TEHNIČKO VELEUČILIŠTE U ZAGREB

POLITEHNIČKI SPECIJALISTIČKI DIPLOMSKI STRUČNI STUDIJ

SPECIJALIZACIJA INFORMATIKA

Daniel Peruško

Otkrivanje spama kod e-pošte

Zagreb, studeni 2023.

Sažetak

Seminarski rad istražuje izazove vezane uz neželjenu poštu u e-pošti i primjenjuje strojno učenje, posebno klasifikaciju s potporom vektora (SVM), kao rješenje. Počinje s pregledom problema poput poplave sandučića, sigurnosnih prijetnji i gubitka vremena korisnika. U uvodnom dijelu koda koriste se biblioteke za učinkovito uvođenje i obradu podataka, implementiraju se funkcije za uređivanje i razvrstavanje, te se primjenjuje nasumično preduzorkovanje za rješavanje neuravnoteženosti klasa. Izgrađuje se SVM model, podaci se dijele, a performanse modela evaluiraju. Optimizacija hiperparametara kroz pretraživanje mreže ključan je korak u poboljšanju modela, a rezultati i analiza prikazuju uspješnost. Na kraju, korištenjem uvježbanog modela predviđa se status sljedećih pet e-poruka, naglašavajući važnost strojnog učenja u rješavanju problema neželjene pošte, poboljšavajući iskustvo korisnika i očuvavajući sigurnost e-pošte.

Sadržaj

Popis slika

1. Uvod

Neželjena pošta u e-pošti predstavlja ozbiljan problem s nekoliko ključnih izazova. Poplavljivanje sandučića, prijetnje sigurnosti i gubitak vremena korisnika su samo neki od problema s kojima se susrećemo. Strojno učenje pruža učinkovite mehanizme za suočavanje s ovim izazovima. Korištenjem tehnika poput klasifikacije s potporom vektora (SVM), možemo trenirati modele koji automatski prepoznaju neželjenu poštu. Ovi modeli analiziraju obrasce u podacima kako bi razlikovali između legitimnih i neželjenih poruka. Implementacijom strojnog učenja možemo značajno poboljšati efikasnost filtriranja neželjene pošte, čime olakšavamo korisnicima upravljanje njihovim e-poštnim iskustvom i čuvanje njihove sigurnosti.

2. Razrada uvodnog koda

Na početku potrebno je uvesti biblioteke koje će nam kasnije biti potrebne za izvršavanje programskog koda.

```
In [ ]: from sklearn.feature_extraction.text import CountVectorizer

In [ ]: import pandas as pd
        import numpy as np
        from sklearn.model_selection import train_test_split , GridSearchCV , KFold
        from sklearn.svm import SVC
        from sklearn.metrics import accuracy_score , classification_report , confusion_m
        import seaborn as sns
        import matplotlib.pyplot as plt
        import re
        import nltk
        from nltk.stem import PorterStemmer
        from sklearn import metrics
```

Slika 1. Uvoz biblioteka.

Definiramo novu varijablu „df“ te u nju učitamo testne podatke koji su spremljeni u oblaku na kojima ćemo kasnije izvršavati testiranje i treniranje. Ispisujemo prvih pet linija varijable.

```
In [ ]: df = pd.read_csv("https://raw.githubusercontent.com/Sanjay-dev-ds/spam_ham_email
        df.head()
```

Out[644]:

	Label	EmailText
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Slika 2. Uvoz podataka.

Filtriramo duplikate iz učitanih podataka.

```
In [ ]: df = df.drop_duplicates(keep='first')
```

Slika 3. Filtriranje podataka.

Zatim razvrstavamo podatke na dvije varijable.

```
In [ ]: x = df['EmailText'].values
        y = df['Label'].values
```

Slika 4. Razvrstavanje podataka.

Definiramo funkciju koja će urediti podatke. Funkcija pretvara sva slova u mala slova, briše posebne znakove, normalizira određene riječi i koristi korijene riječi umjesto izvornih riječi pomoću „Stemmer algoritma“.

```
In [ ]: porter_stemmer=PorterStemmer()
        def preprocessor(text):

            text=text.lower()
            text=re.sub("\\W"," ",text)
            text=re.sub("\\s+(in|the|all|for|and|on)\\s+"," _connector_ ",text)
            words=re.split("\\s+",text)
            stemmed_words=[porter_stemmer.stem(word=word) for word in words]
            return ' '.join(stemmed_words)
```

Slika 5. Funkcija za uređivanje podataka.

Funkcija „tokenizer“ postavlja razmak između posebnih znakova, te podjeljuje na temelju razmaka.

```
In [ ]: def tokenizer(text):
            text=re.sub("(\\W)"," \\1 ",text)
            return re.split("\\s+",text)
```

Slika 6. Funkcija tokenizer.

Uređujemo podatke pomoću funkcije „CountVectorizer“ te ih oblikujemo u matricu pojavljivanja pojedinih riječi. „Tokenizer“ se koristi za razdvajanje teksta na pojedinačne riječi. “ngram_range“ postavlja opseg analiziranih n-grama. “min_df“ postavlja minimalni prag frekvencije dokumenta za uključivanje riječi. „preprocessor“ je funkcija za predprocesiranje teksta. Na kraju, rezultat je matrica broja pojavljivanja riječi i bigrama u tekstu.

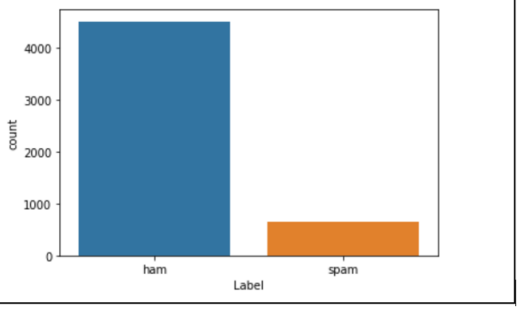
```
In [ ]: vectorizer = CountVectorizer(tokenizer=tokenizer,ngram_range=(1,2),min_df=0.006,preprocessor=preprocessor)
        x  = vectorizer.fit_transform(x)
```

Slika 7. Oblikovanje podataka u matricu.

Kreiranje stupčastog dijagrama.

```
In [ ]: sns.countplot(df['Label'])
```

```
Out[650]: <matplotlib.axes._subplots.AxesSubplot at 0x7f900f9edf50>
```



Slika 8. Stupčasti dijagram.

3. Nasumično preduzorkovanje: nasumično duplicirani primjeri u manjinskoj klasi (neželjena pošta)

Na početku potrebno je uvesti biblioteke koje će nam kasnije biti potrebne za izvršavanje programskog koda. Kreiramo instancu RandomOverSampler-a sa zadatim random seed-om. Ispisujemo broj elemenata u originalnom skupu podataka. Zatim primjenjujemo oversamplinga na skup podataka. Ispisujemo broj elemenata u modificiranom skupu podataka.

```
In [ ]: from imblearn.under_sampling import NearMiss
        from collections import Counter
        from imblearn.over_sampling import RandomOverSampler

        ros = RandomOverSampler(random_state=42)

        print('Original dataset shape', Counter(y))

        # fit predictor and target
        x,y = ros.fit_resample(x, y)

        print('Modified dataset shape', Counter(y))

        Original dataset shape Counter({'ham': 4516, 'spam': 653})
        Modified dataset shape Counter({'ham': 4516, 'spam': 4516})
```

Slika 9. Uvoz biblioteka i obrada podataka.

4. Izgradnja SVM modela i podjela podataka na skup učenja i testiranja.

U našem primjeru ćemo koristiti 80% podataka za učenje.

„test_size“ - proporcija skupa podataka koja će biti uključena u testni dio. Ovdje je postavljeno na 0.2, što znači da će 20% podataka biti korišteno za testiranje. Postavljanje „random_state = 0“ osigurava da će podjela biti ista svaki put kad pokreneš kod.

```
In [ ]: x_train , x_test , y_train , y_test   = train_test_split(x, y, test_size =0.2,random_state = 0)
```

Slika 10. Definiranje parametra i sortiranje.

Izgradimo model i dodijelimo podatke za učenje.

```
In [ ]: model    = SVC(C =1,kernel = "linear" )
        model.fit(x_train,y_train)

Out[653]: SVC(C=1, kernel='linear')
```

Slika 11. Izgradnja modela i podjela podatka.

Koristimo “accuracy_score“ kako bi izračunali točnost modela na testnom skupu i zatim prikazujemo točnost kao postotak. Točnost je omjer ispravno klasificiranih instanci u odnosu na ukupan broj instanci.

Ispis rezultata točnosti.

```
In [ ]: accuracy = metrics.accuracy_score(y_test, model.predict(x_test))
        accuracy_percentage = 100 * accuracy
        accuracy_percentage

Out[654]: 98.83785279468734
```

Slika 12. Postotak točnosti modela.

5. Optimizacija hiperparametara pomoću CV-a pretraživanja mreže

Definiramo parametre koje želite optimizirati i vrijednosti koje želimo ispitati. Postavimo KFold unakrsna validaciju s 2 preklopa, inicijaliziramo SVM model i GridSearchCV objekt, te Izvodimo pretraživanja na skupu za učenje.

```
In [ ]: params  = {"C":[0.2,0.5] , "kernel" : ['linear', 'sigmoid'] }

        cval = KFold(n_splits = 2)
        model =  SVC();
        TunedModel = GridSearchCV(model,params,cv= cval)


        TunedModel.fit(x_train,y_train)
```

```
Out[657]: GridSearchCV(cv=KFold(n_splits=2, random_state=None, shuffle=False),
                       estimator=SVC(),
                       param_grid={'C': [0.2, 0.5], 'kernel': ['linear', 'sigmoid']})
```

Slika 13. Optimizacija hiperparametara pomoću CV-a pretraživanja mreže.

Prikažemo izvješće pretraživanja.

Precision - broj ispravnih pozitivnih predikcija podijeljen s ukupnim brojem predviđenih pozitivnih. Mjeri koliko od predviđenih pozitivnih instanci zaista jesu pozitivne.

Recall - Broj ispravnih pozitivnih predikcija podijeljen s ukupnim brojem stvarnih pozitivnih. Mjeri koliko od stvarnih pozitivnih instanci je ispravno predviđeno.

F1 mjera - Harmonijska sredina preciznosti i odziva. Pruža ravnotežu između preciznosti i odziva.

Support - Broj stvarnih pojava klase u određenom skupu podataka. Pomaže razumjeti broj stvarnih instanci za svaku klasu.

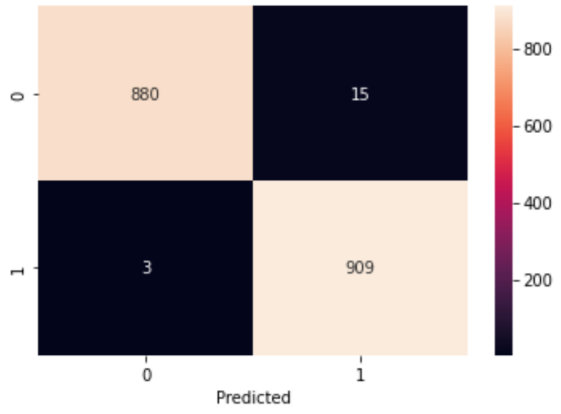
```
In [ ]: print(classification_report(y_test,TunedModel.predict(x_test)))
```

	precision	recall	f1-score	support
ham	1.00	0.98	0.99	895
spam	0.98	1.00	0.99	912
accuracy			0.99	1807
macro avg	0.99	0.99	0.99	1807
weighted avg	0.99	0.99	0.99	1807

Slika 14. Rezultat obrade.

Heatmap matrice zabune vizualno prikazuje performanse vašeg klasifikatora na testnom skupu podataka.

```
In [ ]: sns.heatmap(confusion_matrix(y_test,TunedModel.predict(x_test)),annot = True , fmt ="g")
plt.xlabel("Predicted")
plt.show("Actual")
plt.show()
```



Slika 15. Heatmap matrice zabune.

Ovaj model nam prikazuje rezultat analize.

- Gornji lijevi kvadrant nam prikazuje instance gdje je stvarna klasa pozitivna (npr. spam) i model ju je ispravno predvidio kao pozitivnu.
- Donji lijevi kvadrant nam prikazuje instance gdje je stvarna klasa negativna (npr. ham) i model ju je ispravno predvidio kao negativnu.
- Gornji desni kvadrant nam prikazuje instance gdje je stvarna klasa negativna, ali model ju je pogrešno predvidio kao pozitivnu.

- Donji desni kvadrant nam prikazuje instance gdje je stvarna klasa pozitivna, ali model ju je pogrešno predvidio kao negativnu.

6. Korištenje modela sa vlastitim primjerima

Koristeći uvježbani model, predvidjeti jesu li sljedećih pet e-poruka spam ili korisnik. U varijablu „mails“ upisujemo primjere koje ćemo obraditi. Ispitujemo svaki primjer pomoću ranije treniranog modela. Na kraju ispisujemo primjere i pretpostavku.

```
In [ ]: mails = ["Hey, you have won a car !!!!. Conrgratzz"
                ,"Dear applicant, Your CV has been recieved. Best regards"
                ,"You have received $1000000 to your account"
                ,"Join with our whatsapp group"
                ,"Kindly check the previous email. Kind Regard"]
        for mail in mails:
          is_spam = TunedModel.predict(vectorizer.transform([mail]).toarray())
          print(mail + " : "+ is_spam)

        ['Hey, you have won a car !!!!. Conrgratzz : spam']
        ['Dear applicant, Your CV has been recieved. Best regards : ham']
        ['You have received $1000000 to your account : spam']
        ['Join with our whatsapp group : ham']
        ['Kindly check the previous email. Kind Regard : ham']
```

7. Zaključak

Ovaj seminar istražuje izazove povezane s neželjenom poštom u e-pošti i predstavlja primjenu strojnog učenja, posebno klasifikacije s potporom vektora (SVM), kao rješenje. Počeli smo s pregledom problema poput poplave sandučića, sigurnosnih prijetnji i gubitka vremena korisnika. U uvodnom kodu smo koristili biblioteke za učinkovito uvođenje i obradu podataka te implementirali funkcije za uređivanje i razvrstavanje. Primijenili smo nasumično preduzorkovanje za rješavanje problema neuravnoteženosti klasa. Izgradili smo SVM model, podijelili podatke te evaluirali performanse modela. Optimizacija hiperparametara kroz pretraživanje mreže predstavlja ključni korak u poboljšanju modela, što smo prikazali rezultatima i analizom. Korištenjem uvježbanog modela, predvidjeli smo status sljedećih pet e-poruka. Ovaj rad naglašava važnost strojnog učenja u rješavanju problema neželjene pošte, pružajući učinkovito rješenje za poboljšanje iskustva korisnika i očuvanje sigurnosti e-pošte.

_____

Daniel Peruško

8. Literatura

- https://medium.com/@dssc2022yt/spam-ham-email-detection-d548a6d5869e (21.11.2023)
- https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python(21.11.2023)
- https://www.yourdatateacher.com/2021/05/19/hyperparameter-tuning-grid-search-and-random-search/ (22.11.2023)