**Uvod**

Analiza sentimenta važan je alat za razumijevanje emocija u tekstovima, komentarima na društvenim mrežama, govorima i sl. Koristi se u mnogim područjima ljudskog djelovanja poput marketinga, društvenim mrežama, analiza političkih govora i dr. kako bi se razumjela subjektivna priroda pojedinog sadržaja. S obzirom na rapidan rast količine sadržaja koja se javlja na internetu, potreba za analizom sentimenta raste proporcionalno.

U ovom smo projektu odlučili koristiti postojeći skup podataka te ga anotirati i zatim testirati tri različita algoritma strojnog učenja. Također smo jasno definirali smjernice kojima se svaki anotator vodio pri anotiranju rečenica. Ovaj projekt će pružiti uvid u važnost analize sentimenta, pokazati korake od prikupljanja do evaluacije modela te istaknuti potencijal koji analiza sentimenta nosi u interpretaciji ljudskih emocija.

**Definicija problema**

U ovom projektu želimo izgraditi i evaluirati model za analizu sentimenta na hrvatskom jeziku koristeći recenzije avanturističkih filmova. Cilj je razviti model koji može precizno klasificirati rečenice prema sentimentu (pozitivan, negativan, neutralan, mješovit), unatoč neravnoteži u distribuciji oznaka i složenosti prirodnog jezika uz pomoć anotatora i zadanih smjernica.

**Skup podataka**

Odlučili smo stvoriti skup podataka na temelju recenzija avanturističkih filmova s poveznice <https://www.recenzijefilmova.com/adventure/>. Na stranici je ponuđen veliki broj filmova i za svaki su dostupne recenzije koje smo koristili kako bi stvorili skup podataka za treniranje modela. Prvi je korak bilo prikupljanje dovoljne količine rečenica. Prikupili smo 3063 rečenice te smo ih nakon toga prebacili u Excel dokument gdje je svaka rečenica bila u zasebnom retku. Prvotno smo mislili koristiti samo pozitivan, negativan i neutralan sentiment, no nakon što smo anotirali testni skup podataka od 150 rečenica shvatili smo kako dosta rečenica ima i negativan i pozitivan sentiment. Nakon toga smo dogovorom odlučili dodati i mješoviti sentiment te smo stvorili smjernice za anotiranje.

U smjernicama smo definirali svaki sentiment te smo dodali i primjere rečenica kako bi svaki član grupe u slučaju nejasnoća mogao provjeriti koji sentiment se veže uz koju rečenicu. Pozitivan sentiment ide uz rečenice koje izražavaju zadovoljstvo, sreću i sve druge pozitivne emocije. Negativan sentiment ide uz rečenice koje izražavaju nezadovoljstvo, tugu i ostale negativne emocije. Neutralan sentiment ide uz rečenice koje nemaju izražen sentiment, objektivne su, opisne ili činjenične. Mješoviti sentiment ide uz rečenice koje sadrže pozitivan i negativan sentiment i rečenice koje izražavaju konfuziju i dvosmislenost.

Svaka osoba iz grupe je bila uključena u proces anotiranja. Podijelili smo skup na jednake dijelove tako da je svaku rečenicu anotiralo troje članova grupe kako bi lakše došli do dogovora oko konačne anotacije za rečenicu. Svaki je član anotirao oko 2297 rečenica. Na kraju smo ručno prolazili sve anotacije te smo popunjavali stupac u tablici koja sadržava konačan sentiment. Oko većine rečenica smo se uspjeli složiti, no bilo je i nekih gdje je svaki član upisao drugačiji sentiment. To smo riješili tako da smo ponovno razmotrili rečenicu te smo zajedno došli do zaključka oko odgovarajućeg sentimenta.

Nakon provođenja anotacije kao rezultat smo dobili sljedeće:

poz 1217

neut 1101

mix 414

neg 331

**Metodologija**

Prije nego što smo krenuli sa izgradnjom modela za analizu sentimenta, bilo je potrebno pripremiti podatke i izračunati fleiss – kappa indeks. Fleiss – kappa indeks nam govori koliko pouzdano tri ili više ocjenjivača mjere istu stvar. Prvi korak je bio pretvaranje prethodno anotiranih recenzija, odnosno labela svake rečenice, u brojčane vrijednosti kako bismo ih mogli unijeti u funkciju koja računa prethodno navedeni index. Svakoj labeli dodali smo njen odgovarajući broj:

* poz = 0
* neg = 1
* neut = 2
* mix = 3

To smo postigli koristeći programski jezik python na sljedeći način:

import os

import pandas as pd from statsmodels.stats.inter\_rater

import aggregate\_raters, fleiss\_kappa

!wget <https://raw.githubusercontent.com/BoViNiMa/OPJ/main/Finalne%20anotacije.xlsx>

df = pd.read\_excel('/content/Finalne anotacije.xlsx', index\_col=0) df.head()

df["poz, neg, neut, mix"].unique()

string\_to\_int = {'neg': 0, 'neut': 1,'poz': 2, 'poz ': 2, 'mix ': 3, 'mix': 3, 'pot': 2, 'neuz': 1, 'neur': 1, 'nuet':1}

df = df.replace(to\_replace=string\_to\_int)

df["poz, neg, neut, mix"].unique()

df\_data = df[["Nina","Borna","Viktor"]] df\_dropped = df\_data.dropna() df\_dropped.head().values.tolist()

agg,cat = aggregate\_raters(df\_dropped.values.tolist())

Agg

fleiss\_kappa(agg)

Fleiss – kappa index koji smo dobili je 0.6478310562687184

Također, prije treniranja samih modela/metoda; svaku recenziju i anotaciu (ona koja prevladava između 3 dodjeljenih anotacija) pohranili smo u zasebne liste na sljedeći način:

texts = []

labels = []

with open('/content/Anotacije.tsv') as input\_file:

for i, line in enumerate(input\_file):

line = line.strip()

print(i, line, line.rsplit("\t"))

text, label = line.split("\t")

texts.append(text)

labels.append(label)

print(i, text, label)

Sve labele koje su bile krivo unešene ispravili smo i spremili ih tako da su napisane malim slovima ovako:

label\_replace\_dict = { "neur": "neut", "nuet": "neut", "pot": "poz", "neur": "poz" }

lower\_cased\_label = [x.lower() for x in labels]

lower\_cased\_label

labels = [label\_replace\_dict[x] if x in label\_replace\_dict else x for x in lower\_cased\_label]

set(labels)

Nakon čišćenja, transformiranja labela i dobivenog indeksa, krenuli smo s analizom koristeći tri različita algoritma strojnog učenja: XGBoost, Random Forest i SVM algoritam (također radeći s labelama koje su transformirane u numerički format). Cilj je bilo usporediti učinak ovih algoritama i identificirati onaj koji postiže najbolje rezultate na našem skupu podataka.

1. **Metoda 1 - XGBoost**

XGBoost je algoritam strojnog učenja s distribuiranim gradijentnim stablom odlučivanja (GBDT).

Ovaj smo algoritam tenirali koristeći sljedeći kod:

import xgboost as xgb

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

texts\_train, texts\_test, labels\_train, labels\_test = train\_test\_split( texts, labels, test\_size=0.2, random\_state=42)

"Size of train:",len(texts\_train), "test set", len(texts\_test)

vectorizer = TfidfVectorizer()

X\_train = vectorizer.fit\_transform(texts\_train)

X\_test = vectorizer.transform(texts\_test)

model = xgb.XGBClassifier(n\_jobs=-1)

model.fit(X\_train, labels\_train)

Točnost i F1-score su sljedeći:  
Točnost: 55.95%  
F1-Score: 42.12%

Koje smo dobili pomoću koda:

from sklearn.metrics import accuracy\_score, f1\_score

predictions = model.predict(X\_test)

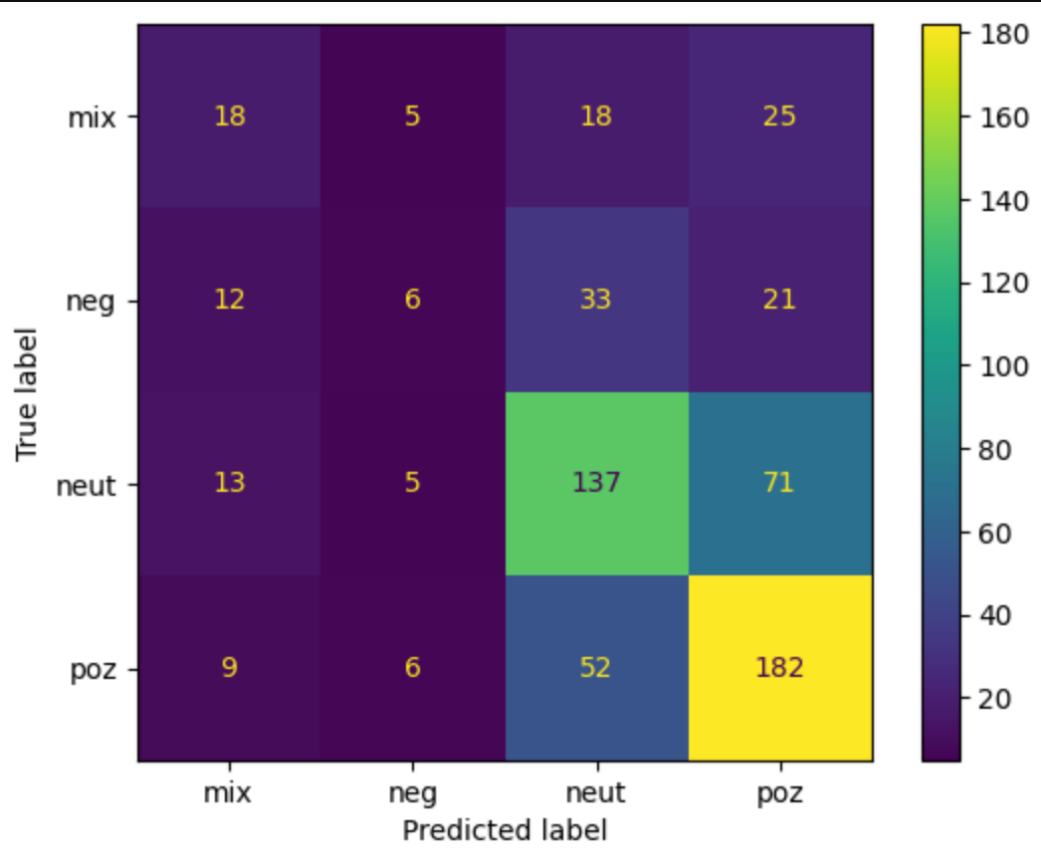
accuracy = accuracy\_score(labels\_test, predictions)

print("Accuracy: {:.2f}%".format(accuracy \* 100))

f1 = f1\_score(labels\_test, predictions, average='macro')

print("F1-Score: {:.2f}%".format(f1 \* 100))

Confusion matix za ovaj algoritam izgleda ovako:



1. **Metoda 2 – Random forrest classifier (RFC)**

U RFC metodi, koristi se više više stabla odlučivanja. Kombiniraju se rezultati stabala odlučivanja kako bi se postigao jedan rezultat.

Ovu smo metodu napravili koristeći sljedeći kod:

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

texts\_train, texts\_test, labels\_train, labels\_test = train\_test\_split( texts, labels, test\_size=0.2, random\_state=42)

"Size of train:",len(texts\_train), "test set", len(texts\_test)

vectorizer = TfidfVectorizer()

X\_train = vectorizer.fit\_transform(texts\_train)

X\_test = vectorizer.transform(texts\_test)

model\_rfc = RandomForestClassifier(n\_estimators=100) model\_rfc.fit(X\_train, labels\_train)

Točnost i F1-score su sljedeći:  
Točnost: 56.61%  
F1-Score: 36.53%

Koje smo dobili pomoću koda:

from sklearn.metrics import accuracy\_score, f1\_score

predictions\_rfc = model.predict(X\_test)

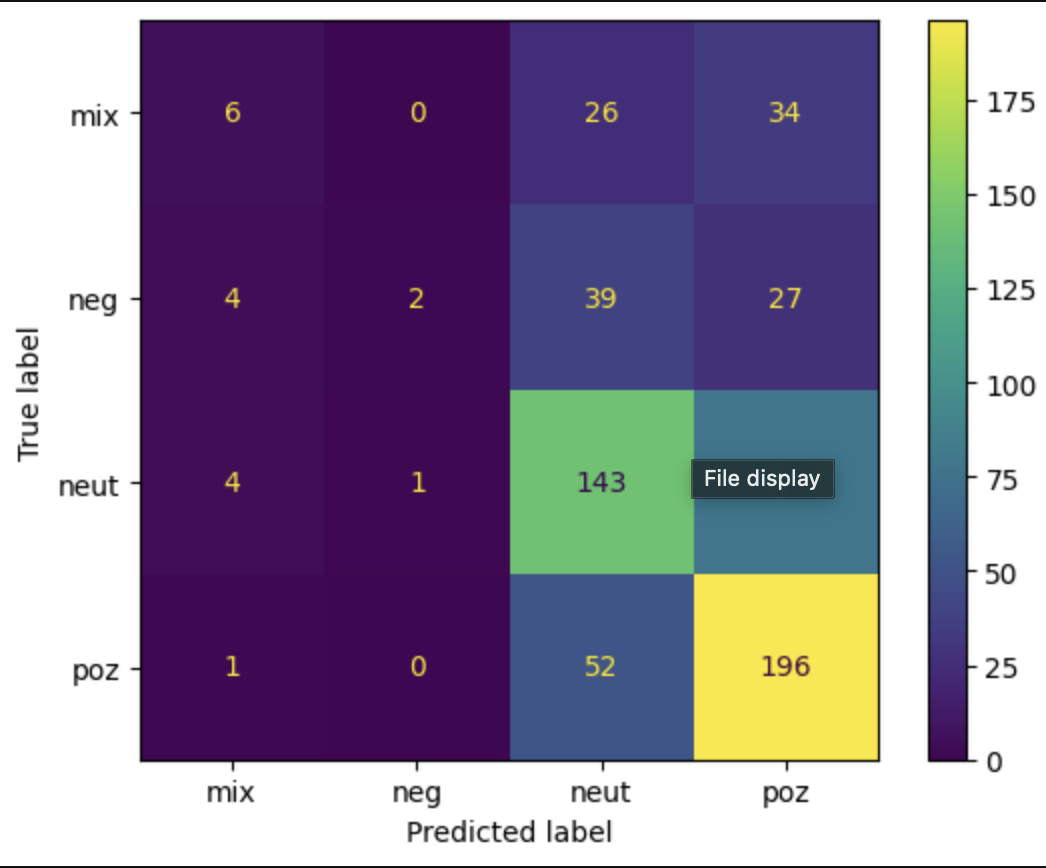
accuracy = accuracy\_score(labels\_test, predictions\_frc)

print("Accuracy: {:.2f}%".format(accuracy \* 100))

f1 = f1\_score(labels\_test, predictions\_rfc, average='macro')

print("F1-Score: {:.2f}%".format(f1 \* 100))

Confusion matrix za ovaj algoritam izgleda ovako:



1. **Metoda 3 – Support vector machine (SVM)**

Svm je algoritam strojnog učenja koji koristi modele nadziranog učenja za rješavanje složenih problema klasifikacije, regresije i otkrivanja izvanrednih vrijednosti izvođenjem optimalnih transformacija podataka koje određuju granice između podatkovnih točaka na temelju unaprijed definiranih klasa, oznaka ili izlaza.

Ovu smo metodu napravili koristeći sljedeći kod:

from sklearn.svm import SVC

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

texts\_train, texts\_test, labels\_train, labels\_test = train\_test\_split( texts, labels, test\_size=0.2, random\_state=42)

"Size of train:",len(texts\_train), "test set", len(texts\_test)

vectorizer = TfidfVectorizer()

X\_train = vectorizer.fit\_transform(texts\_train)

X\_test = vectorizer.transform(texts\_test)

model\_svc = SVC()

model\_svc.fit(X\_train, labels\_train)

Točnost i F1-score su sljedeći:  
Točnost: 61.01%  
F1-Score: 40.95%

Koje smo dobili pomoću koda:

from sklearn.metrics import accuracy\_score, f1\_score

predictions\_svc = model.predict(X\_test)

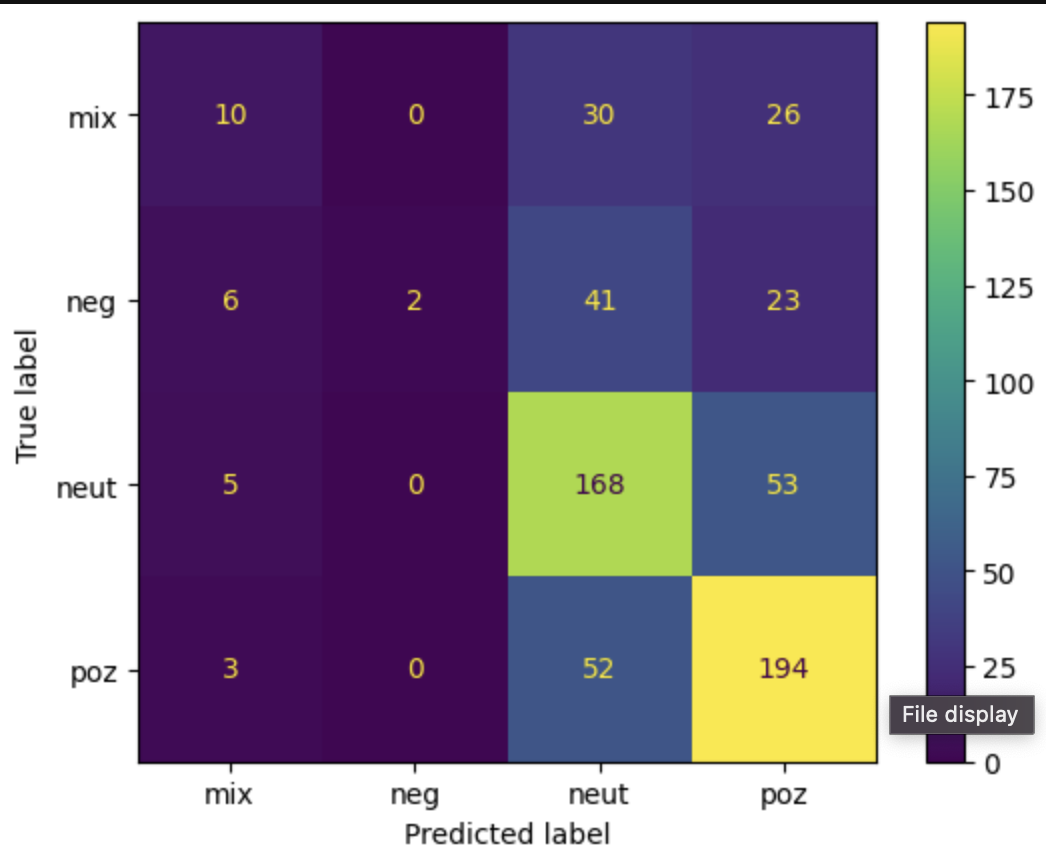
accuracy = accuracy\_score(labels\_test, predictions\_svc)

print("Accuracy: {:.2f}%".format(accuracy \* 100))

f1 = f1\_score(labels\_test, predictions\_svc, average='macro')

print("F1-Score: {:.2f}%".format(f1 \* 100))

Confusion matrix za ovaj algoritam izgleda ovako:



Najveću dobivenu točnost ima support vector machine (SVC) algoritam pa smo iz tog razloga odlučili odabrati njega za izradu demo aplikacije. Kako bi aplikacija funkcionirala potrebno je napraviti pickle file i vectorizer file s ekstenzijom .pkl, što smo napravili ovako:

import pickle

filename = 'svc\_model.pkl'

with open(filename, 'wb') as f:

pickle.dump(model\_svc, f) print(f"SVC model saved to: {filename}")

filename2 = 'vectorizer.pkl'

with open(filename2, 'wb') as f:

pickle.dump(vectorizer, f) print(f"Vectorizer saved to: {filename2}")

Demo aplikaciju za analizu sentimenta, napravili smo koristeći Hugging Face platformu. Hugging Face je popularna platforma koja nudi širok spektar resursa za strojno učenje. Kod aplikacije izgleda ovako:

import numpy as np

import os

import gradio as gr

import xgboost as xgb

import pickle

from sklearn.feature\_extraction.text import TfidfVectorizer

os.environ["WANDB\_DISABLED"] = "true"

label2id = {

0: "negative",

1: "neutral",

2: "positive",

3: "mixed"}

vectorizer\_file\_name = 'vectorizer.pkl'

xgb\_model\_loaded = pickle.load(open(model\_file\_name, "rb"))

vectorizer\_loaded = pickle.load(open(vectorizer\_file\_name, "rb"))

def predict\_sentiment(predict\_texts):

predictions\_loaded = xgb\_model\_loaded.predict(vectorizer\_loaded.transform([predict\_texts]))

print(predictions\_loaded)

return label2id[predictions\_loaded[0]]

interface = gr.Interface(

fn=predict\_sentiment,

inputs='text',

outputs=['text'],

title='Croatian Movie reviews Sentiment Analysis',

examples= ["Volim kavu","Ne volim kavu"],

description='Get the positive/neutral/negative sentiment for the given input.'

)

interface.launch(inline = False)

**Rezultati i rasprava**

Ovaj projekt istražio je proces izgradnje skupa podataka za analizu sentimenta na primjeru 3063 rečenice o recenzijama filmova. Od prikupljanja do njihove anotacije, postavili smo bazu za učinkovitu analizu sentimenta. Procesom anotacije, došlo smo do preciznosti Fleissovom kappom od 0.65. Korištenjem triju algoritama – XGBoost, Random Forest i Support Vector Machine (SVM). SVM je pokazao najbolju učinkovitost, s točnošću od 61.01% i F1-scoreom od 40.95%. Međutim, unatoč tome što su rezultati solidni, F1-score manji od 50% ukazuje na to da algoritam ne daje najbolje vrijednosti za klasifikaciju te se, konzekventno, ovi rezultati mogu poboljšati daljnjim treniranjem modela kroz više generacija, optimizacijom hiperparametara i postizanjem boljeg Fleissovog kappa indeksa. Još jedan od čimbenika koji je mogao unaprijediti performanse modela i pospješiti preciznost je uravnoteženje broja instanci za svaku kategoriju sentimenta (jednako pozitivnih, negativnih itd. rečenica). Izradom demo aplikacije na platformi Hugging Face, demonstrirali smo praktičnu primjenu modela. Aplikacija omogućuje korisniku da unese tekst i dobije vrijednost sentimenta kao povratnu informaciju, čime smo pokazali kako se naši rezultati mogu koristiti u stvarnom svijetu.

**Zaključak**

Projekt je uspješno prikazao cijeli proces analize sentimenta, od samog prikupljanja do treniranja i implementacije modela. Uvelike smo napredovali u razumijevanju analize sentimenta te stvorili bazu za daljnja istraživanja i razvoj u danom području.

**Linkovi**

**GitHub:**

<https://github.com/BoViNiMa/OPJ/tree/main>

**Prikupljeni skup podataka:**

<https://github.com/BoViNiMa/OPJ/blob/main/Re%C4%8Denice>

**Anotirani skup podataka:**

https://github.com/BoViNiMa/OPJ/blob/main/Finalne%20anotacije.xlsx

**Smjernice za anotiranje:**

https://github.com/BoViNiMa/OPJ/blob/main/Upute%20za%20anotiranje%2C%20BoViNiMa.docx

**Projekt:**

<https://github.com/BoViNiMa/OPJ/tree/main/projekt>

**Demo:**

<https://huggingface.co/spaces/SuperViktor2/OPJ-demo>

**Prezentacija:**