		Točnost	Hammingov gubitak
	makro	mikro	makro	mikro	makro	mikro		
TF-IDF	0.79	**0.79**	0.66	**0.72**	**0.71**	**0.75**	**0.493**	**0.1721**
Count Vectorizer	0.72	0.74	**0.67**	0.71	0.69	0.72	0.447	0.1940
Word2Vec	0.75	0.75	0.64	0.70	0.68	0.73	0.459	0.1894
Doc2Vec	0.72	0.72	0.59	0.67	0.64	0.69	0.404	0.2117
Grid Search	0.79	**0.79**	0.66	**0.72**	**0.71**	**0.75**	**0.493**	**0.1721**
Ensemble	**0.81**	**0.79**	0.63	0.71	0.70	**0.75**	0.491	0.1725

Tablica 23 Sažetak rezultata svih modela (min_samples = 11000)

Postavka min_samples na 11000 rezultira sličnim performansama kao u prethodnom scenariju. Najbolji modeli, TF-IDF i najbolje izvedeni model iz Grid Searcha, pokazuju značajnu točnost (0.4932). Ovaj rezultat ukazuje na to da se, uz povećanje uzoraka, klasifikacija žanrova poboljšava, ali su i dalje potrebne dodatne optimizacije, naročito za žanr Young Adult, koji predstavlja izazov u točnosti i predikciji. Doc2Vec ima najmanju preciznost i najveći Hammingov gubitak, što sugerira da ovaj model ima najviše problema s preciznim klasificiranjem žanrova.

6.3.4 Zaključak analize rezultata

U zaključnom dijelu ove analize performansi modela za klasifikaciju žanrova knjiga, pregledat ćemo ključne nalaze i usporedbe između različitih modela te utjecaj različitih vrijednosti parametra *min_samples* na točnost, preciznost, odziv, F1-mjeru i Hammingov gubitak. Kroz ovu analizu, cilj nam je identificirati koji su modeli i parametri bili najuspješniji u prepoznavanju žanrova, te koje metode vektorizacije i algoritmi su pokazali najbolje performanse. Poseban naglasak bit će stavljen na ulogu veličine uzorka za svaki žanr (kroz parametar *min_samples*) te kako su različiti pristupi skaliranja podataka i vektorizacije utjecali na rezultate.

Zbirni rezultati prikazani su prikazani Tablicom 24 koja omogućava jednostavnu usporedbu performansi različitih modela i metoda vektorizacije. Redci tablice predstavljaju različite vektorizacije (TF-IDF, Count Vectorizer, Word2Vec, Doc2Vec), dok stupci obuhvaćaju rezultate za različite vrijednosti parametra min_samples (5000, 8000, 11000).

Model	Metrika		Minimalni broj uzoraka po žanru		
			5000	8000	11000
TF-IDF	Točnost		**0.141**	**0.334**	**0.493**
	Preciznost	makro	0.76	0.76	0.79
		mikro	0.78	0.78	**0.79**
	Odziv	makro	0.39	0.56	0.66
		mikro	0.49	0.64	0.72
	F1-mjera	makro	0.50	**0.63**	**0.71**
		mikro	0.61	**0.70**	**0.75**
	Hammingov gubitak		**0.1208**	**0.1529**	**0.1721**
Count Vectorizer	Točnost		0.107	0.295	0.447
	Preciznost	makro	0.60	0.68	0.72
		mikro	0.66	0.72	0.74
	Odziv	makro	**0.48**	**0.59**	**0.67**
		mikro	**0.55**	**0.65**	0.71
	F1-mjera	makro	**0.53**	**0.63**	0.69
		mikro	0.60	0.68	0.72
	Hammingov gubitak		0.1369	0.1699	0.1940
Word2Vec	Točnost		0.115	0.293	0.459
	Preciznost	makro	0.65	0.71	0.75
		mikro	0.72	0.74	0.75
	Odziv	makro	0.37	0.53	0.64
		mikro	0.47	0.62	0.70
	F1-mjera	makro	0.45	0.59	0.68
		mikro	0.57	0.67	0.73
	Hammingov gubitak		0.1326	0.1684	0.1894
Doc2Vec	Točnost		0.087	0.245	0.404
	Preciznost	makro	0.64	0.69	0.72

		mikro	0.70	0.71	0.72
	Odziv	makro	0.29	0.45	0.59
		mikro	0.40	0.56	0.67
	F1-mjera	makro	0.37	0.52	0.64
		mikro	0.51	0.63	0.69
	Hammingov gubitak		0.1432	0.1861	0.2117
Grid Search	Točnost		**0.141**	**0.334**	**0.493**
	Preciznost	makro	0.76	0.76	0.79
		mikro	0.78	0.78	0.79
	Odziv	makro	0.39	0.56	0.66
		mikro	0.49	0.64	**0.72**
	F1-mjera	makro	0.50	**0.63**	**0.71**
		mikro	**0.61**	**0.70**	**0.75**
	Hammingov gubitak		**0.1208**	**0.1529**	**0.1721**
Ansambl	Točnost		0.133	0.318	0.491
	Preciznost	makro	**0.81**	**0.79**	**0.81**
		mikro	**0.80**	**0.79**	0.79
	Odziv	makro	0.32	0.50	0.63
		mikro	0.44	0.60	0.71
	F1-mjera	makro	0.42	0.58	0.70
		mikro	0.57	0.69	**0.75**
	Hammingov gubitak		0.1249	0.1552	0.1725

Tablica 24 Performanse različitih modela vektorizacije i klasifikacije za višestruku klasifikaciju oznaka (min_samples = 5000, 8000, 11000)

Analiza različitih metoda vektorizacije teksta i algoritama za višestruku klasifikaciju oznaka žanrova knjiga pokazala je značajne razlike u performansama, ovisno o korištenim pristupima. Rezultati su pokazali da su metode vektorizacije TF-IDF i Word2Vec općenito pružile bolje performanse u usporedbi s CountVectorizer-om i Doc2Vec-om. Također, optimizacija modela putem Grid Searcha i primjena ansamblskih metoda dovela je do dodatnih poboljšanja.

- Točnost: Najbolju točnost postigli su modeli trenirani korištenjem TF-IDF metode, uz Grid Search optimizaciju (do 0.493). Ovaj rezultat je postignut korištenjem većih uzoraka (min_samples = 11000), što ukazuje na važnost dovoljnih količina podataka za uspješnu klasifikaciju.

- Hammingov gubitak: Kako raste minimalni broj uzoraka po žanru (min_samples), tako se povećava i Hammingov gubitak. To je razumljivo jer, s većim brojem žanrova u datasetu, zadatak klasifikacije postaje složeniji, te se povećava broj mogućih pogrešaka u predikciji. Na primjer, Hammingov gubitak za TF-IDF model porastao je s 0.1208 pri min_samples = 5000 na 0.1721 pri min_samples = 11000.
- Metode vektorizacije: TF-IDF se pokazao kao najefikasnija metoda vektorizacije, pružajući dosljedno bolje rezultate u kombinaciji s logističkom regresijom i ansamblskim metodama. Word2Vec je također pokazao solidne rezultate, osobito pri analizi semantičkih odnosa među riječima, ali je bio slabiji u odnosu na TF-IDF u većini metrika.
- Modeli klasifikacije: Logička regresija u kombinaciji s TF-IDF i optimiziranim hiperparametrima pružio je najstabilnije rezultate. Slučajna šuma je također pokazala dobru robusnost, posebno u ansamblskoj kombinaciji s logičkom regresijom, gdje je meko glasanje omogućio poboljšanje ukupne točnosti.
- Ansambl metode: Klasifikator s glasanjem, posebno u svojoj verziji s mekim glasanjem, pokazao se kao učinkovit način za kombiniranje prednosti različitih modela, čime se poboljšala točnost i smanjila varijabilnost u rezultatima.

Općenito, rezultati pokazuju da je za uspješnu višestruku klasifikaciju oznaka tekstualnih podataka ključno koristiti naprednije metode vektorizacije poput TF-IDF-a u kombinaciji s optimiziranim algoritmima strojnog učenja. Ansambl metode dodatno doprinose stabilnosti i točnosti rezultata, čineći ih pogodnima za složene zadatke kao što je klasifikacija žanrova knjiga.

6.4 Najčešći izazovi i pogreške

Pri provedbi analize i klasifikacije tekstualnih podataka, posebno kod višestrukih oznaka žanrova knjiga, uočeni su sljedeći izazovi i pogreške:

- Niska točnost kod rjeđih žanrova: Modeli su pokazali tendenciju ka postizanju niske točnosti za žanrove koji su manje zastupljeni u skupu podataka, kao što su "Audiobook" i "Novels". Ovi žanrovi često imaju niže F1-mjere, što ukazuje na poteškoće modela u pravilnom prepoznavanju i klasificiranju manje učestalih kategorija. Ova pojava može

biti uzrokovana neravnotežom u skupu podataka, gdje neki žanrovi imaju znatno više primjera nego drugi.

- Zanemarivanje semantičkih odnosa: Iako su Word2Vec i slične metode vektorizacije dizajnirane za hvatanje semantičkih odnosa među riječima, pokazalo se da one ponekad ne uspijevaju ispravno klasificirati određene žanrove zbog zanemarivanja konteksta i međusobnih odnosa među terminima unutar žanrova. Ovaj problem posebno je vidljiv kod žanrova koji dijele slične riječi, ali se razlikuju po kontekstu.
- Ograničene performanse Doc2Vec modela: Doc2Vec se pokazao manje učinkovit u usporedbi s drugim metodama vektorizacije, postižući niže rezultate u gotovo svim metrikama. Ovo može biti posljedica kompleksnosti algoritma i njegovih zahtjeva za većim količinama podataka ili boljom prilagodbom hiperparametara. Nedovoljno dobro učenje reprezentacija dokumenata dovelo je do slabijih rezultata, posebno kod klasifikacije rjeđih žanrova.
- Nedovoljna optimizacija hiperparametara: Iako su korištene metode poput Grid Searcha za optimizaciju hiperparametara poboljšale rezultate, proces optimizacije nije uvijek doveo do značajnog poboljšanja. To sugerira da su neki modeli možda dosegnuli svoje maksimalne kapacitete u ovom specifičnom zadatku ili da dodatni parametri zahtijevaju drugačiji pristup optimizaciji.
- Utjecaj veličine uzorka (min_samples): Analiza je pokazala da veličina minimalnog uzorka značajno utječe na performanse modela. Modeli trenirani s većim minimalnim uzorkom (min_samples = 11000) postigli su bolje rezultate, no također su pokazali slabosti kod žanrova s manjim brojem uzoraka. Ova pojava ukazuje na potrebu za balansiranjem skupa podataka kako bi se poboljšala točnost klasifikacije rjeđih žanrova.

Ovi izazovi i pogreške ukazuju na važnost pažljivog odabira metode vektorizacije, optimizacije modela, kao i ravnoteže u skupu podataka kako bi se postigli najbolji mogući rezultati u klasifikaciji tekstualnih podataka s višestrukim oznakama.

7 Zaključak

Ovaj diplomski rad istražio je klasifikaciju žanrova knjiga primjenom različitih NLP tehnika i modela strojnog učenja. Korištene su metode vektorizacije poput TF-IDF-a, Word2Vec-a i Doc2Vec-a te klasifikacijski algoritmi poput logističke regresije, slučajne šume i klasifikatora s glasanjem. Najbolje rezultate postigla je kombinacija TF-IDF vektorizacije s logističkom regresijom, posebno uz filtriranje podataka prema minimalnom broju uzoraka po klasi. Iako su naprednije vektorizacijske metode pokazale potencijal, nisu nadmašile jednostavniji TF-IDF pristup. Asambl modeli su dodatno doprinijeli robusnosti i uravnoteženosti klasifikacije. Rad potvrđuje učinkovitost NLP tehnika i strojnog učenja u klasifikaciji književnih žanrova, pri čemu izbor metode ovisi o karakteristikama dostupnih podataka.

Buduća istraživanja mogu se usmjeriti na primjenu naprednih dubokih modela poput BERT-a, rješavanje neravnoteže podataka korištenjem naprednih tehnika uzorkovanja i prilagođenih algoritama, integraciju dodatnih izvora podataka kao što su korisničke recenzije, ocjene i metapodaci radi poboljšanja preciznosti modela.

REFERENCE

[1] N. Xue, S. Bird, E. Klein i E. Loper, »Natural Language Processing with Python,« *O'Reilly Media,* svez. 17, br. 3, pp. 419-424, 2011.

[2] D. Jurafsky i J. H. Martin, Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models, 3 ur., Stanford University, 2024.

[3] A. Vidhya, »Predicting Movie Genres Using NLP,« [Mrežno]. Available: https://www.analyticsvidhya.com/blog/2019/04/predicting-movie-genres-nlp-multi-label-classification/. [Pokušaj pristupa 20 August 2024].

[4] M. Orlando, »How to build a multi-label text classification model using NLP and Machine Learning,« 6 October 2022. [Mrežno]. Available: https://medium.com/@mikeyo4800/how-to-build-a-multi-label-text-classification-model-using-nlp-and-machine-learning-2e05f72aad5f. [Pokušaj pristupa 18 August 2024].

[5] S. S. Vishnu, Y. Rohanth, B. U. G. Naik i J. Chautharia, »Extracting Genre Secrets: Using NLP and Multi-Label Classifiers to Predict Movie Genres from Plot Synopses,« *International Journal of Engineering Science and Advanced Technology (IJESAT),* svez. 24, br. 5, p. 556, 2024.

[6] F. . Z. Unal, M. . S. Guzel, E. Bostanci, K. Acici i T. Asuroglu, »Multi-Label Text Classification Based on Word2Vec and BERT,« *MDPI,* svez. 13, br. 15, p. 8665, 2023.

[7] S. P. A. Rahane, »Evolution of Natural Language Processing,« FreshGravity, 8 September 2020. [Mrežno]. Available: https://www.freshgravity.com/evolution-of-natural-language-processing/. [Pokušaj pristupa 30 August 2024].

[8] I. El Mir, S. El Kafhali i A. Haqiq, »A Hybrid Learning Approach for Text Classification Using Natural Language Processing,« *Proceedings of the 5th International Conference on Big Data and Internet of Things,* pp. 428-439, 2022.

[9] M. Ali, »Understanding Text Classification in Python,« Datacamp, November 2022. [Mrežno]. Available: Understanding Text Classification in Python. [Pokušaj pristupa 15 August 2024].

[10] K. Kowsari, »Text Classification Algorithms: A Survey,« Medium, 22 May 2019. [Mrežno]. Available: https://medium.com/text-classification-algorithms/text-classification-algorithms-a-survey-a215b7ab7e2d. [Pokušaj pristupa 18 August 2024].

[11] GeeksforGeeks, »Multiclass Classification vs Multi-label Classification,« GeeksforGeeks, 6 January 2024. [Mrežno]. Available: https://www.geeksforgeeks.org/multiclass-classification-vs-multi-label-classification/. [Pokušaj pristupa 29 August 2024].

[12] J. Chautharia, »Extracting Genre Secrets Using NLP,« ResearchGate Logo, May 2024. [Mrežno]. Available: https://www.researchgate.net/publication/380880252_Extracting_Genre_Secrets_Using_NLP_and_Multi-Label_Classifiers_to_Predict_Movie_Genres_from_Plot_Synopses. [Pokušaj pristupa 16 August 2024].

[13] A. Jha, »Vectorization Techniques in NLP,« 11 August 2023. [Mrežno]. Available: https://neptune.ai/blog/vectorization-techniques-in-nlp-guide. [Pokušaj pristupa 22 August 2024].

[14] Dremio, »Vectorization in NLP,« December 2023. [Mrežno]. Available: https://www.dremio.com/wiki/vectorization-in-nlp/. [Pokušaj pristupa 22 August 2024].

[15] N. Barla, »The Ultimate Guide to Word Embeddings,« 16 April 2024. [Mrežno]. Available: https://neptune.ai/blog/word-embeddings-guide. [Pokušaj pristupa 10 August 2024].

[16] Pianalytix, »CountVectorizer In NLP,« [Mrežno]. Available: https://pianalytix.com/countvectorizer-in-nlp/. [Pokušaj pristupa 4 August 2024].

[17] IBM, »Using CountVectorizer for NLP feature extraction,« 10 November 2023. [Mrežno]. Available: https://www.ibm.com/reference/python/countvectorizer. [Pokušaj pristupa 18 August 2024].

[18] J. Brownlee, Deep Learning for Natural Language Processing Develop Deep Learning Models for your Natural Language Problems, Machine Learning Mastery, 2017.

[19] M. R. Aman Kedia, Hands-On Python Natural Language Processing, Birmingham, UK: Packt Publishing, 2020.

[20] A. Jain, »TF-IDF in NLP (Term Frequency Inverse Document Frequency),« 4 February 2024. [Mrežno]. Available: https://medium.com/@abhishekjainindore24/tf-idf-in-nlp-term-frequency-inverse-document-frequency-e05b65932f1d. [Pokušaj pristupa 13 August 2024].

[21] Medium, »Understanding TF-IDF in NLP: A Comprehensive Guide,« 21 March 2023. [Mrežno]. Available: https://medium.com/@er.iit.pradeep09/understanding-tf-idf-in-nlp-a-comprehensive-guide-26707db0cec5. [Pokušaj pristupa 23 August 2024].

[22] Medium, »Understanding Word2Vec: A Beginner's Guide to Word Embeddings,« 23 April 2024. [Mrežno]. Available: https://medium.com/@pooja93palod/understanding-word2vec-a-beginners-guide-to-word-embeddings-6ecb893dbf61. [Pokušaj pristupa 24 August 2024].

[23] D. Karani, »Introduction to Word Embedding and Word2Vec,« 1 September 2018. [Mrežno]. Available: https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa. [Pokušaj pristupa 20 August 2024].

[24] D. Johnson, »Word Embedding i Word2Vec model s primjerom,« 16 March 2024. [Mrežno]. Available: https://www.guru99.com/hr/word-embedding-word2vec.html. [Pokušaj pristupa 23 July 2024].

[25] S. Li, »Multi-Class Text Classification with Doc2Vec & Logistic Regression,« 18 September 2018. [Mrežno]. Available: https://towardsdatascience.com/multi-class-text-classification-with-doc2vec-logistic-regression-9da9947b43f4. [Pokušaj pristupa 27 July 2024].

[26] GeeksforGeeks, »Doc2Vec in NLP,« 11 July 2023. [Mrežno]. Available: https://www.geeksforgeeks.org/doc2vec-in-nlp/. [Pokušaj pristupa 28 July 2024].

[27] G. Shperber, »A gentle introduction to Doc2Vec,« 26 July 2017. [Mrežno]. Available: https://medium.com/wisio/a-gentle-introduction-to-doc2vec-db3e8c0cce5e. [Pokušaj pristupa 20 August 2024].

[28] GENSIM, »Doc2Vec Model,« August 2021. [Mrežno]. Available: https://radimrehurek.com/gensim/auto_examples/tutorials/run_doc2vec_lee.html. [Pokušaj pristupa 4 August 2024].

[29] GeeksforGeeks, »Text Classification using Logistic Regression,« 4 March 2024. [Mrežno]. Available: https://www.geeksforgeeks.org/text-classification-using-logistic-regression/. [Pokušaj pristupa 14 August 2024].

[30] IBM, »What is random forest?,« [Mrežno]. Available: https://www.ibm.com/topics/random-forest. [Pokušaj pristupa 26 July 2024].

[31] Scikit-Learn, »RandomForestClassifier,« [Mrežno]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. [Pokušaj pristupa 26 July 2024].

[32] GeeksforGeeks, »Random Forest Classifier using Scikit-learn,« 31 January 2024. [Mrežno]. Available: https://www.geeksforgeeks.org/random-forest-classifier-using-scikit-learn/. [Pokušaj pristupa 8 August 2024].

[33] Scikit-Learn, »VotingClassifier,« [Mrežno]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html. [Pokušaj pristupa 28 July 2024].

[34] GeeksforGeeks, »Voting Classifier,« 11 October 2023. [Mrežno]. Available: https://www.geeksforgeeks.org/voting-classifier/. [Pokušaj pristupa 8 August 2024].

[35] GeeksforGeeks, »Stratified K Fold Cross Validation,« 10 January 2023. [Mrežno]. Available: https://www.geeksforgeeks.org/stratified-k-fold-cross-validation/. [Pokušaj pristupa 10 August 2024].

[36] s. learn, »CountVectorizer,« [Mrežno]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html. [Pokušaj pristupa 12 July 2024].

[37] M. G., »CountVectorizer Tips & Tricks¶,« [Mrežno]. Available: https://maartengr.github.io/KeyBERT/guides/countvectorizer.html. [Pokušaj pristupa 23 August 2024].

[38] R. H. S. Vivian Siahaan, TEXT PROCESSING AND SENTIMENT ANALYSIS USING MACHINE LEARNING AND DEEP LEARNING WITH PYTHON GUI, Balige: Independently published, 2022.

[39] GeeksforGeeks, »Using CountVectorizer to Extracting Features from Text,« 7 July 2022. [Mrežno]. Available: https://www.geeksforgeeks.org/using-countvectorizer-to-extracting-features-from-text/. [Pokušaj pristupa 16 August 2024].

POPIS SLIKA

POPIS TABLICA