ML.NET tehnologija za praktičnu primjenu modela strojnog učenja

Gajski, Benjamin

VELEUČILIŠTE SUVREMENIH INFORMACIJSKIH TEHNOLOGIJA

STRUČNI PRIJEDIPLOMSKI STUDIJ INFORMACIJSKIH TEHNOLOGIJA

Benjamin Gajski

ZAVRŠNI RAD

ML.NET TEHNOLOGIJA ZA PRAKTIČNU PRIMJENU MODELA STROJNOG UČENJA

Zagreb, listopada 2024.

Veleučilište suvremenih informacijskih tehnologija
10000 Zagreb, Ulica Vjekoslava Klaića 7

Studij:	Stručni prijediplomski studij informacijskih tehnologija smjer baze podataka i web dizajn
Student:	**Benjamin Gajski**
Matični broj:	2017090

Zadatak završnog rada

Predmet:	Uvod u umjetnu inteligenciju i strojno učenje
Naslov:	**ML.NET tehnologija za praktičnu primjenu modela strojnog učenja**
Zadatak:	Definirati osnovne pojmove korištene u radu. Demonstrirati implementaciju aplikacije razvijene u ML.NET okruženju pomoću ML.NET knjižnice i objasniti unos i predobradu podataka. Metodama klasifikacije i modela strojnog učenja naći i interpretirati dobivene rezultate.
Mentor:	Marijan Čančarević, v. pred.
Zadatak uručen kandidatu:	28.2.2024.
Rok za predaju rada:	31.10.2024.
Rad predan:	_____

Povjerenstvo:

Dragana Čulina, pred.	član predsjednik	_____
Marijan Čančarević, v. pred.	mentor	_____
Jurica Đurić, v. pred.	član	_____

SADRŽAJ

POPIS SLIKA

POPIS GRAFIKONA

POPIS TABLICA

POPIS KODOVA

1. UVOD

Strojno učenje predstavlja metodologiju analize podataka koja omogućuje automatiziranu izgradnju analitičkih modela. Korištenjem ove metode računalni sustavi stječu sposobnost usvajanja informacija iz podataka, identifikacije obrazaca te donošenja odluka uz minimalnu potrebu za ljudskom intervencijom. Ova grana umjetne inteligencije značajno unapređuje industrije poput telekomunikacija zahvaljujući sve većim količinama dostupnih podataka i naprednim tehnologijama. U medicinskoj dijagnostici koristi se za analizu medicinskih slika i predviđanje bolesti, dok u telekomunikacijskom sektoru doprinosi optimizaciji mrežnih parametara i poboljšanju filtriranja poruka.

Posebno se ističe njegova primjena u klasifikaciji podataka, bilo da se radi o binarnoj klasifikaciji za predikciju dijabetesa s pomoću medicinskih podataka ili o tekstualnoj klasifikaciji za filtriranje spam poruka. Ovi sustavi omogućuju razvoj inteligentnih aplikacija koje analiziraju velike količine podataka i donose točne odluke. Široki spektar primjena čini strojno učenje neizostavnim alatom, omogućujući inovacije i optimizaciju procesa u raznim industrijama.

.NET je fleksibilna razvojna platforma koja omogućuje izradu aplikacija za web, mobilne uređaje i desktop računala. Razvijena od strane Microsofta, platforma nudi raznolik skup alata i knjižnica za različite vrste aplikacija. Implementacija programa unutar .NET okruženja preporučeno se dizajnira principima objektno orijentiranog programiranja (OOP, engl. Object-Oriented Programming), što omogućuje modularnost, ponovnu upotrebljivost i skalabilnost koda. Automatizirani testovi su također bitna komponenta, uključujući unit testove za pojedine komponente koda, te testove za provjeru ispravne funkcionalnosti među komponentama. Sve ove karakteristike čine .NET platformu izuzetno funkcionalnim alatom, idealnim za razvoj širokog spektra aplikacija, uključujući one koje koriste strojno učenje i analitiku podataka.

ML.NET je Microsoftova knjižnica namijenjena integraciji modela strojnog učenja unutar .NET okruženja, omogućujući programerima implementaciju naprednih algoritama strojnog učenja bez potrebe za prelaskom na druge platforme. Ova knjižnica pruža alate za transformaciju podataka, treniranje modela i evaluaciju performansi, čime omogućuje stvaranje sofisticiranih modela unutar .NET ekosustava. Optimizirana za nesmetan rad i testiranje većih podatkovnih skupova, ML.NET nudi rješenja za širok spektar problema analize podataka i

prediktivnih modeliranja. Povezanost ML.NET-a s .NET platformom omogućuje developerima da iskoriste postojeća znanja, čime se ubrzava razvoj i implementacija složenih modela strojnog učenja.

Cilj ovog rada je istraživanje praktične mogućnosti implementacije ML.NET knjižnice unutar .NET okruženja za rješavanje problema binarne i tekstualne klasifikacije. Planiran je razvoj aplikacije koja će omogućiti korisnicima unos i analizu podataka, odabir algoritama za klasifikaciju te vizualizaciju rezultata koristeći grafičko sučelje putem Windows Forms tehnologije.

Dizajn aplikacije pruža edukativnu vrijednost, omogućujući bolje razumijevanje modela strojnog učenja kroz matematičke izračune, tekstualna objašnjenja i grafičke prikaze. Posebna pažnja bit će posvećena predobradi podataka, implementaciji čišćenja, normalizacije i balansiranje klasa, kako bi se osigurala kvaliteta ulaznih podataka za modele binarne klasifikacije (npr. predikcija dijabetesa) i tekstualne klasifikacije (npr. filtriranje spam poruka).

2. PRIMJENA STROJNOG UČENJA U KLASIFIKACIJI PODATKOVNIH SKUPOVA

U ovom poglavlju analizirane su metode i pristupi primjene strojnog učenja u klasifikaciji binarnih i tekstualnih skupova podataka.

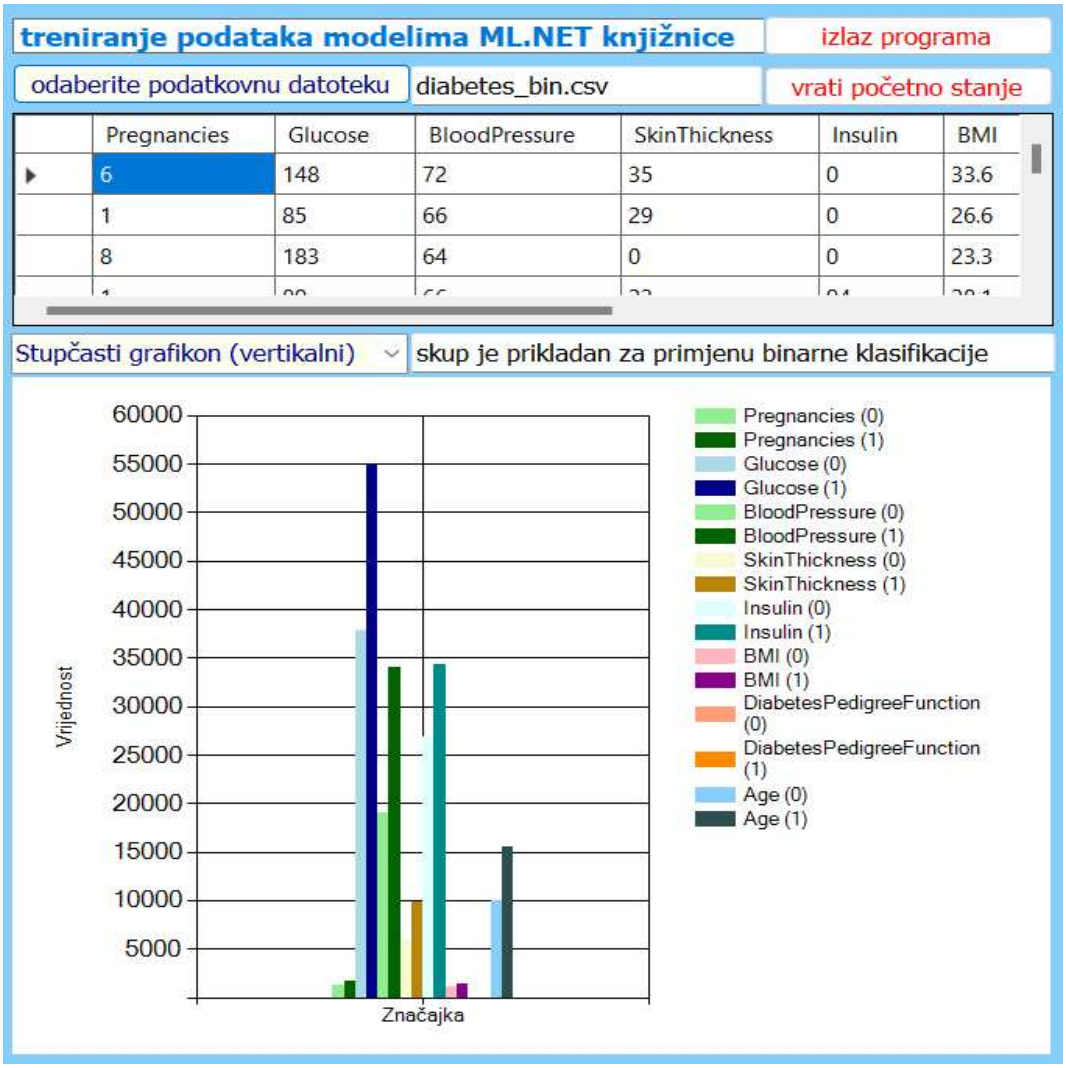
2.1. Klasifikacija binarnog podatkovnog skupa

Ključna mogućnost obrade podataka je implementacija algoritama strojnog učenja za raspodjelu značajnih stavki u odgovarajuće skupove, koristeći prethodno označene podatke. Ovaj postupak, poznat kao binarna klasifikacija, omogućava modelima da uče iz označenih podataka i predviđaju nove podatke unutar dvije međusobno isključive kategorije. Binarna klasifikacija je temeljna za automatizaciju zadataka koji bi inače zahtijevali ljudsku intervenciju, kao što su zdravstvena dijagnostika ili detekcija anomalija. Ovakav pristup povećava učinkovitost i značajno smanjuje vrijeme potrebno za obavljanje složenih zadataka.

Za primjer treniranja podataka s problematikom binarne klasifikacije korišten je javno dostupan podatkovni skup podataka s platforme otvorenog karaktera Kaggle.com, s ciljem objavljivanja i objedinjavanja podataka te analiza modela strojnog učenja.

Podatkovni skup: UCI Machine Learning, Pima Indians Diabetes Database,
https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database/
(pristupljeno 5. travnja 2024.).

Skup podataka dostupan je iz internacionalnog instituta za dijabetes, probavne i bubrežne bolesti s intencijom razvijanja metoda za predviđanje ima li subjekt dijabetes na osnovu značajki. Navedeni skup sastoji se od prediktivnih, odnosno nezavisnih varijabli: broj trudnoća, nivo glukoze u krvi, krvni pritisak, debljina kožnog nabora, nivo inzulina, indeks tjelesne mase, funkcija obiteljske povijesti dijabetesa, starost, te od jedne zavisne varijable definirane kao ishod. Prikaz podatkovnog skupa nakon unosa podataka prikazan je grafikonom Grafikon 1.



Grafikon 1. Binarni podatkovni skup (autorski rad)

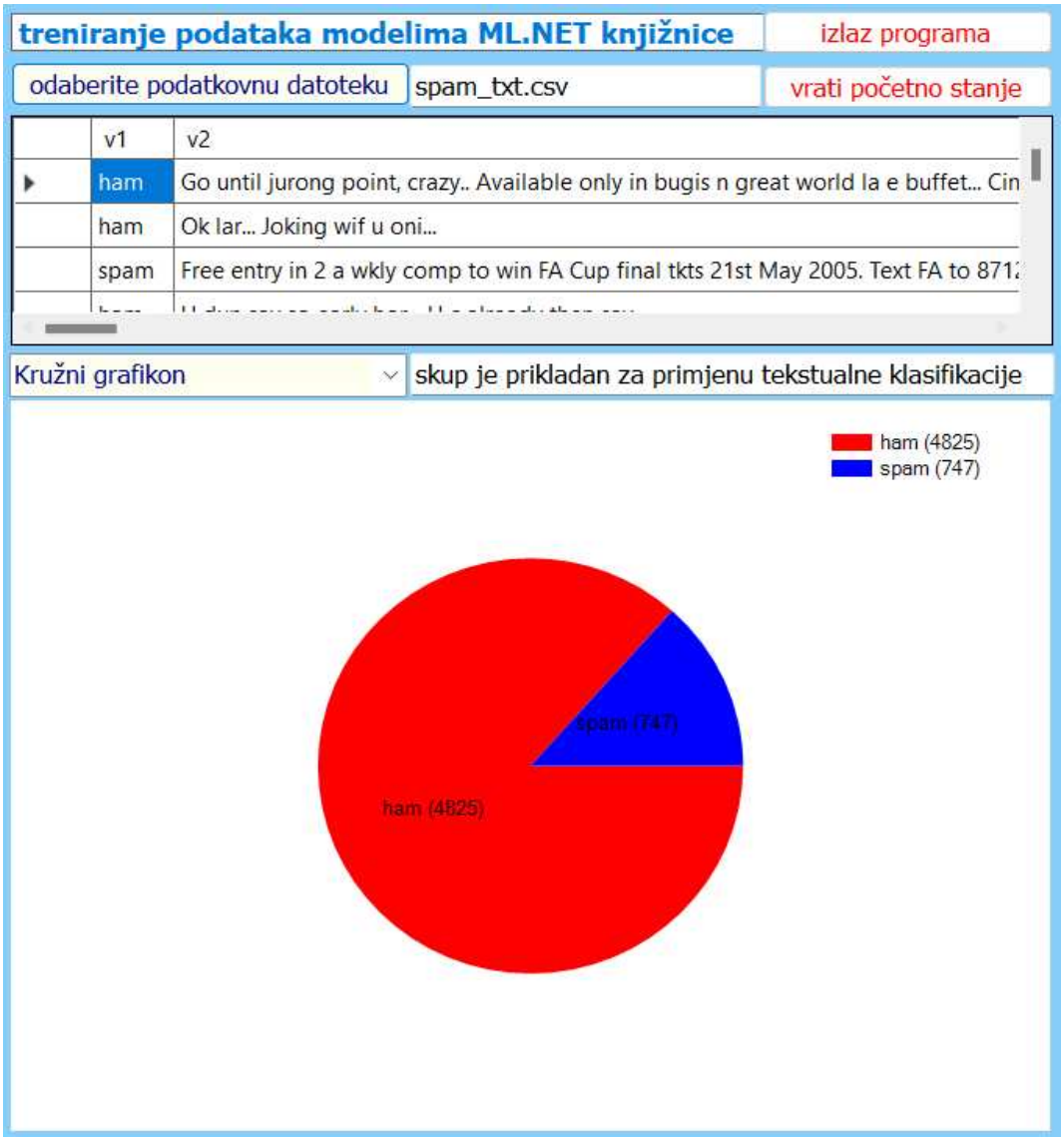
2.2. Klasifikacija tekstualnog podatkovnog skupa

Tekstualna klasifikacija usmjerena je na rješavanje problema s tekstualnim sadržajem, poput kategorizacije elektronske pošte. Ova vrsta klasifikacije koristi se za prepoznavanje obrazaca i značajki unutar tekstualnih podataka kako bi se poruke automatski svrstale u određene kategorije. Implementacijom algoritama strojnog učenja, tekstualna klasifikacija omogućava modelima da precizno i učinkovito filtriraju neželjene poruke (engl. spam) od željenih (engl. ham).

Za primjer problematike tekstualne klasifikacije također je korišten javno dostupan podatkovni skup podataka s platforme Kaggle.com.

Podatkovni skup: UCI Machine Learning, SMS Spam Collection DataSet, https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset/ (pristupljeno 5. travnja 2024.).

Skup podataka kreiran je s različitih inozemnih izvora s vizijom kreiranje metoda za filtriranje poruka u SMS (engl. Short Message Service) komunikaciji. Skup se sastoji od dvije kolone podataka koje označavaju je li poruka željenog sadržaja (ham) ili neželjenog (spam) te teksta poruke. Prikaz podatkovnog skupa nakon unosa podataka prikazan je grafikonom Grafikon 2.



Grafikon 2. Tekstualni podatkovni skup (autorski rad)

2.3. ML.NET knjižnica

ML.NET je Microsoftova knjižnica razvijena za problematiku strojnog učenja koja omogućava sistemsku integraciju modela strojnog učenja unutar .NET okruženja prilikom izrade aplikacije, kroz korake:

1. Transformacija podataka: kritičan korak u pripremi podataka za treniranje strojnog učenja predefiniranim transformacija za čišćenje, normalizaciju i pripremu podataka.
2. Treniranje modela: podržano je treniranje podataka različitim algoritmima strojnog učenja koji se lako implementiraju, kombiniraju ili zamjenjuju sukladno problemima strojnog učenja.
3. Provjera modela: evaluacija algoritama i performansi modela omogućena je kroz ugrađene metode za procjenu.

Ove ključni koraci omogućuju implementaciju strojnog učenja unutar .NET aplikacija, pružajući rješenja za širok spektar problema analize podataka i prediktivnih modeliranja. Demonstrativno, u implementaciji programa koristi se alat za unit testing u koraku transformacije podataka, što olakšava testiranje prije samog debugiranja.

3. ALGORITMI I METODE EVALUACIJE PERFORMANSI

Sljedeće poglavlje posvećeno je detaljnoj analizi metoda i pristupa primjene strojnog učenja u klasifikaciji različitih vrsta podatkovnih skupova, s posebnim naglaskom na binarne i tekstualne skupove podataka. Posebna pažnja usmjerena je na odabir odgovarajućih metrika za evaluaciju performansi modela, kao što su točnost, preciznost i odaziv, koje omogućuju preciznu i objektivnu procjenu učinkovitosti primijenjenih algoritama.

3.1. Algoritmi i metode evaluacije binarne klasifikacije

Za treniranje skupa podataka binarne klasifikacije izabran je model logističke regresije, koji je idealan za problematiku analize predviđanja dijabetesa putem modeliranja vjerojatnosti da određeni podatak pripada jednoj od dvije klase. Model je jednostavan za implementaciju, efikasan, te ga je lako razumjeti u kontekstu podatkovnog skupa, što omogućava stručnjacima da lako utvrde kako različiti faktori, poput nivoa glukoze u krvi i krvnog pritiska, utječu na rizik od dijabetesa.

Prosječni perceptron, kao drugi model za usporedbu, jednostavan je neuronski model koji kombinira linearnu funkciju težine i karakteristika za donošenje odluka. Odlikuje ga brzina, jednostavno kalibracije parametara te robusnost u smislu generalizacije na testnim podacima, što je osobito bitno u analizi problema s većim varijacijama u podacima.

Za evaluaciju performansi modela izabrani su sljedeći pristupi: evaluacija performansi predikcija i krivulja radne karakteristike (ROC, engl. Receiver Operating Characteristic). Evaluacija performansi predikcija omogućava mjerenje točnosti i preciznosti modela, dok krivulja radne karakteristike (ROC) i površina ispod krivulje (AUC, engl. Area Under the Curve) daju cjelovit uvid u performanse modela kroz sve pragove klasifikacije.

3.1.1. Logistička regresija

Logistička regresija je model korišten za binarnu klasifikaciju koji se koristi za predviđanje pripadnosti značajke jednoj od dvije moguće klase. Temelji se na logističkoj funkciji koja mapira stvarne vrijednosti u interval između 0 i 1, što se u kontekstu projekta koristi za predviđanje vjerojatnosti da li osoba ima dijabetes na temelju ulaznih značajki kao što su indeks tjelesne mase ili razina glukoze u krvi. Logistička funkcija je definirana izrazom:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

U jednadžbi e označava osnovu prirodnog logaritma, z predstavlja linearnu kombinaciju ulaznih značajki i njihovih koeficijenata, te se može zapisati kao:

$$z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots \beta_n X_n$$

U gornjoj jednadžbi β_i označavaju koeficijente modela, dok X_i predstavljaju ulazne značajke. Za potrebe treniranja i optimiziranja modela koristi se metoda maksimalne vrijednosti, kojom se omogućava prilagodba modela podacima. Funkcija vjerojatnosti modela je definirana jednadžbom:

$$L(\beta) = \prod_{i=1}^{m} P(y^{(i)} \mid X^{(i)}; \beta)$$

U jednadžbi $L(\beta)$ predstavlja funkciju vjerojatnosti za skup koeficijenata β koji se koriste za treniranje modela strojnog učenja, m je broj primjera u skupu podataka, $y^{(i)}$ označava stvarne oznake za i-ti primjer, dok $X^{(i)}$ predstavlja ulazne značajke za i-ti primjer.

U svrhu sprječavanja prenaučenosti modela koristi se princip regularizacije koji dodaje kaznu za velike vrijednosti koeficijenata u funkciji gubitka. Regularizacija smanjuje složenost modela i pomaže u izbjegavanju prekomjernog prilagođavanja specifičnostima skupa za obuku. Izraz kazne za ograničenje vrijednosti koeficijenata modela:

$$Kazna = \lambda \sum_{i=1}^{n} \beta i^2$$

U jednadžbi λ predstavlja regularizacijski parametar koji kontrolira jačinu kazne, dok je n ukupan broj koeficijenata β_i koje odgovaraju ulaznim značajkama X_i. Veće vrijednosti povećavaju kaznu za velike koeficijente i smanjuju složenost modela.
Izraz $\sum_{i=1}^{n} \beta i^2$ predstavlja pojam poznat kao L2 regularizacija.

3.1.2. Perceptron s prosjekom

Perceptron s prosjekom predviđa pripadnost značajke jednoj od dvije moguće klase linearnom separacijom podataka, gdje su težine iterativno ažurirane na temelju pogrešaka u predikcijama, a srednje vrijednosti težina koriste se kroz sve iteracije kako bi se poboljšala točnost i konzistentnost modela.

Matematički opis perceptrona kao linearne kombinacije ulaznih značajki i težina:

$$z = \beta_0 + \lambda \sum_{i=1}^{n} \beta_i X_i$$

Vrijednosti X_i predstavljaju ulazne značajke, β_0 je oznaka pristranosti (bias), a β_i su težine značajki.

Sam postupak treniranja perceptrona provodi se iterativno, pri čemu se težine ažuriraju na temelju pravila:

$$\beta_j := \beta_j + \eta(y - \hat{y})X_j$$

Stopom učenja η određuje se brzina kojom su prilagođavaju težine u procesu učenja. Težina β_j odnosi se na j-tu značajku, dok X_j označava vrijednost te značajke. Stvarna oznaka y predstavlja postojeću klasu u podacima, dok predviđena oznaka $\hat{y}$ predstavlja procjenu modela na temelju izračunatog rezultata z:

$$\hat{y} = \begin{cases} 1 \; ako \; z \geq 0 \\ 0 \; ako \; z < 0 \end{cases}$$

Za dodatno poboljšanje performansi modela koriste se konačne težine kao prosjek težina $\beta_j^{(t)}$ kroz sve iteracije, kako bi se smanjila varijanca kod fluktuacija težina u ranim fazama učenja i otklonio problem oslanjanja na pojedinačne iteracije koje mogu biti pogođene lokalnim pogreškama, osiguravajući konzistentnost:

$$\beta = \frac{1}{T} \sum_{i=1}^{T} \beta_j^{(t)}$$

U gornjoj jednadžbi, T je ukupan broj iteracija kroz koje je model treniran, dok $\frac{1}{T}$ predstavlja prosječan faktor osiguravajući da se uzimaju u obzir sve iteracije podjednako.

Model se može dodatno podesiti Plattovom kalibracijom koja poboljšava točnost predikcija. Teoretski, logistička regresija se koristi za prilagodbu skoriranih vrijednosti z u vjerojatnost da ulaz pripada pozitivnoj klasi:

$$P(y = 1|z) = \frac{1}{1 + e^{-(Az+B)}}$$

$P(y = 1|z)$ predstavlja vjerojatnost da ulaz pripada pozitivnoj klasi, a parametri A i B logističke regresije uče se tijekom kalibracije.

3.1.3. Evaluacija performansi predikcija

Primjenom evaluacije performansi predikcija prikazana je raspodjela dodijeljenih vjerojatnosti uzorcima za određenu klasu.

- Preciznost pozitivnih predikcija (PPV, engl. Positive Predictive Value): omjer stvarno pozitivnih slučajeva među svim predikcijama označenima kao pozitivni, što ukazuje na to da model rijetko daje lažne pozitivne rezultate.
- Odaziv pozitivnih predikcija (TPR, engl. True Positive Rate): pokazatelj stvarno pozitivnih slučajeva koji su ispravno potvrđeni. Visok odaziv predstavlja sposobnost modela da identificira veći broj stvarno pozitivnih slučajeva.
- Preciznost negativnih predikcija (NPV, engl. Negative Predictive Value): omjer stvarno negativnih slučajeva među svim slučajevima koje model predviđa kao negativne, što ukazuje na to da model rijetko daje lažno negativne rezultate, odnosno rijetko označava stvarno pozitivne slučajeve kao negativne.
- Odaziv negativnih predikcija (TNR, engl. True Negative Rate): pokazatelj stvarno negativnih slučajeva koji su ispravno identificirani kao negativni. Visok odaziv predstavlja sposobnost modela da rijetko daje lažno pozitivne rezultate tako što učinkovito identificira negativne ishode.

3.1.4. Krivulja radne karakteristike

Grafičkim prikazom krivulje radne karakteristike (ROC) ocjenjuje se performansa binarnih klasifikacijskih modela analiziranjem odnosa između odaziva pozitivnih slučajeva (TPR) i stope lažnih pozitivnih slučajeva (FPR, engl. False Positive Rate) pri različitim pragovima odlučivanja. FPR pokazuje koliko često model pogrešno identificira negativne slučajeve kao pozitivne. Površina ispod ROC krivulje (AUC) pruža jedinstvenu mjeru performansi modela, s vrijednostima između 0 i 1, gdje veća vrijednost AUC-a ukazuje na bolju sposobnost modela u razlikovanju između klasa.

3.2. Algoritmi i metode evaluacije tekstualne klasifikacije

Za treniranje skupa podataka u problematici tekstualne klasifikacije izabran je model stablo odluke, prikladan za analizu predviđanja putem modeliranja vjerojatnosti da određeni podatak pripada jednoj do dvije ponuđene klase. Stabla odluke učinkovito modeliraju složene interakcije među značajkama, što je ključno za identifikaciju utjecaja različitih faktora, poput specifičnih fraza i riječi, na vjerojatnost klasifikacije poruke kao neželjenog sadržaja (spam).

Naivni Bayesov klasifikator, koji koristi Bayesov teorem za donošenje odluka, izabran je zbog brzine, jednostavnosti kalibracije parametara i robusnosti, čime se postiže generalizacija s testni podacima kada postoji velika raznolikost varijacija u podacima. Model teži što točnijoj analizi u slučaju kada postoji velika raznolikost u načinu na koji su poruke semantički sastavljene.

Za evaluaciju performansi modela izabrani su sljedeći pristupi: evaluacija matrice zabune i kumulativni graf dobitka. Evaluacija matrice zabune omogućava detaljan pregled točnih i netočnih predikcija, dok kumulativni graf dobitka pruža uvid u ukupni učinak modela kroz sve razine klasifikacije.

3.2.1. Stablo odluka

Stablo odluke se temelji na hijerarhijskoj strukturi za klasifikaciju ulaznih podataka. Čvorovi predstavljaju odluke o pojedinim značajkama, a svaka grana ishod te odluke. Klasifikacija je predstavljena listovima stabla te se sam proces izgradnje stabla podešava kroz sljedeće postupke:

1. Odabirom značajke za podjelu: algoritam za svaki čvor bira značajku koja najbolje razdvaja podatke u odvojene klase. Postupak se temelji na mjerama:
   - Gini indeks: indeks korišten za mjerenje stupnja čistoće čvora. Kad je indeks 0, svi elementi čvora pripadaju istoj klasi, dok indeks 1 označava da su podaci ravnomjerno raspoređeni među klasama, definicija izraza Gini indeksa:

$$Gini(S) = 1 - \sum_{i=1}^{C} {p_i}^2$$

     U jednadžbi S označuje skup podataka, C ukupan broj klasa, a p_i je udio podataka koji pripadaju klasi i.

- Entropija: mjera kaotičnosti ili nepredvidivosti u skupu podataka. U kontekstu strojnog učenja, entropija se koristi za procjenu homogenosti skupa podataka. Niža entropija ukazuje na homogeniji skup podataka, dok viša entropija ukazuje na veći stupanj nepredvidivosti. Entropija se računa formulom:

$$Entropy(S) = -\sum_{i=1}^{C} p_i \, log_2(p_i)$$

- Informacijski dobitak: korišten za mjerenje smanjenja entropije na podijeljenom skupu podataka. Pokazuje koliko određena značajka A doprinosi povećanju čistoće skupa podataka nakon što se podaci podijele prema vrijednostima iste. Informacijski dobitak definiran je formulom:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

  Podskup S_v dobiven je dijeljenjem skupa podataka prema vrijednostima značajke A, a udio podataka u podskupu S_v korišten je za vaganje entropije podskupa s obzirom na odnos prema podskupu $S(\frac{|S_v|}{|S|})$.

2. Podjela skupa podataka: podaci su podijeljeni na podskupove na temelju vrijednosti značajki.
3. Rekurzivna izgradnja: proces rekurzivnog ponavljanja podataka vrši se sve dok podaci u podskupu ne pripadaju istoj klasi ili se ne ispune neki od mogućih kriterija zaustavljanja.

3.2.2. Naivan Bayesov klasifikator

Bayesov klasifikator se temelji na Bayesovom teoremu, koji daje metodu za izračunavanje uvjetne vjerojatnosti hipoteze y s obzirom na zapažanje X.

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

- $P(y|X)$: uvjetna vjerojatnost da ulaz X pripada klasi y, predstavlja vjerojatnost koju treba izračunati.
- $P(X|y)$: vjerojatnost dobivanja ulaznog podataka X pod uvjetom pripadnosti klasi y.
- $P(y)$: vjerojatnost da će klasa y biti odabrana bez obzira na druge podatke.
- $P(X)$: predstavlja ukupnu vrijednost opažanja X u svim klasama.

3.2.3. Evaluacija matrice zabune

Matricom zabune su prikazane stvarne u odnosu na predviđene vrijednosti modela. Sadrži četiri ključne komponente:

- Ispravno pozitivni (TP, engl. True Positives): ispravno predviđeni pozitivni slučajevi.
- Ispravno negativni (TN, engl. True Negatives): ispravno predviđeni negativni slučajevi.
- Lažno pozitivni (FP, engl. False Positives): pogrešno predviđeni pozitivni slučajevi.
- Lažno negativni (FN, engl. False Negatives): pogrešno predviđeni negativni slučajevi.

3.2.4. Kumulativni graf dobitka

Kumulativni graf dobitka vizualizira koliko dobro model klasifikacije razdvaja pozitivne od negativnih primjera unutar ukupne populacije, prikazujući to kroz odnos između stope istinitih pozitivnih predikcija (TPR) i stope lažnih pozitivnih predikcija (FPR) za različite pragove odlučivanja:

- TPR izračun:

$$TPR = \frac{TP}{TP + FN}$$

- FPR izračun:

$$FPR = \frac{FP}{FP + TN}$$

4. PRAKTIČNI RAD – PREDOBRADA PODATAKA I IMPLEMENTACIJA ALGORITAMA KLASIFIKACIJE

U ovom poglavlju detaljno se prikazuje praktična primjena teorijskih koncepata kroz proces predobrade podataka i implementaciju algoritama klasifikacije. Fokus je na korištenju ML.NET tehnologije za obradu stvarnih skupova podataka i evaluaciju performansi različitih modela strojnog učenja.

4.1. Predobrada ulaznih podataka

Predobrada ulaznih podataka ključan je korak svakog projekta strojnog učenja te kvaliteta obrade podataka utječe na mjerilo uspješnosti projekta. Postupak ML.NET tehnologije obuhvaća aktivnosti:

- Učitavanje podataka: početni korak učitavanja podataka iz različitih izvora poput baze podataka ili formata vrijednosti razdvojene zarezom (CSV, engl. Comma-Separated Values).
- Postupak čišćenja podataka: čišćenje podataka podrazumijeva identifikaciju i uklanjanje ili zamjenu nepravilnih vrijednosti, poput nedostajuće vrijednosti. To uključuje uklanjanje dupliciranih redaka ili zamjenu nedostajućih vrijednosti s prosječnim vrijednostima. Kvalitetno očišćeni podaci osiguravaju da model dobiva točne podatke za učenje.
- Transformacija ulaznih podataka: transformacija podataka uključuje prilagodbu podataka kako bi bili prikladni za modeliranje. Neke najčešće korištene metode su:
    - Normalizacija: skaliranje podataka tako da imaju isti raspon vrijednosti.
    - Kodiranje kategorijskih podataka: pretvaranje kategorijskih varijabli u numeričke vrijednosti.
    - Konkatenacija značajki: spajanje više značajki u jednu kako bi se stvorila nova.
    - Standardizacija: skaliranje podataka tako da imaju srednju vrijednost 0 i standardnu devijaciju 1.
- Podjela skupa podataka: posljednji korak u predobradi podataka je podjela skup na skup za treniranje modela i testni skup. Ovaj korak je ključan za procjenu učinkovitosti modela, jer omogućava testiranje modela na jednom dijelu podataka i testiranje njegove sposobnosti generalizacije predviđanja na nove podatke.

Problemi obrade podataka razrađeni su za svaki klasifikacijski problem zbog razlika binarnog i klasifikacijskog skupa podataka, uz primjer testa za transformaciju podataka binarne klasifikacije.

4.1.1. Priprema podataka binarnog podatkovnog skupa

1. Učitavanje podataka: osnovni korak pripreme prikazan kodom Kod 1.

```
Separators = new[] { ',' },
HasHeader = true,
Columns = new[]
{
      new TextLoader.Column("Značajka1", DataKind.Single, 0),
      new TextLoader.Column("Značajka2", DataKind.Single, 1),
      new TextLoader.Column("Oznaka", DataKind.Boolean, 2)
}
```

Kod 1. Učitavanje binarnih podataka

Za učitavanje korištena je klasa 'TextLoader' koja zahtjeva označavanje zaglavlja te definiranje stupaca sa značajkama. Obrađeni tipovi podataka su 'DataKind.Single' , koji obuhvaća različite numeričke značajke, te 'DataKind.Boolean' s oznakama sadržanim vrijednostima istina ili laž.

2. Čišćenja binarnih podataka: postupak čišćenja podataka vrši se kroz identifikaciju nedostajućih vrijednosti te zamjenom odgovarajućima ili uklanjanjem istih pošto je ključno da su svi podaci koji ulaze u model budu konzistentni. Prikazano kodom Kod 2.

```
var filterNedostajucihVrijednosti =
mlContext.Data.FilterRowsByMissingValues(dataView);
var naziviStupaca = dataView.Schema.Select(stupac =>
stupac.Name).ToArray();
var ciscenjePipeline =
mlContext.Transforms.ReplaceMissingValues(naziviStupaca.Select(ime
=> new InputOutputColumnPair(ime)).ToArray());
```

Kod 2. Čišćenje binarnih podataka identifikacijom nedostajućih vrijednosti

Čišćenje je ostvareno metodama 'FilterRowsByMissingValues' za uklanjanje redova gdje vrijednosti nedostaju, te 'ReplaceMissingValues' za zamjenu istih vrijednosti s prosječnim vrijednostima stupaca.

3. Transformacija ulaznih podataka: postupak prikazan kodovima, Kod *3* i Kod 4.

```
var stupci = new TextLoader.Column[brojZnačajki + 1];
for (int i = 0; i < brojZnačajki; i++)
{
stupci[i] = new TextLoader.Column($"Značajka{i}", tipZnačajki, i);
}
      stupci[brojZnačajki] = new TextLoader.Column("Oznaka",
tipOznake, brojZnačajki);
```

Kod 3. Transformacija generiranjem stupaca značajki binarne klasifikacije

```
var naziviStupacaZnačajki = new string[brojZnačajki];
for (int i = 0; i < brojZnačajki; i++)
{
      naziviStupacaZnačajki[i] = $"Značajka{i}";
}
```

Kod 4. Transformacija imenovanjem značajki stupaca binarne klasifikacije

"Metoda 'GenerateColumns' osigurava da su stupci konfigurirani s odgovarajućim tipovima podataka, a metodom 'GenerateFeatureColumnName' da je svaki stupac identificiran jedinstvenim nazivom. Nakon poziva svih ovih metoda u model se prosljeđuju završno obrađeni ulazni podaci na testiranje nakon konačne transformacije izvršene metodom 'Transforms.Concatenate' za spajanje svih značajki u jedan stupac.

Transformacija binarnih podataka dodatno je provjerena kreiranjem unit testa kojim se provjerava je li logika transformacija ispravna, provjerom jesu li stupci pravilno generirani te sadrže li ispravne tipove podataka. Ovo također demonstrira učinkovitost automatiziranih testova prikazano kodom Kod 5, u izradi programske podrške za strojno učenje unutar .NET okruženja:

```
[Test]
public void GenerirajStupce_KreiraIspravneStupce()
{
int brojZnačajki = 2;
var tipZnačajki = DataKind.Single;
var tipOznake = DataKind.Boolean;
var pripremaBinarnihPodataka = new BinaryDataPreparation();
```

```
var stupci = pripremaBinarnihPodataka.GenerirajStupce(brojZnačajki,
tipZnačajki, tipOznake);

Assert.AreEqual(brojZnačajki + 1, stupci.Length, "Broj stupaca
trebao bi biti brojZnačajki + 1.");
for (int i = 0; i < brojZnačajki; i++)
{
      Assert.AreEqual(tipZnačajki, stupci[i].DataKind, $"Tip podataka
      stupca treba biti {tipZnačajki}.");
}
Assert.AreEqual(tipOznake, stupci[brojZnačajki].DataKind, "Tip
podataka posljednjeg stupca treba biti tip Oznake.");}
```

Kod 5. Unit test transformacije binarne klasifikacije

Pozvanom metodom 'Assert' provjerava se ispravnost kreiranih metoda za transformacije, odnosno je li ispravan tip podataka za svaki stupac značajke, te je li tip podataka za posljednji stupac pravilno označen kao oznaka.

4.1.2. Priprema podataka tekstualnog podatkovnog skupa

1. Učitavanje podataka: primjena koda s obzirom na specifičnosti tekstualne klasifikacije prikazana kodom Kod *6*.

```
Separators = new[] { ',' },
HasHeader = true,
Columns = new[]
{
      new TextLoader.Column("Oznaka", DataKind.String, 0),
      new TextLoader.Column("Tekst", DataKind.String, 1)
}
```

Kod 6. Učitavanje tekstualnih podataka

Podatkovni tip mora sadržavati mogućnost rada s tekstualnim podacima, 'DataKind.String'.

2. Čišćenja tekstualnih podataka: kod za provjeru podataka tijekom čišćenja tekstualnih podataka, demonstrirano kodom Kod 7.

```
var filterNedostajucihVrijednosti =
mlContext.Data.FilterRowsByMissingValues(dataView, "Tekst");
```

```
var ciscenjePipeline = mlContext.Transforms.CustomMapping(
(in InputData input, out OutputData output) => {
output.Tekst = string.IsNullOrEmpty(input.Tekst) ? "" : input.Tekst;
output.Oznaka = input.Oznaka; },
null);
```

Kod 7. Čišćenje tekstualnih podataka uklanjanjem praznih polja i zamjenom praznih podataka

Bitan uvjet za treniranje tekstualnih skupova su uklonjena prazna tekstualna polja što je ostvareno metodom 'FilterRowsByMissingValues' i zamjenom s praznim podatkom tipa 'String' metodom 'Transforms.CustomMapping'.

3. Transformacija ulaznih podataka: transformacija za tekstualne podatke predstavlja kompleksniji problem pošto je nužno osigurati da su tekstualni podaci okarakterizirani kao numeričke vrijednosti iz kojeg model može učiti prikazano kodom Kod 8.

```
var pipeline = mlContext.Transforms.Text.FeaturizeText("Značajke",
"Tekst")
.Append(mlContext.Transforms.Conversion.MapValueToKey("Oznaka"))
.Append(mlContext.Transforms.Concatenate("Značajke", "Tekst"));
var transformiraniPodaci =
pipeline.Fit(dataView).Transform(dataView);

private TextLoader.Column[] GenerirajStupce(int brojZnacajki,
DataKind tipZnacajke, DataKind tipOznake)
{
return new TextLoader.Column[]
{
      new TextLoader.Column("Oznaka", tipOznake, 0),
      new TextLoader.Column("Tekst", tipZnacajke, 1)
}; }
```

Kod 8. Transformacija tekstualnih podataka konverzijom oznaka

Problem je riješen pozivom metode 'Transforms.Conversion.MapValueToKey' za konverziju oznaka u tražene vrijednosti.

Kako se kod tekstualnih podatkovnih skupova često radi s neuravnoteženim klasama zbog razlike u broju primjera između različitih kategorija, tipa željene i neželjene elektroničke pošte, bitno je podatke balansirati postupkom dupliciranja ili smanjivanja podataka tako da svaka

klasa ima približno jednak broj uzoraka. Primjer za balansiranje podataka prikazan je kodom Kod 9.

```
var uzorciPodataka =
mlContext.Data.CreateEnumerable<OriginalLabelData>(treningPodaci,
reuseRowObject: false).ToList();
var brojOznaka = uzorciPodataka.GroupBy(x =>
x.Oznaka).ToDictionary(g => g.Key, g => g.Count());
var maxBroj = brojOznaka.Values.Max();
var balansiraniPodaci = new List<OriginalLabelData>();
foreach (var oznaka u brojOznaka.Keys)
{
var items = uzorciPodataka.Where(x => x.Oznaka == oznaka).ToList();
while (items.Count < maxBroj)
{
      items.AddRange(items.Take(maxBroj - items.Count));
}
balansiraniPodaci.AddRange(items);
```

Kod 9. Transformacija tekstualnih podataka balansiranjem podataka

Implementacijom metoda 'ToList()' za ravnomjernu distribuciju oznaka i metode 'AddRange()' za repliciranje uzoraka, osigurava se da se uzorci svake oznake ponavljaju dok se ne dosegne broj uzoraka oznake s najviše uzoraka. Pravilnost transformacije tekstualnih podataka je provjerena kreiranjem unit testa demonstiranog kodom Kod 10.

```
[Test]
public void PripremiPodatke_TransformiraTekstualnePodatkeIspravno()
{
var pripremaTekstualnihPodataka = new PripremaTekstualnihPodataka();
var podaci = new[]
{
new { Oznaka = "pozitivno", Tekst = "Ovaj mail je točan" },
new { Oznaka = "negativno", Tekst = "Ovaj mail je lažan" },
new { Oznaka = "pozitivno", Tekst = "Odlična brzina" },
new { Oznaka = "negativno", Tekst = "Niska brzina" }
```

```
};
File.WriteAllLines(_putanjaDoPodataka, new[]
{
"Oznaka,Tekst",
"pozitivno,Ovaj mail je točan",
"negativno,Ovaj mail je lažan",
"pozitivno,Odlična brzina",
"negativno,Niska brzina"
});

var (treningPodaci, testniPodaci) =
pripremaTekstualnihPodataka.PripremiPodatke(_mlKontekst,
_putanjaDoPodataka, 1);

Assert.IsNotNull(treningPodaci, "Trening podaci ne bi smjeli biti
null.");
Assert.IsNotNull(testniPodaci, "Testni podaci ne bi smjeli biti
null.");

var pregledTreningPodataka = treningPodaci.Preview();
var pregledTestnihPodataka = testniPodaci.Preview();

Console.WriteLine("Pregled Trening Podataka:");
foreach (var red in pregledTreningPodataka.RowView)
{
Console.WriteLine(string.Join(", ", red.Values.Select(v =>
v.Value)));
}
Console.WriteLine("Pregled Testnih Podataka:");
foreach (var red u pregledTestnihPodataka.RowView)
{
Console.WriteLine(string.Join(", ", red.Values.Select(v =>
v.Value)));
```

```
}

Assert.IsTrue(pregledTreningPodataka.RowView.Length > 0, "Trening
podaci trebaju sadržavati retke.");
Assert.IsTrue(pregledTestnihPodataka.RowView.Length > 0, "Testni
podaci trebaju sadržavati retke.");
}
```

Kod 10. Unit test transformacije tekstualne klasifikacije

Funkcija 'PripremiPodakte' transformira tekstualne podatke i dijeli ih na trening i testne skupove. Test koristi metode 'Assert' kako bi provjerio postojanje podataka, odnosno da trening i testni podaci nisu 'null' vrijednosti i da sadrže ispravne retke podataka.

4.2. Implementacija algoritama i metoda evaluacije performansi

Implementacija algoritama izvodi se pozivom ugrađenih metoda. Tema ovog poglavlja je demonstracija idealnih metoda optimizacije modela strojnog učenja. Za potrebe analize postavljene su metode evaluacije performansi svakog modela.

4.2.1. Implementacija logističke regresije

Za postupak treniranja modela logističke regresije je primijenjen algoritam s ograničenom memorijom Broyden–Fletcher–Goldfarb–Shanno (L-BFGS). Koristi se za pronalaženje minimalne vrijednosti funkcije uz optimizaciju log-gubitka funkcije, koja mjeri razliku između predviđenih vjerojatnosti i stvarnih oznaka, s ciljem poboljšanja točnosti predikcije. Primjer je prikazan kodom Kod 11.

```
var pipeline = _mlKontekst.Transforms.Concatenate("Značajke",
"Značajke")
      .Append(_mlKontekst.Transforms.NormalizeMinMax("Značajke"))
      .Append(_mlKontekst.BinaryClassification.Trainers.LbfgsLogistic
Regression(new LbfgsLogisticRegressionBinaryTrainer.Options
{
      MaximumNumberOfIterations = 100,
      L2Regularization = 0.1f
}));
```

Kod 11. Implementacija logističke regresije

Potrebama optimizacije algoritma prilagođena je maksimalna brojčanost iteracija metodom 'MaximumNumberOfIterations' postavljena na 100, što predstavlja maksimalan broj iteracija prije nego li se pronađu optimalne vrijednosti koeficijenata što sprječava predugo trajanje treniranja modela na većem podatkovnom skupu.

Metoda 'L2Regularization' postavlja kaznu za kvadratnu veličinu koeficijenata u funkciji gubitka s ciljem održavanjem koeficijenata manjim kako bi se održala sposobnost generalizacije.

4.2.2. Implementacija perceptrona s prosjekom

Za implementaciju modela perceptrona korišten je algoritam perceptrona s prosjekom realiziran kodom Kod 12, koristeći prosječne težine kroz iteracije pozivanjem funkcije:

```
var pipeline = _mlKontekst.Transforms.Concatenate("Značajke",
"Značajke")
      .Append(_mlKontekst.Transforms.NormalizeMinMax("Značajke"))
      .Append(_mlKontekst.BinaryClassification.Trainers.AveragedPerce
ptron(new AveragedPerceptronTrainer.Options
{
      LearningRate = 0.1f,
      NumberOfIterations = 10
}));
var treniranModel = pipeline.Fit(trenirajuciPodaci);
var skoriraniPodaci = treniranModel.Transform(trenirajuciPodaci);
var kalibratorProcjenitelj =
_mlKontekst.BinaryClassification.Calibrators.Platt();
var kalibratorTransformator =
kalibratorEstimator.Fit(skoriraniPodaci);
_treniranModel = new TransformerChain<ITransformer>(treniranModel,
kalibratorTransformer);
```

Kod 12. Implementacija perceptrona s prosjekom te kalibracija Plattovom metodom

Algoritam je dodatno kalibriran metodom 'LearningRate', koja određuje koliko se brzo model prilagođava tijekom učenja, te metodom za Plattovu kalibraciju 'Calibrators.Platt'.

4.2.3. Implementacija evaluacije performansi binarne klasifikacije

Odabrane metrike su dobivene pozivom ugrađenih metoda.

- 'Accuracy' : točnost.
- 'PositivePrecision': preciznost.
- 'PositiveRecall': odaziv.

4.2.4. Implementacija krivulja radne karakteristike

Metrika krivulje je pozvana metodom 'AreaUnderRocCurve'.

4.2.5. Implementacija evaluacije performansi predikcija

Primjenom ML.NET knjižnice navedeni rezultati izračunati su pozivom metoda.

- 'PositivePrecision': preciznost pozitivnih predikcija (PPV).
- 'PositiveRecall': odaziv pozitivnih predikcija (TPR).
- 'NegativePrecision': preciznost negativnih predikcija (NPV).
- 'NegativeRecall': odaziv negativnih predikcija (TNR).

4.2.6. Implementacija stabla odluke

Za postupak treniranja modela stabla odluke primijenjen je algoritam FastForest, kao varijanta algoritma Random Forest temeljen na skupu stabala odluke za tekstualnu klasifikaciju. Algoritam nasumično koristi podskupove značajki i podataka za treniranje svakog stabla rezultirajući robusnim modelom implementiranim kodom Kod 13.

```
var pipeline =
_mlContext.Transforms.Conversion.MapValueToKey("Oznake")
.Append(_mlContext.Transforms.Text.FeaturizeText("Značajke",
"Tekst"))
.AppendCacheCheckpoint(_mlContext)
.Append(_mlContext.MulticlassClassification.Trainers.OneVersusAll(
binaryEstimator:
_mlContext.BinaryClassification.Trainers.FastForest(new
FastForestBinaryTrainer.Options
{
      NumberOfLeaves = 50,
      NumberOfTrees = 100,
```

```
      MinimumExampleCountPerLeaf = 1
}), labelColumnName: " Oznake "))
.Append(_mlContext.Transforms.Conversion.MapKeyToValue("PredviđeneOz
nake"));
```

Kod 13. Implementacija stabla odluke

Model je kalibriran metodama za optimizaciju stabla odluke:

- 'NumberOfLeaves': broj listova postavljen na 50, određujući maksimalan broj listova na svakom stablu. Više listova pridonosi preciznijem modelu, ali povećava vrijeme treniranje i može dovesti do prenaučenosti modela (overfitting).
- 'MinimumExampleCountPerLeaf': minimalni broj primjera po listu postavljen na 1. Vrijednost određuje minimalan broj primjera koji moraju biti prisutni u listu stabla, s nižom vrijednosti povećava se preciznost modela ali i s time razmjerno rizik od prekomjernog učenja.
- 'NumberOfTrees': broj stabla postavljen na 100 određuje koliko je stabla trenirano u šumi.

4.2.7. Implementacija naivnog Bayesovog klasifikatora

Korišten je Naivni Bayesov algoritam temeljen na Bayesovu teoremu. U fazi treniranja model je podešen različitim duljinama sekvenci od n elemenata(n-gram) za optimalnije prepoznavanje tekstualnih obrazaca, prikazano kodom Kod 14.

```
var pipeline =
_mlContext.Transforms.Conversion.MapValueToKey("Oznaka")
.Append(_mlContext.Transforms.Text.FeaturizeText("Značajke", new
TextFeaturizingEstimator.Options
{
WordFeatureExtractor = new WordBagEstimator.Options
{
      NgramLength = ngramLength,
      UseAllLengths = true,
      Weighting = NgramExtractingEstimator.WeightingCriteria.TfIdf
},
CharFeatureExtractor = new WordBagEstimator.Options
```

```
{
      NgramLength = 3,
      UseAllLengths = false,
      Weighting = NgramExtractingEstimator.WeightingCriteria.TfIdf
},
Norm = TextFeaturizingEstimator.NormFunction.L2,
KeepPunctuations = false,
StopWordsRemoverOptions = new StopWordsRemovingEstimator.Options
{
      Language = TextFeaturizingEstimator.Language.English
}
}, "Tekst"))
.Append(_mlContext.Transforms.NormalizeMinMax("Značajke"))
.Append(_mlContext.MulticlassClassification.Trainers.NaiveBayes(labe
lColumnName: "Oznaka", featureColumnName: "Značajke"))
.Append(_mlContext.Transforms.Conversion.MapKeyToValue("PredviđenaOzn
aka"));
```

Kod 14. Implementacija naivnog Bayesovog klasifikatora

Implementacija gornjeg koda zahtjeva prilagodbu više parametra:

- 'NormalizeMinMax': metoda se koristi za normalizaciju značajki tako da njihove vrijednosti skalira unutar zadanog raspona, najčešće između minimalne i maksimalne vrijednosti skupa podataka, odnosno između 0 i 1. Ovo osigurava da su sve značajke na istoj skali, tako da značajke s većim numeričkim vrijednosti ne dominiraju u skupu.
- 'MapKeyToValue' i 'MapValueToKey': dvije ključne metode za obradu tekstualnih podataka. 'MapKeyToValue' pretvara tekstualne oznake u numeričke ključeve što je bitno poradi činjenice da algoritmi strojnog učenja zahtijevaju numeričke podatke za rad. Nakon što model završi s predikcijama metoda 'MapValueToKey' pretvara ključeve natrag u originalne tekstualne vrijednosti.
- 'WordFeatureExtractor' i 'CharFeatureExtractor': podešene dvije bitne metode za ekstrakciju značajki u tekstu te pretvaranje tekstualnih podataka u numeričke značajke. Prva se fokusira na značajke iz riječi u tekstu, a druga na sekvence znakove. Bitni parametri ovih metoda su:

- 'NgramLength': parametar određuje broj elemenata koji će se grupirati zajedno kako bi formirali n-gram omogućujući analizu različitih kombinacija riječi ili znakova u tekstu.
- 'UseAllLengths': kada je metoda postavljena kao istinita, koriste se sve duljine n-grama, počevši od 1 pa sve do postavljene maksimalne duljine. Parametar pruža analizu raznih kombinacija elemenata.
- 'Weighting': korištenjem parametra određuje se bitnost n-grama. Metoda 'TfIdf' (Term Frequency-Inverse Document Frequency) povećava težinu rijetkih, ali značajnih n-grama dok smanjuje težinu uobičajenima, omogućujući modelu da se fokusira na kjučne pojmove.

- 'StopWordsRemoverOptions': metoda za uklanjanje beskorisnih riječi (stop words).
- 'NormFunction.L2: L2 norma, poznata kao i Euklidska norma za normalizaciju značajki. Vrijednosti značajki se normaliziraju tako da se njihov kvadratni zbroj izjednačava s 1 što sprječava da značajke s većim apsolutnim vrijednostima prevladavaju nad manjim.

4.2.8. Implementacija evaluacije performansi tekstualne klasifikacije

Odabrane metrike su dobivene pozivom ugrađenih metoda.

- 'MacroAccuracy': makro točnost.
- 'MicroAccuracy': mikro točnost.
- 'LogLoss': log-gubitak.

4.2.9. Implementacija evaluacije matrice zabune

Metrike matrice zabune prebrojane su iz objekta metrika klasifikacije 'metrics', ugrađenom klasom unutar NET.ML knjižnice. Prikaz realiziranog prebrojavanja dan je kodom Kod 15.

```
{var brojac = metrics.ConfusionMatrix.Counts;
int tpHam = (int)counts[0][0];
int tnHam = (int)counts[1][1];
int fpHam = (int)counts[1][0];
int fnHam = (int)counts[0][1];
```

```
int tpSpam = (int)counts[1][1];
int tnSpam = (int)counts[0][0];
int fpSpam = (int)counts[0][1];
int fnSpam = (int)counts[1][0];};
```

Kod 15. Implementacija evaluacije matrice zabune

4.2.10. Implementacija kumulativnog grafa dobitka

Nadalje, vrijednosti za određivanje kumulativnog grafa dobitka su izračunate iz metrika matrice zabune, prikazano kodom Kod 16.

```
double tprHam = (double)tpHam / (tpHam + fnHam);
double fprHam = (double)fpHam / (fpHam + tnHam);
double tprSpam = (double)tpSpam / (tpSpam + fnSpam);
double fprSpam = (double)fpSpam / (fpSpam + tnSpam);
```

Kod 16. Implementacija kumulativnog grafa dobitka

4.3. Metode evaluacije

Za svaki od modela izračunate su metrike koje uključuju točnost, preciznost i druge pokazatelje performansi. Osim numeričkih vrijednosti metrika, priloženi su i detaljni opisi te grafikoni koji vizualiziraju rezultate. Cilj ovih evaluacija je omogućiti donošenje relevantnih zaključaka o učinkovitosti i pouzdanosti svakog modela.

4.3.1. Performanse binarne klasifikacije

Za modele logističke regresije, prikazan slikom Slika 1 i perceptrona s prosjekom, prikazan slikom Slika 2, primjenom algoritama dobivene su sljedeće metrike:

- Logistička regresija: točnost: 0.77, preciznost: 0.76, odaziv: 0.51.

Slika 1. Rezultati performansi logističke regresije (autorski rad)

- Perceptron s prosjekom: točnost: 0.76, preciznost: 0.75, odaziv: 0.49.

Slika 2. Rezultati performansi perceptrona s prosjekom (autorski rad)

- Točnost modela: određena omjerom ispravnih predikcija u odnosu na ukupan broj predikcija. U kontekstu projekta, vidljivo je da oba modela klasificiraju minimalno 76 % primjera, što ukazuje na to da model dobro razlikuje osobe s dijabetesom od onih bez njega."
- Preciznost modela: omjer ispravno pozitivnih predikcija u odnosu na ukupan broj stvarno pozitivnih primjera. Bitna je visoka vrijednost kako bi se smanjio broj lažno pozitivnih predikcija što je ključno za medicinsku dijagnostiku u vidu smanjenja nepotrebnih daljnjih pregleda.
- Odaziv modela: predstavlja bitnu karakteristiku modela, omjer koliko je ispravnih pozitivnih predikcija u odnosu na ukupan broj stvarno pozitivnih primjera. Osrednji rezultat označava da modeli propuštaju dijagnosticirati određen postotak stvarno pozitivnih slučajeva.

4.3.2. Evaluacija performansi predikcija

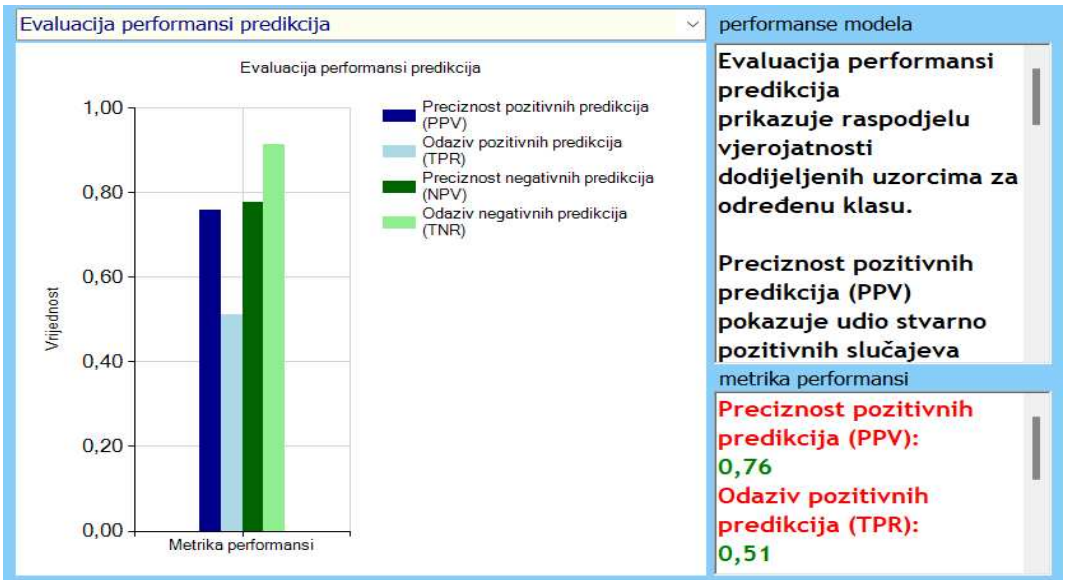
Prikaz rezultata evaluacije performansi predikcija prikazan je tablicom Tablica 1.

	Preciznost pozitivnih predikcija (PPV)	Odaziv pozitivnih predikcija (TPR)	Preciznost negativnih predikcija (NPV)	Odaziv negativnih predikcija (TNR)
Logistička regresija	0.76	0.51	0.78	0.91
Perceptron s prosjekom	0.75	0.49	0.77	0.91

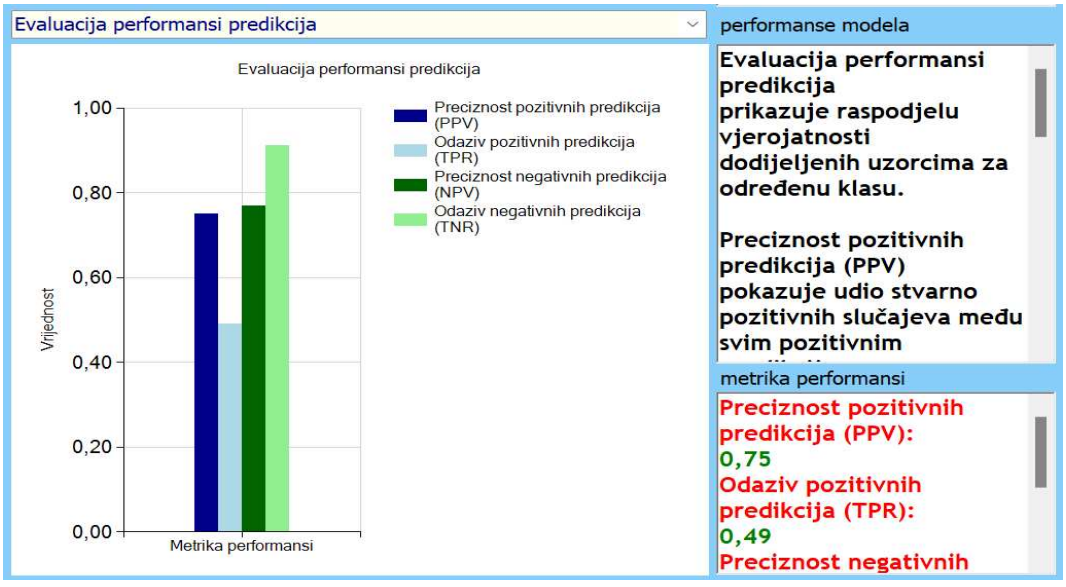
Tablica 1. Evaluacija performansi predikcija

Oba dva modela imaju visoku preciznost negativnih predikcija i odaziv negativnih predikcija, što znači da su izrazito pouzdani u identificiranju negativnih slučajeva.

Grafički prikaz rezultata za logističku regresiju dan je grafikonom Grafikon 3, dok su rezultati perceptrona s prosjekom prikazani grafikonom Grafikon 4.



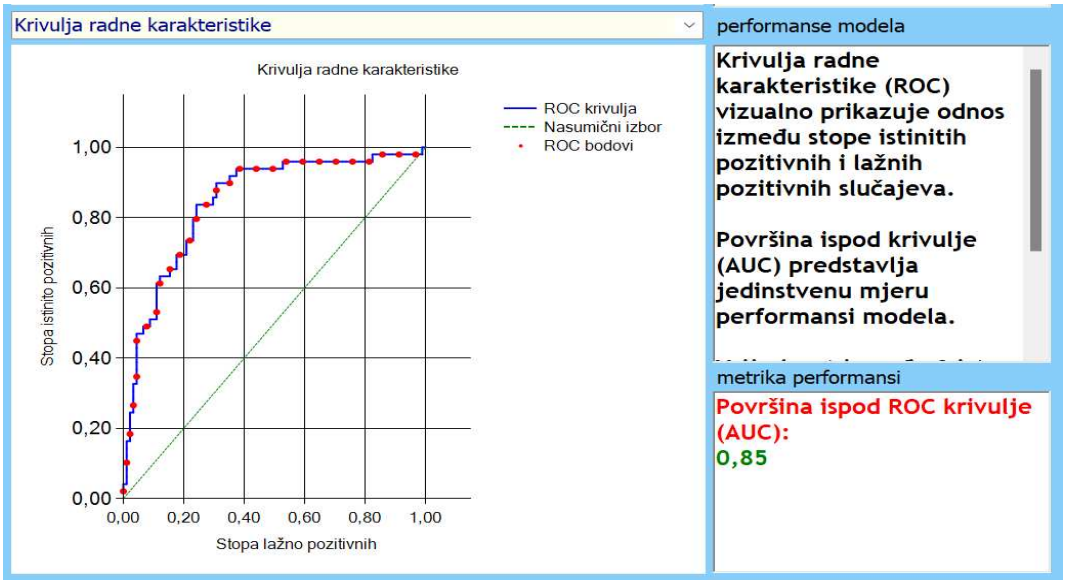
Grafikon 3. Evaluacija performansi predikcija logističke regresije (autorski rad)



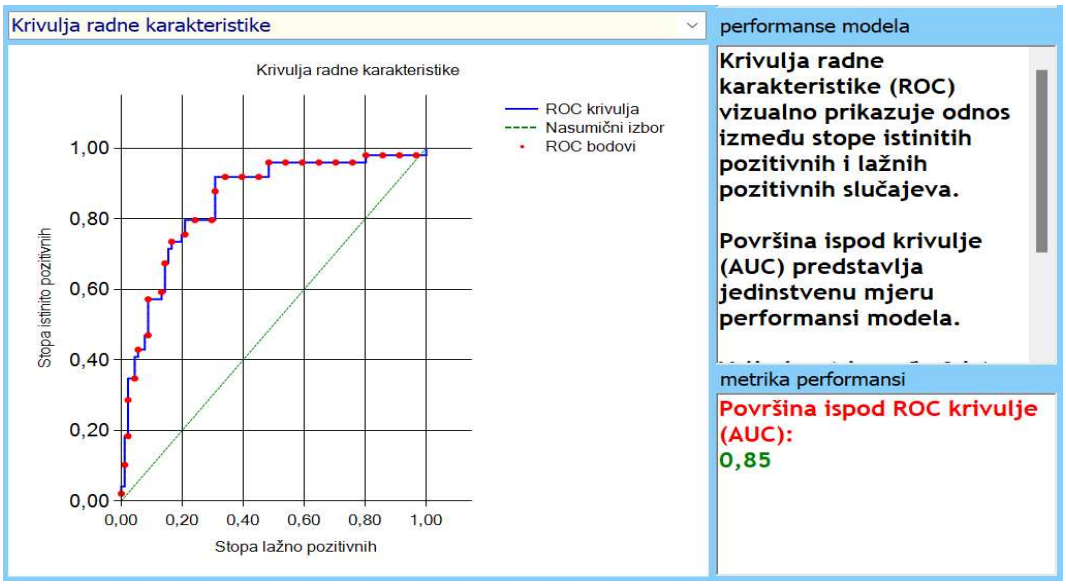
Grafikon 4. Evaluacija performansi predikcija perceptrona s prosjekom (autorski rad)

4.3.3. Krivulja radne karakteristike

Dobivena metrika za površinu ispod krivulje (AUC) iznosi 0.85 za oba modela binarne klasifikacije. ROC krivulje prikazuju točke koje odgovaraju različitim pragovima klasifikacije, prikazano grafikonom krivulje radne karakteristike za logističku regresiju, Grafikon 5 i grafikonom krivulje radne karakteristike perceptrona s prosjekom, Grafikon 6.



Grafikon 5. Krivulja radne karakteristike logističke regresije (autorski rad)



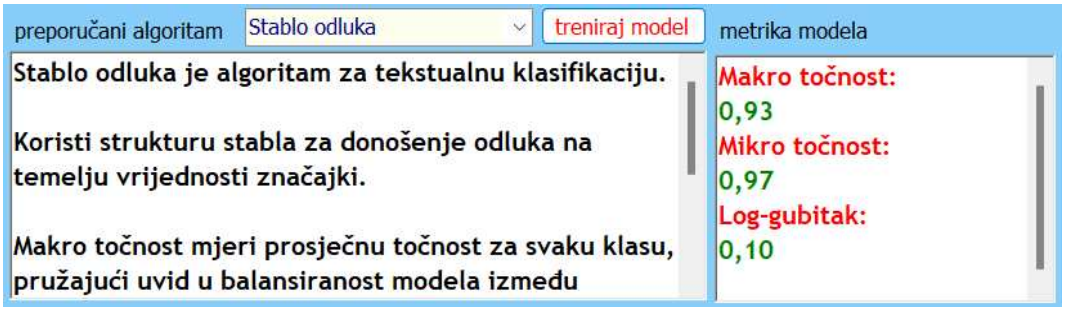
Grafikon 6. Krivulja radne karakteristike perceptrona s prosjekom (autorski rad)

Oba modela učinkovito razlikuju između pozitivnih i negativnih slučajeva, što ih čini pouzdanim opcijama za binarnu klasifikaciju. Male međusobne razlike u pragovima klasifikacije su vidljive na grafikonima.

4.3.4. Performanse tekstualne klasifikacije

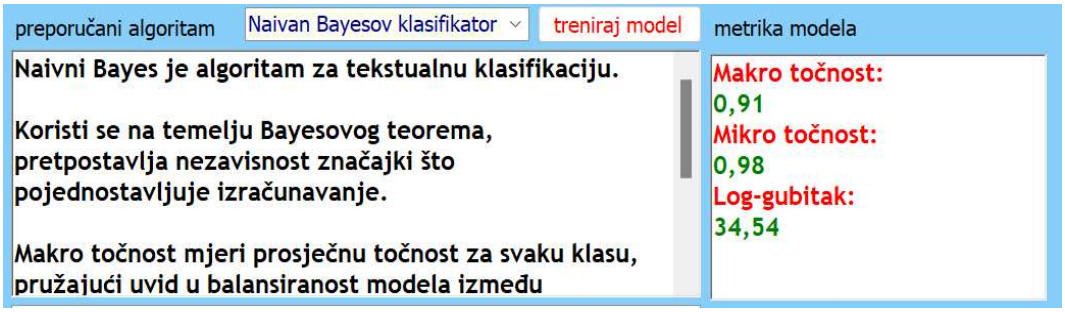
Za modele stablo odluke, prikazan slikom Slika 3 i naivan Bayesov klasifikator, prikazan slikom Slika 4 primjenom algoritama dobivene su sljedeće metrike:

- Stablo odluka: makro točnost: 0.93, mikro točnost: 0.97, log-gubitak: 0.10.



Slika 3. Rezultati performansi stabla odluke (autorski rad)

- Naivan Bayesov klasifikator: makro točnost: 0.91, mikro točnost: 0.98, log-gubitak: 34.54.



Slika 4. Rezultati performansi naivnog Bayesova klasifikatora (autorski rad)

- Makro točnost modela: mjera prosječne točnosti za svaku klasu, pružajući uvid u balansiranost između različitih klasa. Visoke vrijednosti pokazuju da modeli uspješno prepoznaju ham i spam poruke što je bitno za uravnoteženu klasifikaciju.
- Mikro točnost modela: mjera ukupne točnosti modela uzimajući u obzir sve primjere bez obzira na klasu. Visoke vrijednosti ukazuju da modeli uspješno klasificiraju velik broj primjera, što je pozitivno u pogledu praktičnog problema gdje je većina poruka označena kao ham.
- Log-gubitak modela: log-gubitak mjeri prosječni log-gubitak klasifikatora. Mjera ukazuje na pouzdanost predikcija i kvalitetu kalibracije. U projektnom zadatku stablo odluka pokazuje veću sigurnost svojih predikcija.

4.3.5. Evaluacija matrice zabune

Rezultat evaluacije matrice zabune za klasu označenu kao ham prikazan tablicom Tablica 2.

	Ispravno pozitivni (TP)	Ispravno negativni (TN)	Lažno pozitivni (FP)	Lažno negativni (FN)
Stablo odluka	982	140	19	16
Naivan Bayesov klasifikator	998	131	28	0

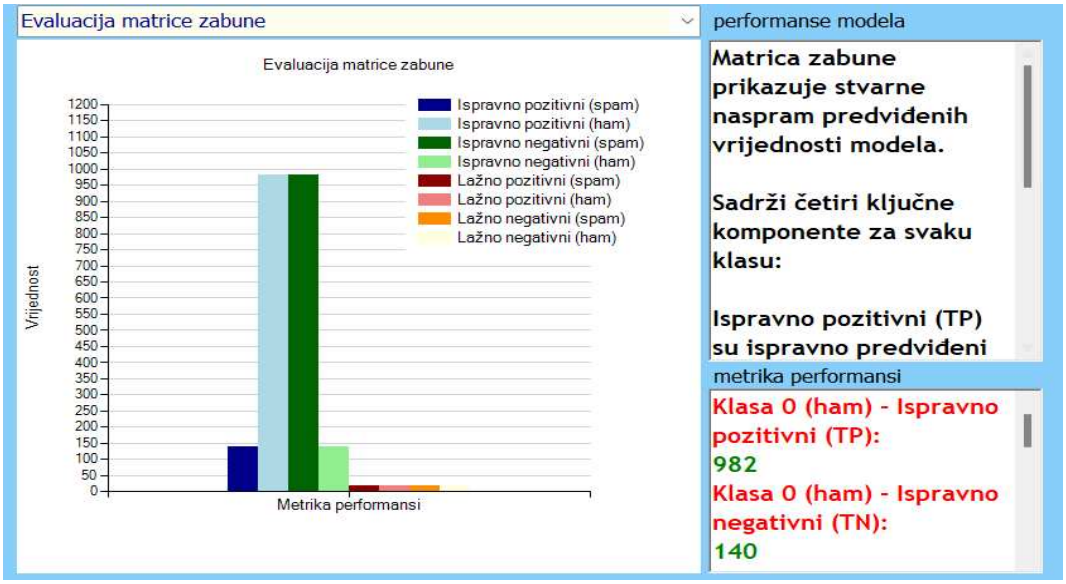
Tablica 2. Rezultati evaluacije matrice zabune za klasu označenu kao ham

Rezultat evaluacije matrice zabune za klasu označenu kao spam prikazan tablicom Tablica 3.

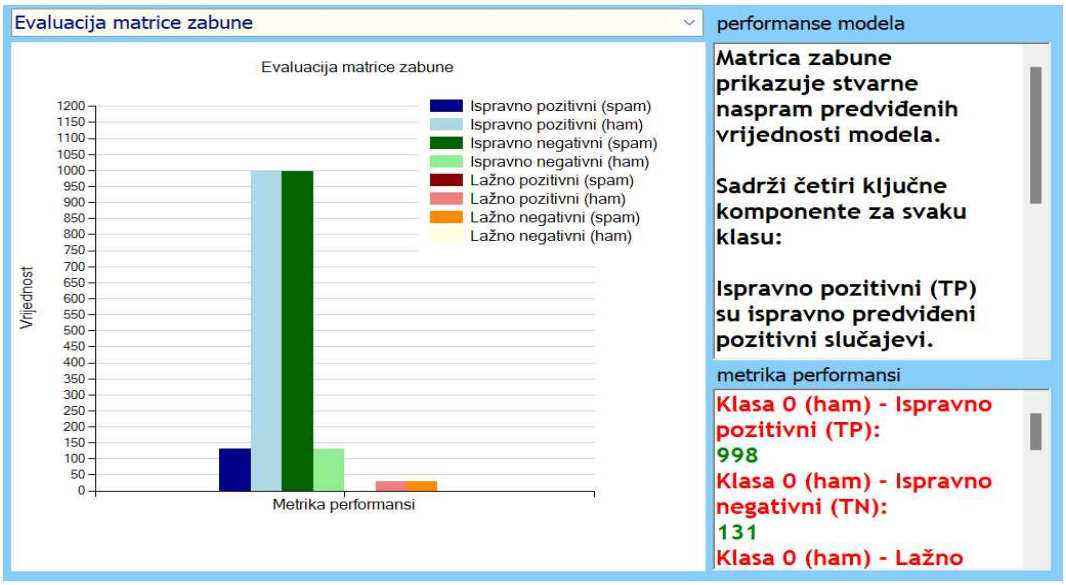
	Ispravno pozitivni (TP)	Ispravno negativni (TN)	Lažno pozitivni (FP)	Lažno negativni (FN)
Stablo odluka	140	982	16	19
Naivan Bayesov klasifikator	131	998	0	28

Tablica 3. Rezultati evaluacije matrice zabune za klasu označenu kao spam

Oba dva modela pokazuju visoku učinkovitost u klasifikaciji ham i spam poruka, s različitim karakteristikama, prikazano grafikonom Grafikon 7 za stablo odluke i grafikonom Grafikon 8 za naivan Bayesov klasifikator:



Grafikon 7. Evaluacija matrice zabune stabla odluke (autorski rad)



Grafikon 8. Evaluacija matrice zabune naivnog Bayesova klasifikatora (autorski rad)

Bitno je primijetiti simetričnost između klasa ham i spam za oba dva modela, što ukazuje na konzistentnost modela u prepoznavanju klasa bez pristranosti.

4.3.6. Kumulativni graf dobitka

Prikaz rezultata kumulativnog grafa dobitka za klasu označenu kao ham nalazi se u tablici Tablica 4.

	Stopa istinitih pozitivnih (TPR)	Stopa lažnih pozitivnih (FPR)
Stablo odluka	0.98	0.12
Naivan Bayesov klasifikator	1.00	1.00

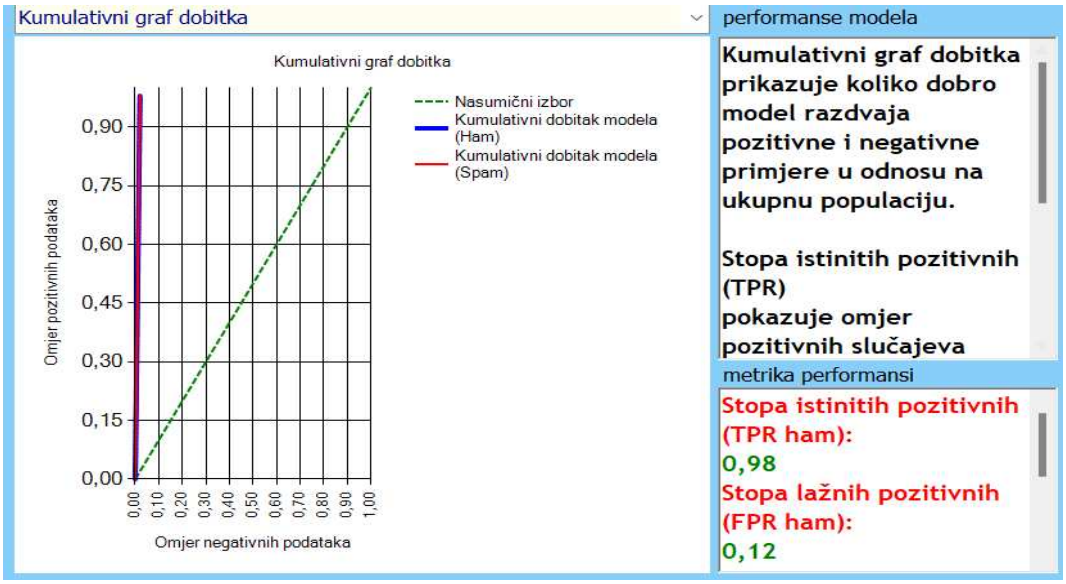
Tablica 4. Rezultat kumulativnog grafa dobitka za klasu označenu kao ham

Detaljni rezultati kumulativnog grafa dobitka za klasu označenu kao spam prikazani su u tablici Tablica 5.

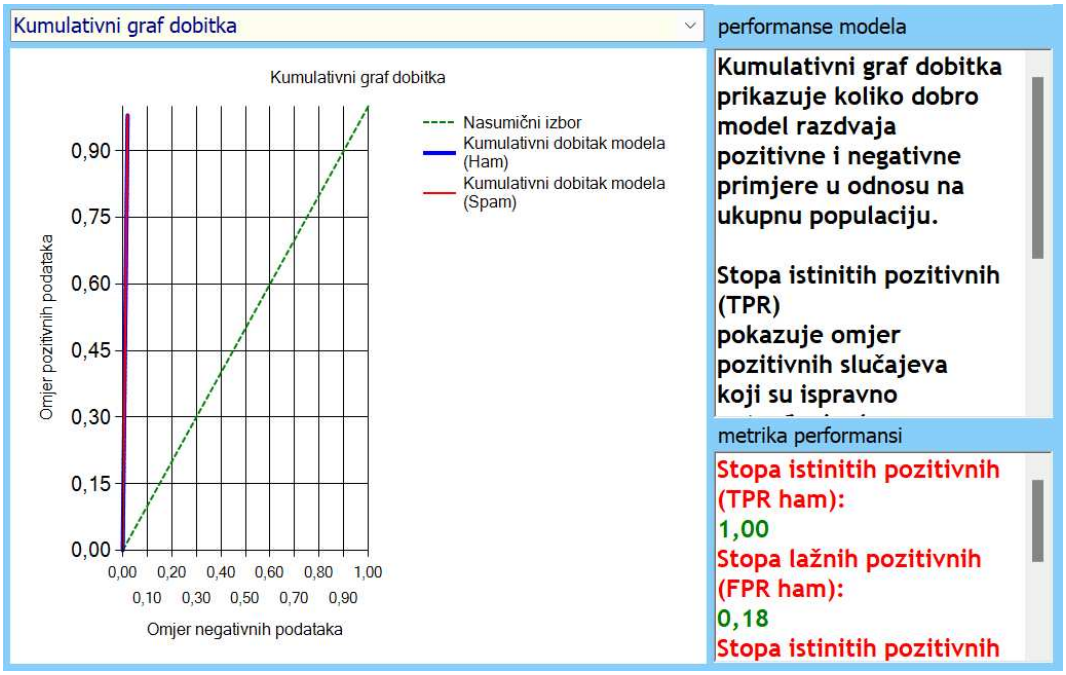
	Stopa istinitih pozitivnih (TPR)	Stopa lažnih pozitivnih (FPR)
Stablo odluka	0.88	0.02
Naivan Bayesov klasifikator	0.82	0.00

Tablica 5. Rezultati kumulativnog grafa dobitka za klasu označenu kao spam

Kumulativni graf dobitka prikazan je grafikonom Grafikon 9 za stablo odluka te grafikonom Grafikon 10 za naivan Bayesov klasifikator.



Grafikon 9. Kumulativni graf dobitka stabla odluke (autorski rad)



Grafikon 10. Kumulativni graf dobitka naivnog Bayesova klasifikatora (autorski rad)

Rezultati pokazuju da oba modela imaju svoje prednosti i nedostatke, ovisno o potrebama klasifikacije. Stablo odluka pokazuje visoku točnost u prepoznavanju ham poruka uz relativno nisku stopu lažnih pozitivnih, što predstavlja uravnoteženu klasifikaciju. Naivan Bayesov klasifikator ima visoku točnost za ham, ali se slabije nosi s prepoznavanjem spam poruka.

5. ZAKLJUČAK

Predmet ovog rada bila je demonstracija izrade programskog rješenja unutar .NET okruženja korištenjem ML.NET knjižnice za rješavanje problema klasifikacije strojnog učenja. Izrada aplikacije koja omogućuje unos podataka, prikaz kroz ugrađene grafikone i odabir algoritma ovisno o tipu klasifikacije uspješno je ostvarena pravilnom implementacijom modela. Integracija ML.NET knjižnice omogućila je jednostavno upravljanje procesom treniranja modela, predobrade podataka i evaluacije performansi.

Algoritmi kao što su logistička regresija, perceptron s prosjekom, stablo odluke i naivni Bayesov klasifikator pažljivo su odabrani i prilagođeni specifičnim klasifikacijskim zadacima, što je omogućilo optimizaciju rezultata. Prikaz rezultata, opis algoritama i grafikona omogućuje bolje razumijevanje modela strojnog učenja te olakšava korisnicima shvaćanje funkcioniranja različitih algoritama i njihovih optimizacija.

Logistička regresija, korištena za binarnu klasifikaciju, pokazala se kao jednostavan, ali vrlo učinkovit model. Sposobnost predviđanja pripadnosti jednoj od dvije klase, temeljena na značajkama poput razine glukoze i krvnog pritiska, omogućila je precizne i pouzdane rezultate. Proces predobrade podataka uključivao je čišćenje podataka, uklanjanje nedostajućih vrijednosti i normalizaciju značajki kako bi se osigurala konzistentnost i kvaliteta ulaznih podataka. Normalizacija značajki bila je ključna jer je omogućila modelu ravnomjernu obradu različitih varijabli, smanjujući pristranost prema bilo kojoj značajki.

Perceptron s prosjekom, prilagođen Plattovom kalibracijom, dodatno je poboljšao točnost predikcija smanjujući prenaučenost i osiguravajući bolju generalizaciju na neviđenim podacima. Ovaj model pokazao je dobre rezultate u prepoznavanju dijabetesa, gdje su preciznost i odaziv bili ključni.

Stablo odluke i naivni Bayesov klasifikator korišteni su za tekstualnu klasifikaciju, gdje je demonstrirana robusnost i sposobnost rješavanja složenih interakcija među značajkama. Stablo odluke istaknulo se u prepoznavanju obrazaca unutar tekstualnih podataka, omogućujući jasne i lako razumljive odluke temeljene na različitim frazama i riječima. Naivni Bayesov klasifikator, zahvaljujući svojoj sposobnosti brze analize velikih količina tekstualnih podataka, pokazao se kao izuzetno brz i precizan model za klasifikaciju poruka.

Proces predobrade tekstualnih podataka uključivao je ekstrakciju značajki, kodiranje tekstualnih podataka i balansiranje klasa, čime se osigurala točnost klasifikacije. Balansiranje

klasa bilo je ključno kako bi se osiguralo da model jednako dobro prepoznaje spam i ham poruke, smanjujući pristranost prema bilo kojoj klasi

Razvijena aplikacija omogućuje preciznu analizu i vizualizaciju podataka, što korisnicima omogućuje donošenje informiranih odluka na temelju rezultata strojnog učenja. Detaljna implementacija i testiranje uspješno su demonstrirali kako se ML.NET knjižnica može koristiti za rješavanje složenih problema klasifikacije unutar .NET okruženja. Aplikacija ne samo da olakšava proces treniranja i evaluacije modela, već i omogućuje korisnicima da kroz interaktivno grafičko sučelje bolje razumiju funkcioniranje i optimizaciju različitih algoritama strojnog učenja.

LITERATURA

1. Esposito, D., Esposito, F., (2020), *Programming ML.NET (Developer Reference)*, Microsoft Press, Redmond, WA.
2. Mukherjee, S., (2019), *ML.NET Revealed: Simple Tools for Applying Machine Learning to Your Applications*, Apress, New York, NY.
3. Duda, R. O., Hart, P. E., Stork, D. G., (2000), *Pattern Classification*, 2nd ed., Wiley-Interscience, New York, NY.
4. Microsoft, (n.d.), *.NET Machine Learning*, dostupno na: https://learn.microsoft.com/en-us/dotnet/machine-learning (pristupljeno 5. travnja 2024.)
5. Scikit-learn, (n.d.), *Metrics and Scoring: Quantifying the Quality of Predictions*, dostupno na: https://scikit-learn.org/stable/modules/model_evaluation.html (pristupljeno 5. travnja 2024.)
6. Hajian-Tilaki, K., (2013), *Receiver Operating Characteristic (ROC) Curve Analysis for Medical Diagnostic Test Evaluation*, International Journal of Medical Sciences, 10(8), pp. 604–615, dostupno na: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3755824 (pristupljeno 20. travnja 2024.)
7. Ahmed, M., (2019), *Symmetry in Machine Learning*, Symmetry, 11(1), pp. 1–13, dostupno na: https://www.mdpi.com/2073-8994/11/1/47 (pristupljeno 1. svibnja 2024.)

SAŽETAK

Ovaj rad istražuje mogućnosti izrade .NET aplikacije koristeći ML.NET tehnologiju za rješavanje problema binarne i tekstualne klasifikacije. Rad je organiziran tako da je prvo objašnjena teorijska pozadina, a zatim je u praktičnom dijelu rada opisana implementacija obrađenih metoda. U teorijskom dijelu obrađena je osnova modela strojnog učenja, s naglaskom na klasifikaciju binarnih i tekstualnih podataka. Detaljno su opisani algoritmi korišteni u radu: logistička regresija, perceptron s prosjekom, stablo odluke i naivni Bayesov klasifikator.

Praktični dio rada obuhvaća izradu interaktivne .NET aplikacije koja koristi Windows Forms tehnologiju za grafičko sučelje. Za binarnu klasifikaciju primijenjeni su algoritmi logističke regresije i perceptrona s prosjekom koristeći podatkovni skup Pima Indians Diabetes. Za tekstualnu klasifikaciju korišteni su algoritmi stabla odluke i naivnog Bayesovog klasifikatora koristeći SMS Spam Collection DataSet. Evaluacija uspješnosti modela uključuje: metrike točnosti, preciznosti i odaziva, evaluaciju performansi predikcija, krivulje radne karakteristike (ROC) i površine ispod krivulje (AUC) za binarnu klasifikaciju, te makro točnost, mikro točnost, log-gubitak, evaluaciju matrice zabune i kumulativni graf dobitka za tekstualnu klasifikaciju.

Ključne riječi: ML.NET, strojno učenje, .NET, Web Forms, klasifikacija, logistička regresija, perceptron s prosjekom, stablo odluke, naivni Bayesov klasifikator.

SUMMARY

This paper explores the possibilities of creating a .NET application using ML.NET technology to solve binary and textual classification problems. The structure includes an initial theoretical background, followed by a practical implementation of these methods. The theoretical part discusses the basics of machine learning models, focusing on the classification of binary and textual data. Detailed descriptions are provided for logistic regression, averaged perceptron, decision tree, and naive Bayes classifier algorithms.

The practical part involves creating an interactive .NET application using Windows Forms technology for the graphical interface. Logistic regression and averaged perceptron are applied to binary classification using the Pima Indians Diabetes dataset. For textual classification, decision tree and naive Bayes classifier algorithms are used, utilizing the SMS Spam Collection dataset. The evaluation of model performance includes: accuracy, precision, and recall metrics, prediction performance evaluation, receiver operating characteristic curves (ROC) and area under the curve (AUC) for binary classification, as well as macro accuracy, micro accuracy, log-loss, confusion matrix evaluation, and cumulative gain chart for text classification.

Keywords: ML.NET, machine learning, .NET, Web Forms, classification, logistic regression, averaged perceptron, decision tree, naive Bayes classifier.