**Преглед литературе**

У овом прегледу, представљени су различити приступи и алгоритми за класификацију твитова на основу сентимента, како би се истражила њихова ефикасност и примена у пракси.

1. ***Сентимент анализа путем мултислојног перцептрона оптимизованог мета-хеуристиком (Албоанеен, Тианфилд, Џанг, 2017)*** У овом раду, предложен је нови приступ анализи твитова који се састоји од две главне фазе: селекција карактеристика и класификација твитова. У првој фази, коришћена је међусобна информација (MI) за одабир најбољег скупа карактеристика како би се смањила димензија карактеристика. У другој фази, мета-хеуристички алгоритам је коришћен за оптимизацију тежина и пристрасности мултислојних перцептрона (MLPs) и затим имплементиран за класификацију сентимената твитова.
2. **Експериментално поређење техника класификације текста (Лакотија, Бресон, 2018)** Рад се фокусира на поређење перформанси актуелних техника класификације текста, од стандардних модела базираних на SVM-у, статистичким моделима и мултислојним перцептронима (MLP) до недавно побољшаних дубоких учења попут конволуционих неуралних мрежа и њихове фузије са теоријом графова. Резултати су показали да граф конволуционе неуралне мреже имају бољу или сличну тачност теста у поређењу са стандардним конволуционим неуралним мрежама, моделима базираним на SVM-у и статистичким моделима. Изненађујуће, једноставнији MLP модели и даље надмашују недавно уведене дубоке учења упркос мањем броју параметара.
3. **Преглед алгоритама за класификацију текста: од текста до предикција (Гаспарето, Маркуцо, Зангари, Албарели, 2022)** Рад пружа преглед недавних модела за класификацију текста са нагласком на проток података, од сировог текста до излазних етикета. Истичу се разлике између ранијих метода и новијих, метода базираних на дубоком учењу како у функционисању, тако и у начину трансформације улазних података. Такође, рад даје преглед сетова података за енглески језик и нуди инструкције за синтезу нових мулти-етикетних сетова података.
4. **Сентимент анализа користећи логистичку регресију и ефективну хеуристику за оцену речи (Тјаги, Шарма, 2018)** У овом раду, логистичка регресија је коришћена као класификатор и униграм као вектор карактеристика. Коришћена је техника унакрсне валидације како би се побољшала тачност. Такође је представљена хеуристика ефективне оцене речи како би се убрзао процес класификације сентимената.
5. **Мултислојни перцептрон и неуралне мреже (Попеску, Балас, Переску-Попеску, Масторакис, 2009)** Рад истражује различите варијације броја слојева неурона и функција активације које се користе за решавање линеарно неодвојивих проблема. Представљени су методи попут момента и примене променљиве стопе учења како би се убрзало учење.
6. **Логистичка регресија (Nick & Campbell, 2007)** Овај рад истражује логистичку регресију као један од најчешће коришћених статистичких модела у медицинским истраживањима. Логистички регресиони модели се користе за проучавање ефеката предиктора на категоријалне исходе, обично бинарне природе. Рад обухвата и једноставне и вишеструке бинарне логистичке моделе, укључујући разматрање интеракција, категоријалних и континуалних предиктора, као и процену доброте прилагођености модела.

**Опис проблема**

У овом истраживању, фокусирамо се на примену Classifier-а у анализи и предвиђању сентимента текстуалних података. Сентимент анализа представља изазовну област у обради природног језика, која има за циљ идентификацију и класификацију емоционалног тона у тексту. Претходећи овом истраживању, посвећена је пажња детаљном претпроцесирању текстуалних података и изградњи модела који омогућава предвиђање сентимента на основу анализе уноса текстуалних порука. Кроз овај рад, истражујемо методологију и изазове у примени MLP Classifier-а у сентимент анализи.

**Опис скупа података**

Наш скуп података састоји се од базе *sentiment\_analysis.csv* проширене ручно унесеним подацима из базе *sentimentNewest.csv*. Састоји се од следећих атрибута:

Kolona,Tip podataka,Opis,Dozvoljene vrednosti

Year,Integer,Godina kada je tweet objavljen,2010-2023

Month,Integer,Mesec kada je tweet objavljen,1-12

Day,Integer,Dan kada je tweet objavljen,1-31

Time of Tweet,String,Vreme dana kada je tweet objavljen,"morning, noon, night"

text,String,Tekst tweet-a,Bilo koji tekst

sentiment,String,Sentiment izražen u tweet-u,"positive, negative, neutral"

Platform,String,Platforma na kojoj je tweet objavljen,"Facebook, Twitter, Instagram"

Има 746 инстанци, од којих су: 316 позитивног, 259 неутралног и 171 негативног сентимента. То нам указује да скуп података није добро избалансиран, пошто има значајно мање негативног сентимента, што ће сигурно утицати на резултате модела машинског учења.

Пре спајања база избацили смо дупликате из *sentiment\_analysis.csv* скупа и над подацима о класи и платформи из *sentimentNewest.csv,* који су били неправилно груписани, извршили смо агрегацију.

df = df.drop\_duplicates(subset=['text'])

# Uklanjanje nepotrebne praznine s početka i kraja svake oznake

df\_newest['sentiment'] = df\_newest['sentiment'].str.strip()

df\_newest['sentiment'] = df\_newest['sentiment'].str.lower()

# Zamena različitih varijanti s jednom standardnom

df\_newest['sentiment'].replace({'positive': 'positive', 'negative': 'negative', 'neutral': 'neutral'}, inplace=True)

# Uklanjanje nepotrebne praznine s početka i kraja svake oznake

df\_newest['Platform'] = df\_newest['Platform'].str.strip()

# Zamena različitih varijanti s jednom standardnom

df\_newest['Platform'].replace({'Facebook': 'Facebook', 'Instagram': 'Instagram', 'Twitter': 'Twitter'}, inplace=True)

**ЕДА –** **Експлораторна анализа података**

ЕДА је процес истраживања, анализирања и визуализације скупа података ради разумевања његових карактеристика, идентификације образаца и припреме за даљу анализу. Овај корак је кључан јер омогућава увид у податке пре примене сложенијих аналитичких техника, као и идентификацију потенцијалних проблема или аномалија које захтевају додатно истраживање. ЕДА укључује технике као што су статистичке методе, визуализација података, испитивање расподеле, идентификација недостајућих вредности и генерисање хипотеза.

Укључене библиотеке су Pandas за рад са табеларним подацима, Seaborn и Matplotlib за визуализацију података, NumPy за рад са нумеричким подацима и Plotly Express за интерактивну визуализацију.

***Основне статистичке информације о нумеричким колонама***

У нашем скупу података имамо 746 узорака. Просечна година објављивања твита је отприлике 2020.70, са стандардном девијацијом од отприлике 2.99. Најмања година је 2010, док је највећа 2023. Половина података има годину објављивања до 2023, док је три четвртине података до 2023. Што се тиче дана и месеца објављивања њихова средња вредност је 5.9 односно 15.8 што је близу средине њихових дозвољених вредности 1-12 и 1-31.

На основу расподеле година можемо да закључимо да број објава расте са годином, и то у три блока са сличним бројевима објава. Први су најстарије године 2010-2014 где има од 3 до 7 објава. Затим 2015-2022 са 29 до 55 објава. На крају усамљена је 2023 са 375 објава.

По месецима највише објава у нашој бази има у Јануару и Фебруару, а затим у Августу и Септембру, што може указати на период око празника, распуста и летовања/зимовања. По дану у месецу - највише је објава постављено у средини месеца а најмање на почетку и на крају месеца.

***Дужина текста***

Користећи библиотеку Plotly Express, извршили смо поделу колоне “text” по броју карактера на укупно 100 бинова. Највећи број објава састоји се од 50 до 69 карактера.

***Корелација између улазних атрибута и излазне променљиве***

На основу анализе средње вредности дужине текста по сентименту, уочава се да је текст у просеку мало краћи у случају неутралног сентимента у поређењу са позитивним и негативним. Што се тиче доба дана, примећено је да људи показују највећу негативност ујутру, док је позитиван став најизраженији око поднева. Увече се поново јавља тенденција ка негативности. У контексту платформи, на Instagramу је најчешће забележен неутралан сентимент, на Facebookу преовладавају негативне објаве, док су корисници Twitterа најчешће позитивни.

***Међукорелације улазних атрибута***

У контексту доба дана, дужина објава је најкраћа ујутру, а затим се појачава увече, са најдужим објавама око поднева. Што се тиче платформи, није уочена значајна разлика у дужини објава. Ујутру се највише објаваљује на Facebook-у, док је на Инстаграму број објава најмањи. Увече, највише објава је на Facebook-у, док је на Твитеру најмање. Око поднева, значајно највише објава је на Твитеру, док је на Facebook-у најмање.

**Претпроцесирање података**

У овом истраживању, темељно смо обрадили текстуалне податке како бисмо припремили скуп података за анализу сентимента на друштвеним мрежама. Почели смо од провере постојања нултих вредности или празних стрингова у некој од колона. То није био случај.

Наставили смо са конверзијом свих текстуалних података у мала слова ради уједначености, а затим смо уклонили сувишне размаке с почетка и краја сваког текста ради елиминације потенцијалних несагласности у форматирању.

df['text']=df['text'].str.lower()

df['text']=df['text'].str.strip()

Након тога, идентификовали смо и уклонили URL-ове и сличне везе које се могу појавити у текстовима, користећи функцију за претрагу и брисање делова који подсећају на URL-ове или повезане ознаке. Овим кораком смо успешно издвојили само суштински текстуални садржај за даљу анализу.

def remove\_html(text: str):

     text = text.replace("href", "")

     text = text.replace("www", "")

     text = text.replace("src", "")

     text = text.replace("alt", "")

     text = text.replace("img", "")

     try:

          # kako posle slike ima naziv slike mi brisemo sve nakon http

          index\_found = text.index("http")

          return text[:index\_found]

     except ValueError:

          return text

Важан аспект обраде текста било је и уклањање честих речи и фраза, као и замена скраћеница и често коришћених израза(stopwords) њиховим пуним облицима или њихово потпуно уклањање. Овај корак је допринео бољој разумљивости текста и избегавању могућих неспоразума приликом анализе. (приложен је део кода)

def remove\_common(text: str):

     text = text.replace(" like ", " ")

     text = text.replace(" im ", " ")

     text = text.replace("im ", " ")

...

def expand(text: str):

   text = text.replace(" let's ", " let us ")

   text = text.replace("'s ", " is ")

   text = text.replace("n't ", " not ")

...

chat\_words = {

    "AFAIK": "As Far As I Know",

    "AFK": "Away From Keyboard",

    "ASAP": "As Soon As Possible",

...

def replace\_chat\_words(text):

    words=text.split()

    for i, word in enumerate(words):

        if word.upper() in chat\_words:

            words[i]=chat\_words[word.upper()].lower()

    return ' '.join(words)

import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

stop\_words=set(stopwords.words('english'))

def remove\_stopwords(text):

    words=text.split()

    filtered\_words=[word for word in words if word.lower() not in stop\_words]

    return ' '.join(filtered\_words)

#Sve funkcije su primenjene na sledeci nacin

df['text']=df['text'].apply(remove\_stopwords)

Уклањени су и интерпункцијски знакови ради олакшане даље обраде текста и избегавања потенцијалних проблема са интерпретацијом речи или фраза које садрже интерпункцијске симболе.

def remove\_punctuation(text: str):

    # Make a regular expression that matches all punctuation

    regex = re.compile('[%s]' % re.escape(string.punctuation))

    # Use the regex

    return regex.sub('', text)

Једна од кључних операција била је и замена емотикона њиховим текстуалним описима како бисмо олакшали анализу сентимента. Емотикони су преведени у језички разумљиве описе, што је допринело прецизнијој анализи сентимента на друштвеним мрежама.

import emoji

def is\_emoji(text):

    return emoji.demojize(text)

df['text'].apply(is\_emoji)

Након чишћења текста од сувишних речи, применили смо проверу правописа како бисмо исправили грешке у куцању и обезбедили тачност свих појмовних јединица. Овај корак је био кључан за осигуравање прецизности у даљој анализи.

from spellchecker import SpellChecker

# Inicijalizacija SpellChecker objekta

spell\_checker = SpellChecker()

def spell\_check\_text(text):

    words = text.split()

    corrected\_words = []

    for word in words:

        corrected\_word = spell\_checker.correction(word)

        if corrected\_word is not None:

            corrected\_words.append(corrected\_word)

        else:

            corrected\_words.append(word)  # Dodaj originalnu reč ako je nije moguće ispraviti

    corrected\_text = ' '.join(corrected\_words)

    return corrected\_text

df['text'] = df['text'].apply(spell\_check\_text)

Наредни корак у процесу претпроцесирања било је стемовање текста помоћу Портеровог стемера. Ова техника је редуковала речи на њихове основне облике, што је смањило димензионалност података и олакшало њихову интерпретацију током анализе сентимента. Такође додата је нова колона у којој се налази нови стемовани текст.

from nltk.stem import PorterStemmer

porter\_stemmer=PorterStemmer()

df['Text\_Stemmed']=df['text'].apply(lambda x: ' '.join([porter\_stemmer.stem(word) for word in x.split()]))

Након свих претходних корака поново смо проверили да ли постоје нулте вредности и празни стрингови, и како је то овог пута био случај у 4 инстанце и то у колонама са текстом, одлучили смо да те инстанце избацимо из скупа података.

У наставку, користећи CountVectorizer, трансформисали смо стемовани текст у низ вектора са бројем појављања сваке од речи које се налазе у корусу нашег скупа података. Овај корак омогућава да сваки документ буде представљен као вектор са димензијама које одговарају броју различитих речи у целом корпусу.

vectorizer = CountVectorizer()

X\_text = vectorizer.fit\_transform(data['Text\_Stemmed'])

За преостале категоричке променљиве, као што су време објављивања твита и платформа, користили смо OneHotEncoder како бисмо их трансформисали у бинарне векторе. Овај поступак омогућава моделу да посматра ове променљиве као независне категорије, чиме се спречава увођење лажних веза између категорија које немају нумерички поредак.

time\_encoder = OneHotEncoder()

X\_time = time\_encoder.fit\_transform(data[['Time of Tweet']])

platform\_encoder = OneHotEncoder()

X\_platform = platform\_encoder.fit\_transform(data[['Platform']])

Нумеричке податке, као што су година, месец и дан, нормализовали смо користећи StandardScaler. Овај корак је важан јер омогућава да све нумеричке особине буду на истој скали, чиме се избегава доминација променљивих са већим нумеричким распоном и обезбеђује боље перформансе модела.

X\_numeric = data[['Year', 'Month', 'Day']]

scaler = StandardScaler()

X\_numeric\_scaled = scaler.fit\_transform(X\_numeric)

На крају, све ове трансформисане податке смо спојили у један датасет X користећи hstack. Овај датасет садржи све улазне атрибуте неопходне за тренирање модела, укључујући и препроцесиране текстуалне, категоричке и нумеричке особине. Истовремено, формирали смо циљни сет y, који садржи излазну променљиву - сентимент.

X = hstack((X\_text, X\_time, X\_platform, X\_numeric\_scaled))

y = data['sentiment']

Овако припремљени подаци спремни су за даљу анализу и тренирање модела машинског учења за анализу сентимента.

**Подела на тренинг и тест**

Поделили смо податке на тренинг и тест сетове како бисмо могли да тренирамо и евалуирамо модел. За овај корак користили смо функцију train\_test\_split из библиотеке scikit-learn, која омогућава насумично дељење података на два скупа - тренинг и тест сет.

Параметар test\_size=0.3 означава да ће 30% података бити издвојено за тест сет, док ће преосталих 70% бити коришћено за тренинг модела. Параметар random\_state=42 осигурава да је дељење података репродуктивно, тј. да ће сваки пут када извршимо овај код подела бити иста.

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

**Вишеслојни перцептрон (MLPC)**

Вишеслојни Перцептрон (MLPC) класификатор је врста неуронске мреже која се користи за класификацију, регресију и друге задатке машинског учења. Састоји се од више слојева перцептрона, укључујући један улазни слој, један или више скривених слојева и један излазни слој. Сваки перцептрон у мрежи је повезан са свим перцептронима из претходног и наредног слоја, формирајући потпуно повезану мрежу.

***Структура МЛП Класификатора:***

Улазни слој: Прима улазне податке и прослеђује их скривеним слојевима. Број неурона у улазном слоју одговара броју атрибута у подацима.

Скривени слојеви: Један или више слојева који обављају комплексну обраду података. Сваки скривени слој садржи одређен број неурона, који су повезани са активацијом функција (као што су ReLU, сигмоид или tanh). Скривени слојеви омогућавају мрежи да научи и моделира сложене обрасце у подацима.

Излазни слој: Последњи слој у мрежи који генерише коначне прогнозе. Број неурона у излазном слоју зависи од броја класа у класификационом проблему. На пример, за трокласни проблем, излазни слој ће имати три неурона.

***Обучавање МЛП Класификатора:***

Форвард пропагација: Улазни подаци се прослеђују кроз мрежу, слој по слој, све до излазног слоја. Сваком неурону се примењује активациона функција, чиме се добијају излазне вредности за сваки слој.

Беквард пропагација: Након форвард пропагације, израчуната грешка између предвиђених и стварних вредности се пропагира уназад кроз мрежу. Тежине и преклопи (биаси) се ажурирају коришћењем алгоритма као што је стохастички градијентски спуст (SGD) или Adam, како би се минимизовала грешка.

Епохе: Процес форвард и беквард пропагације се понавља током више епоха, што омогућава мрежи да постепено учи и побољшава своје перформансе.

***Примена МЛП Класификатора у сентимент анализи:***

MLP класификатор је веома користан за сентимент анализу јер може да научи сложене обрасце у текстуалним подацима, што му омогућава да прецизно предвиди емоције и расположења у текстовима. Захваљујући својој способности да моделира нелинеарне релације између карактеристика, MLP класификатор може да пружи боље резултате у поређењу са једноставнијим моделима, попут логистичке регресије. Такође, због своје флексибилности, може се прилагодити различитим типовима података и проблемским ситуацијама, што га чини моћним алатом у анализи и интерпретацији сложених људских емоција.

***Тренинг модела***

# Initialize and train the model

model = MLPClassifier(max\_iter=1000, random\_state=42)

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

***Евалуација модела – Тренинг сет***

# Predict sentiment on train set

y\_pred = model.predict(X\_train)

# Confusion matrix

cm = confusion\_matrix(y\_train, y\_pred, labels=model.classes\_)

cmd = ConfusionMatrixDisplay(cm, display\_labels=model.classes\_)

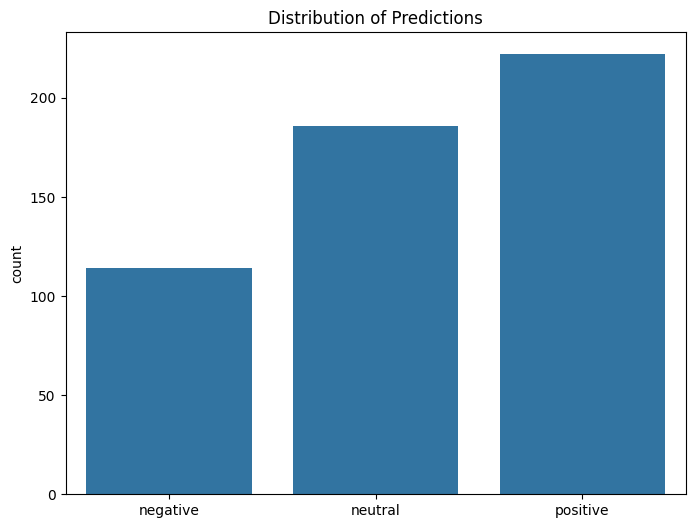
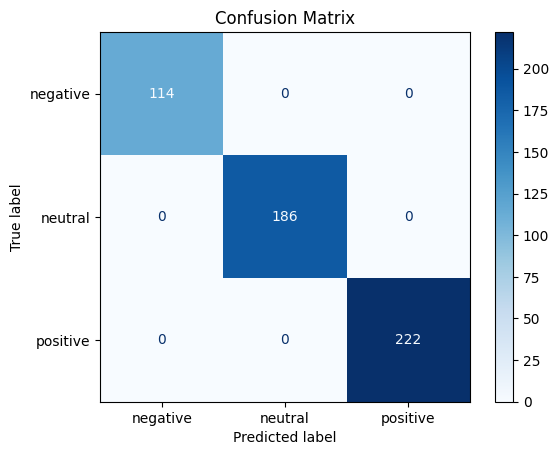
# Plot confusion matrix

plt.figure(figsize=(10, 8))

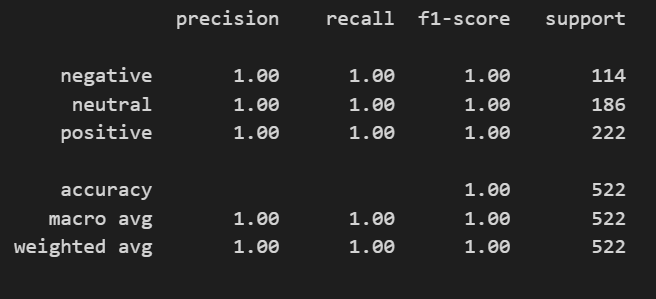
cmd.plot(cmap='Blues')

plt.title('Confusion Matrix')

plt.show()



На основу добијених резултата, примећујемо да је модел тачно класификовао све инстанце у тренинг сету. Овакав резултат врло вероватно указује на прекомерно прилагођавање модела (overfitting). Ово значи да модел може имати ограничену способност да добро ради на новим, невиђеним подацима. Потребно је додатно испитивање модела како би се утврдили могући проблеми са прекомерном прилагођавању и побољшала његова генерализација на тест сету.

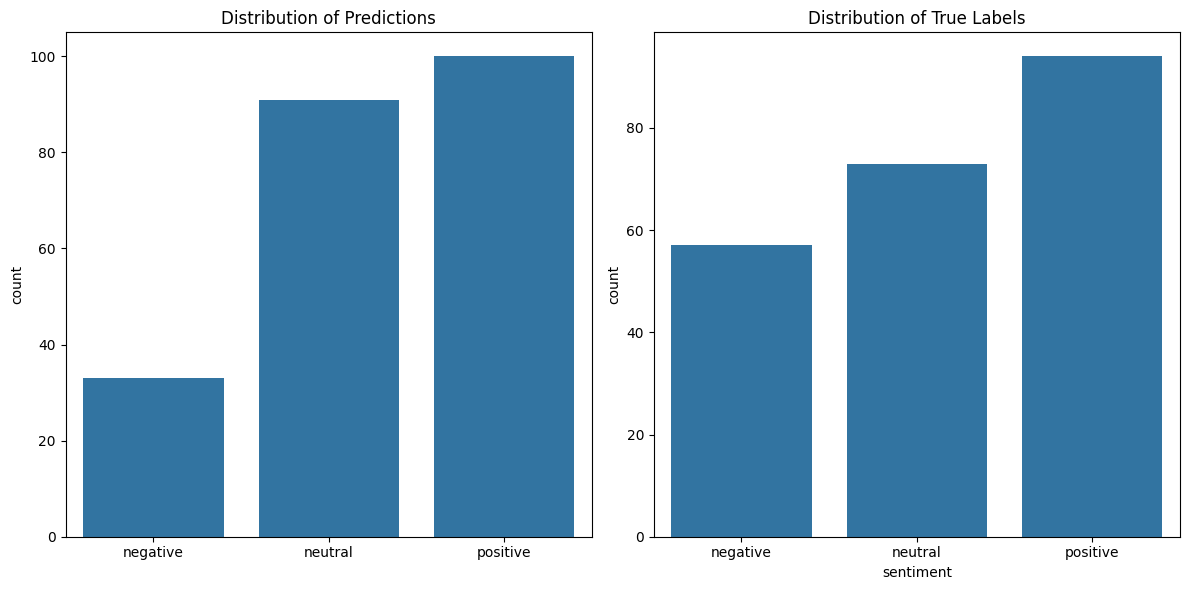


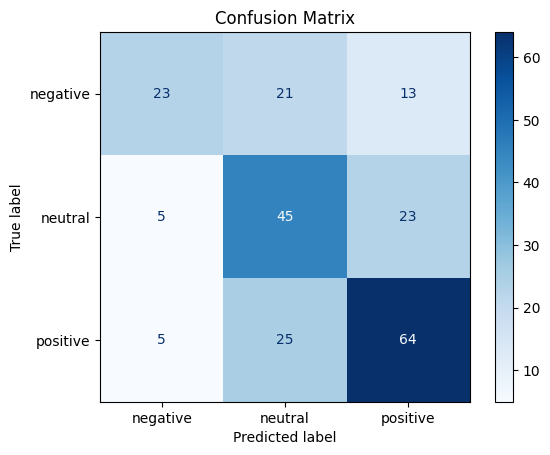
- Support: Ukupan broj stvarnih instanci za svaku klasu.

- Macro avg: Prosečna vrednost svih klasa, bez obzira na njihovu distribuciju.

- Weighted avg: Težinska prosečna vrednost svih klasa, pri čemu se veći značaj daje klasi sa većim brojem instanci. Ovo se koristi kada su klase nejednako distribuirane.

***Евалуација модела – Тест сет***

******



Confusion matrix for negative class:

TP: 23 FP: 10

FN: 34 TN: 157

Accuracy: 0.80

Precision: 0.70

Recall: 0.40

F1 Score: 0.51

Confusion matrix for neutral class:

TP: 45 FP: 46

FN: 28 TN: 105

Accuracy: 0.67

Precision: 0.49

Recall: 0.62

F1 Score: 0.55

Confusion matrix for positive class:

TP: 64 FP: 36

FN: 30 TN: 94

Accuracy: 0.71

Precision: 0.64

Recall: 0.68

F1 Score: 0.66

Ови резултати пружају увид у перформансе класификатора за сваку од сентимент класа. Ево неколико закључака које можемо извући:

**Негативна класа:** Класификатор има високу прецизност (0.70), што значи да када предвиђа негативну сентимент класу, тачно предвиђа већину негативних инстанци. Међутим, одзив (recall) је релативно низак (0.40), што значи да је много негативних инстанци остало неидентификовано. Ово указује на то да класификатор није довољно сензитиван за негативну класу. F1 скор (0.51) наглашава баланс између прецизности и одзива, али указује на потребу за побољшањем одзива.

**Неутрална класа:** Тачност је прилично солидна (0.67), што указује да је класификатор релативно добро идентификовао неутралне инстанце. Прецизност је нижа (0.49), што значи да је велики број инстанци које су класификоване као неутралне заправо нешто друго. Одзив је релативно добар (0.62), што указује на то да класификатор прилично добро идентификује стварне неутралне инстанце. F1 скор је такође солидан (0.55), али указује на могућност унапређења прецизности.

**Позитивна класа:** Класификатор има релативно високу прецизност (0.64), што значи да када предвиђа позитивну сентимент класу, тачно предвиђа већину позитивних инстанци. Одзив је такође прилично висок (0.68), што указује на то да класификатор добро идентификује стварне позитивне инстанце. F1 скор (0.66) сугерише добар баланс између прецизности и одзива, што је позитивно.

Уопштено, ови резултати указују на то да класификатор има најбоље перформансе у идентификацији позитивних инстанци, док постоји простор за побољшање у идентификацији негативних и неутралних инстанци.

from sklearn.metrics import balanced\_accuracy\_score

# Izračunavamo izbalansiranu tačnost

balanced\_acc = balanced\_accuracy\_score(y\_test, y\_pred)

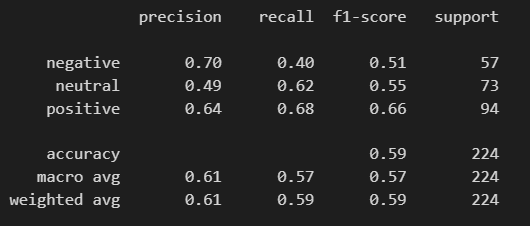
print(f"Balanced Accuracy: {balanced\_acc:.2f}")

**Balanced Accuracy vs. Accuracy**

**Balanced Accuracy**: Избалансирана тачност је мера перформанси класификатора која узима у обзир неравнотежу класа. Она се рачуна као просечна вредност тачности постигнуте на свакој класи. Избалансирана тачност може бити корисна када се суочавамо са несразмерним класама, јер даје подједнак значај свакој класи.

**Accuracy**: Тачност је мера перформанси класификатора која се рачуна као однос броја тачно предвиђених инстанци према укупном броју инстанци. Тачност може бити варљива у случају несразмерних класа, јер може приказати високе вредности чак и ако је перформанса на мањој класи лоша.

Избалансирана тачност (0.57) је нижа од класичне тачности (0.59). Ово указује на то да, иако класификатор има солидне перформансе према обичној тачности, његова способност да подједнако добро класификује све класе није оптимална. Нижа избалансирана тачност сугерише да класификатор не функционише једнако добро за све класе, што може бити посебно важно у контексту неравнотеже класа.



Макро просек представља просечну вредност прецизности, одзива и Ф1 скора за све класе, без обзира на дистрибуцију класа. На пример, макро прецизност се израчунава као просечна прецизност свих класа, макро одзив као просечан одзив свих класа, док је макро Ф1 скор просечан Ф1 скор свих класа.   
С друге стране, пондерисани просек представља тежинску просечну вредност прецизности, одзива и Ф1 скора, при чему се већи значај даје класама са већим бројем инстанци. Пондерисана средина је средња вредност добијена тако што се свака вредност множи одговарајућим пондером (у овом случају, бројем инстанци у свакој класи), а затим се збир тих производа дели са укупним бројем инстанци. Пондерисана прецизност се израчунава тако што се прецизност рачуна за сваку класу, а затим се рачуна пондерисана средина у односу на број инстанци у свакој класи. Исто тако, пондерисани одзив се рачуна тако што се одзив рачуна за сваку класу, а затим се рачуна пондерисана средина у односу на број инстанци у свакој класи. На крају, пондерисани Ф1 скор се рачуна тако што се Ф1 скор рачуна за сваку класу, а затим се рачуна пондерисана средина у односу на број инстанци у свакој класи.

Закључци из извештаја показују да макро просек за прецизност (0.61), одзив (0.57) и Ф1 скор (0.57) представља просечне перформансе модела без обзира на дистрибуцију класа. Ови резултати указују на то да модел има сличне перформансе за све класе, али не узима у обзир колико су класе неједнако дистрибуиране. С друге стране, пондерисани просек за прецизност (0.61), одзив (0.59) и Ф1 скор (0.59) узимају у обзир дистрибуцију класа. Ови резултати пружају реалнију слику перформанси модела, с обзиром на то да класе нису једнако заступљене. Пондерисани просек указује на то да модел боље перформира на учесталијим класама, али може имати проблема са ређе заступљеним класама.

Укупни закључци показују да модел има боље перформансе у идентификацији учесталијих класа (позитивних инстанци), док има ниже перформансе за ређе класе (негативне инстанце). Потребно је даље истражити и побољшати перформансе модела, посебно за идентификацију негативних и неутралних инстанци.