SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij

Klasifikacija Youtube komentara

Seminarski rad

Mihovil Šimić

Osijek, 2022.

SADRŽAJ

1. UVOD

Youtube je jedna od najpopularnijih i najkorištenijih platformi danas. Zbog velikog broja korisnika, postoji mnogo kanala kojima je Youtube posao i glavni izvor zarade. Iz navedenog razloga kanali moraju neprekidno smišljati nove ideje kako bi zadržali postojeću publiku i po mogućnosti ju proširili. Jedno od najvažnijih mjerila kvalitete i prihvaćenosti sadržaja od strane publike su komentari. Putem komentara kanali mogu procijeniti koliko dobro je njihov novi video prihvaćen te što točno publika misli o njemu. Iz navedenog razloga seminarski rad orijentiran je klasifikaciju komentara na dobre, loše i neutralne. Iako se komentari mogu klasificirati u još nekoliko grupa, navedene tri grupe su osnovne te predstavljaju jednostavan i efikasan uvid u mišljenje publike o videu.

U prvom dijelu rada objašnjeno je područje i problematika područja gdje su temeljno objašnjeni pojmovi strojnog učenja, klasifikacije i klasifikacijskih algoritama. U drugom dijelu rada detaljno je opisano programsko rješenje, te su uspoređeni, vizualizirani i objašnjeni dobiveni rezultati.

1.1. Zadatak seminarskog rada

Zadatak seminarskog rada je implementirati i objasniti klasifikaciju Youtube komentara na dobre, loše i neutralne koristeći programski jezik Python. Za klasifikaciju je potrebno koristiti više različitih algoritama te ih usporediti i objasniti koji je najoptimalniji za zadani zadatak. Rezultate je potrebno vizualizirati putem grafova i matrica.

2. PREGLED PODRUČJA I PROBLEMATIKE

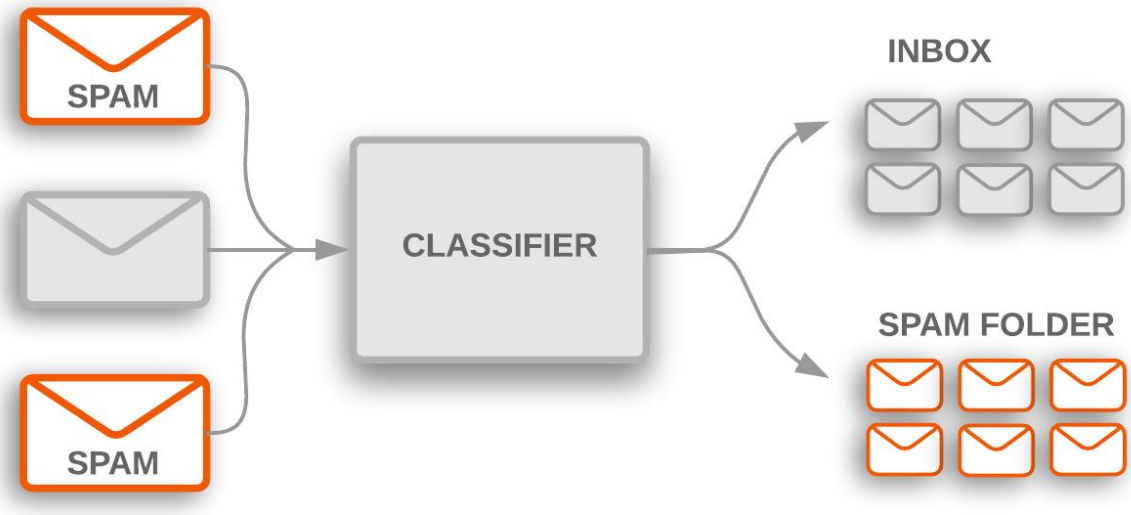
2.1. Strojno učenje

Strojno učenje je grana umjetne inteligencije i računalne znanosti koja se usredotočuje na korištenje podataka i algoritama za oponašanje načina na koji ljudi uče, postupno poboljšavajući njegovu točnost. Strojno učenje važna je komponenta rastućeg područja podatkovnih znanosti.

Strojno učenje može se podijeliti na 3 glavna dijela:

- Proces odlučivanja: općenito, algoritmi strojnog učenja koriste se za predviđanje ili klasifikaciju
- Funkcija pogreške: funkcija pogreške služi za procjenu predviđanja modela. Ako postoje poznati primjeri, funkcija pogreške može napraviti usporedbu kako bi procijenila točnost modela
- Proces optimizacije modela: ako model može bolje odgovarati točkama podataka u skupu za obuku, tada se parametri prilagođavaju kako bi se smanjila razlika između poznatog primjera i procjene modela. Algoritam će ponoviti ovaj proces evaluacije i optimizacije, samostalno ažurirajući težine dok se ne dosegne prag točnosti

Podjela strojnog učenja moguća je na dvije vrste, a to su nadzirano i nenadzirano. Nadzirano učenje, definirano je korištenjem označenih skupova podataka za obuku algoritama koji klasificiraju podatke ili točno predviđaju ishode. Kako se ulazni podaci unose u model, on prilagođava svoje težine dok se model ne uklopi na odgovarajući način. To se događa kao dio procesa unakrsne provjere kako bi se osiguralo da model izbjegne prekomjerno ili nedovoljno uklapanje. Nadzirano učenje pomaže organizacijama u rješavanju raznih problema iz stvarnog svijeta u velikim razmjerima, kao što je razvrstavanje neželjene pošte. Neke metode koje se koriste u nadziranom učenju uključuju neuronske mreže, linearnu regresiju, logističku regresiju, stroj potpornih vektora (SVM) i još mnogo toga [1].



Slika 1. Vizualizacija razvrstavanja neželjene pošte [2]

Nenadzirano učenje, koristi algoritme strojnog učenja za analizu i grupiranje neoznačenih skupova podataka. Ovi algoritmi otkrivaju skrivene obrasce bez potrebe za ljudskom intervencijom. Karakteristika otkrivanja sličnosti i razlika u informacijama čini nenadzirano učenje idealnim rješenjem za istraživačku analizu podataka, strategije unakrsne prodaje, segmentaciju kupaca te prepoznavanje slika i uzoraka. Algoritmi koji se koriste u nenadziranom učenju uključuju neuronske mreže, grupiranje k-srednjih vrijednosti, metode vjerojatnog grupiranja i još mnogo toga [1].

2.2. Klasifikacija

U strojnom učenju, klasifikacija se odnosi na problem prediktivnog modeliranja gdje se predviđa oznaka klase za dani primjer ulaznih podataka. Primjeri klasifikacijskih problema uključuju:

- S obzirom na primjer, klasificirati je li pošta neželjena ili ne
- S obzirom na rukopisni znak, klasificirati ga kao jedan od poznatih znakova
- S obzirom na nedavno ponašanje korisnika, klasificirati kao mogućeg kupca ili ne

Iz perspektive modeliranja, klasifikacija zahtijeva skup podataka za obuku s mnogo primjera ulaza i izlaza iz kojih se uči. Model će koristiti skup podataka za obuku i izračunat će kako najbolje mapirati primjere ulaznih podataka u određene oznake klase. Kao takav, skup podataka za obuku mora biti dovoljno reprezentativan za problem i imati mnogo primjera svake oznake klase. Oznake klasa često su vrijednosti niza, npr. “neželjena pošta”, “ne neželjena pošta” i moraju se preslikati na numeričke vrijednosti prije nego što se daju algoritmu za modeliranje. Ovo se često naziva

kodiranjem oznake, gdje se svakoj oznaci klase dodjeljuje jedinstveni cijeli broj, npr. "neželjena pošta" = 0, "bez neželjene pošte" = 1. Algoritmi za prediktivno modeliranje klasifikacije ocjenjuju se na temelju njihovih rezultata. Točnost klasifikacije popularna je metrika koja se koristi za procjenu izvedbe modela na temelju predviđenih oznaka klasa. Točnost klasifikacije nije savršena, ali je dobra polazna točka za mnoge zadatke klasifikacije. Umjesto oznaka klasa, neki zadaci mogu zahtijevati predviđanje vjerojatnosti članstva u klasi za svaki primjer. To daje dodatnu nesigurnost u predviđanju koje aplikacija ili korisnik tada mogu protumačiti [3]. Postoje četiri glavne vrste klasifikacijskih zadataka:

- Binarna klasifikacija
- Klasifikacija s više klasa
- Klasifikacija s više oznaka
- Neuravnotežena klasifikacija

2.3. Klasifikacijski algoritmi

2.3.1. Logistička regresija (engl. Logistic regression)

Logistička regresija je proces modeliranja vjerojatnosti diskretnog ishoda s obzirom na ulaznu varijablu. Najčešće logistička regresija ima binarni ishod, što znači da može imati dvije vrijednosti kao što su točno/netočno, da/ne i tako dalje. Multinomska logistička regresija koristi se u situacijama u kojima postoji više od dva moguća diskretna ishoda koji ne moraju biti poredani i imati redoslijed. Redna logistička regresija se koristi kada postoji tri ili više klasa koje trebaju biti poredani i imati jasan redoslijed. Logistička regresija korisna je metoda analize za probleme klasifikacije, gdje se pokušava utvrditi u koju se kategoriju novi uzorak najbolje uklapa. Kako su aspekti kibernetičke sigurnosti problemi klasifikacije, kao što je otkrivanje napada, logistička je regresija korisna analitička tehnika. [4]

Multinomska logistička regresija prikladna je za svaku situaciju u kojoj se modelira ograničen broj kategorija ishoda (više od dvije) i gdje te kategorije ishoda nemaju redoslijed. Temeljna pretpostavka je neovisnost irelevantnih alternativa (IIA). Inače rečeno, ova pretpostavka znači da ne postoji druga alternativa za ishod koja bi, ako se uključi, nerazmjerno utjecala na članstvo u jednoj od drugih kategorija. Primjeri tipičnih situacija koje se mogu modelirati multinomskom logističkom regresijom uključuju:

- Modeliranje biračkog izbora na izborima s više kandidata

- Modeliranje izbora karijera od strane studenata
- Modeliranje izbora opcija pogodnosti od strane zaposlenika

Multinomski logistički model temeljit će se na definiranoj referentnoj kategoriji i izvoditi generalizirani linearni model na log-izgledima članstva svake druge kategorije u odnosu na referentnu kategoriju. Zbog svoje široke upotrebe u epidemiologiji i medicini, to se često naziva relativnim rizikom jedne kategorije u usporedbi s referentnom kategorijom. Matematički govoreći, ako je X vektor ulaznih varijabli, a y uzima vrijednost A, B ili C, s A kao referencom, multinomski logistički regresijski model će izračunati:

$$\ln\left(\frac{P(y=B)}{P(y=A)}\right) = \alpha X$$

Slika 2. Formula multinomske logističke regresije

i

$$\ln\left(\frac{P(y=C)}{P(y=A)}\right) = \beta X$$

Slika 3. Formula multinomske logističke regresije

za različite vektore koeficijenata α i β [4].

2.3.2. Naive Bayes

Naive Byes metode skup su algoritama nadziranog učenja koji se temelje na primjeni Bayesova teorema s "naivnom" pretpostavkom uvjetne neovisnosti između svakog para značajki s obzirom na vrijednost varijable klase. Bayesov teorem navodi sljedeću vezu, s obzirom na varijablu klase y i vektor ovisnih značajki x_1 kroz x_n :

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)P(x_1, \ldots, x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Koristeći naivnu pretpostavku uvjetne neovisnosti da

$$P(x_i|y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y),$$

za sve i, ovaj odnos je pojednostavljen na

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} \quad P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Budući da je $P(x_1, \ldots, x_n)$ konstanta s obzirom na ulaz, može se koristiti sljedeće pravilo klasifikacije:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} \quad P(x_i \mid y)$$

$$\Downarrow$$

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} \quad P(x_i \mid y)$$

Različiti naivni Bayesovi klasifikatori razlikuju se uglavnom po pretpostavkama koje donose u pogledu distribucije . Unatoč svojim naizgled previše pojednostavljenim pretpostavkama, naivni Bayesovi klasifikatori su prilično dobro funkcionirali u mnogim stvarnim situacijama, poput klasificiranja dokumenata i filtriranje neželjene pošte. Za procjenu potrebnih parametara potrebna im je mala količina podataka za treniranje. Naivni Bayesovi klasifikatori mogu biti iznimno brzi u usporedbi sa sofisticiranijim metodama. Odvajanje distribucija uvjetnih obilježja klase znači da se svaka distribucija može neovisno procijeniti kao jednodimenzionalna distribucija.

MultinomialNB implementira naivni Bayesov algoritam za multinomsko distribuirane podatke i jedna je od dvije klasične naivne Bayesove varijante koje se koriste u klasifikaciji teksta (gdje su podaci obično predstavljeni kao broj vektora riječi). Distribucija je parametrizirana vektorima $\theta_y = (\theta_{y1}, \ldots, \theta_{yn})$ za svaki razred , gdje je broj obilježja (u klasifikaciji teksta, veličina rječnika) i je vjerojatnost značajke koje se pojavljuju u uzorku koji pripada klasi.

ComplementNB implementira komplementarni naivni Bayesov algoritam (CNB). CNB je prilagodba standardnog multinomskog naivnog Bayesovog algoritma (MNB) koji je posebno prikladan za neuravnotežene skupove podataka. Izumitelji CNB-a empirijski pokazuju da su procjene parametara za CNB stabilnije od onih za MNB. Nadalje, CNB redovito nadmašuje MNB (često sa značajnom razlikom) u zadacima klasifikacije teksta. Postupak za izračun težine je sljedeći:

$$\hat{\theta}_{ci} = \frac{\alpha_i + \sum_{j:y_j \neq c} \quad d_{ij}}{\alpha + \sum_{j:y_j \neq c} \quad \sum_k \quad d_{kj}}$$

$$w_{ci} = \log \hat{\theta}_{ci}$$

$$w_{ci} = \frac{w_{ci}}{\sum_j |\, w_{cj}|}$$

BernoulliNB implementira naivne Bayesove algoritme za obuku i klasifikaciju za podatke koji se distribuiraju prema multi varijantnim Bernoullijevim distribucijama; tj. može postojati više značajki, ali se pretpostavlja da je svaka varijabla binarne vrijednosti (Bernoulli, boolean). Stoga ova klasa zahtijeva da uzorci budu predstavljeni kao vektori značajki binarnih vrijednosti; ako se preda bilo koja druga vrsta podataka, BernoulliNB instanca može binarizirati svoj ulaz (ovisno o parametru binarizacije).

Pravilo odluke za Bernoullijevog naivnog Bayesa temelji se na

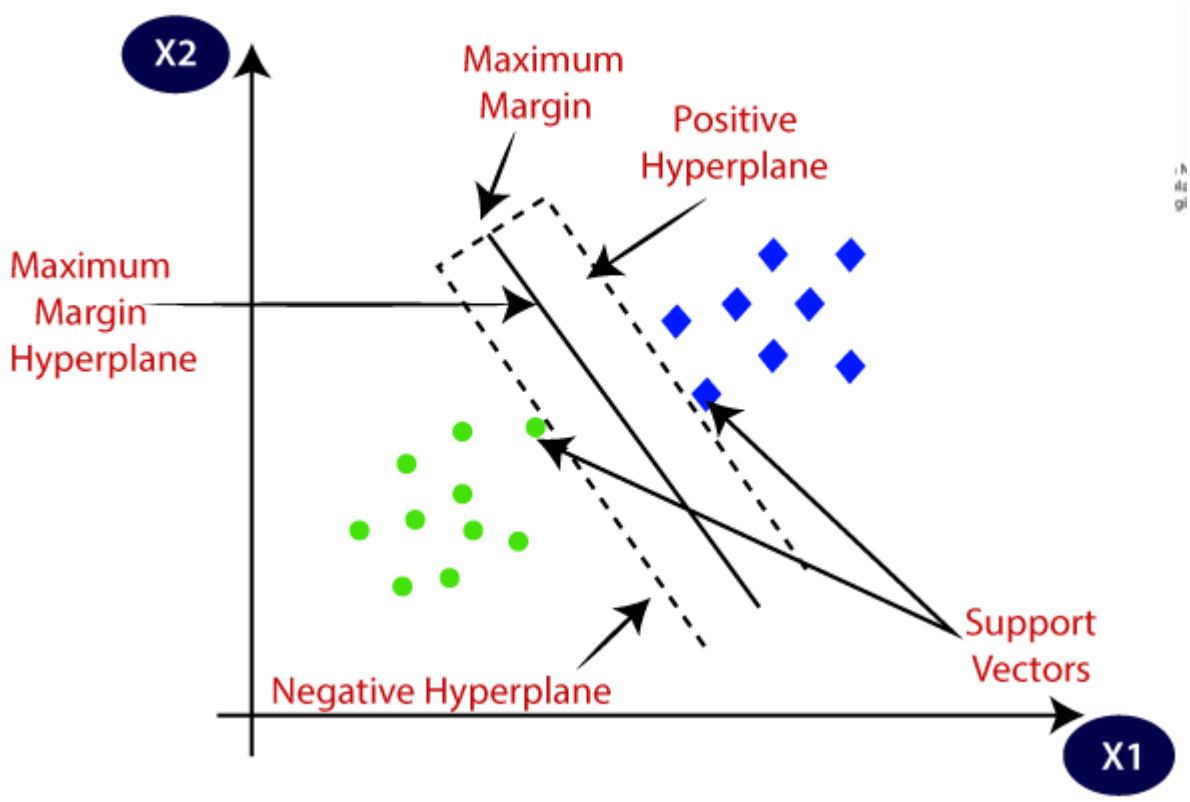
$$P(x_i \mid y) = P(i \mid y)x_i + (1 - P(i \mid y))(1 - x_i)$$

koje se razlikuje od multinomskog NB-ovog pravila po tome što izričito kažnjava nepojavljivanje značajke koje je pokazatelj za klasu , gdje bi multinomska varijanta jednostavno zanemarila značajku koja se ne pojavljuje. U slučaju klasifikacije teksta, vektori pojavljivanja riječi mogu se koristiti za obuku i korištenje ovog klasifikatora. BernoulliNB bi mogao imati bolji učinak na nekim skupovima podataka, osobito onima s kraćim dokumentima.[5]

2.3.3. Vektor linearne potpore (engl. Linear support vector)

Za početak je potrebno objasniti stroj potpornih vektora. Stroj potpornih vektora ili SVM jedan je od najpopularnijih algoritama za nadzirano učenje, koji se koristi za probleme klasifikacije i regresije. Cilj SVM algoritma je stvoriti najbolju liniju ili granicu odluke koja može odvojiti n-dimenzionalni prostor u klase tako da možemo lako staviti novu podatkovnu točku u ispravnu kategoriju u budućnosti. Ova granica najbolje odluke naziva se hiperravnina.

SVM bira ekstremne točke/vektore koji pomažu u stvaranju hiperravnine. Ovi ekstremni slučajevi se nazivaju vektori podrške, pa se algoritam može i nazvati strojem vektora podrške. Donji dijagram prikazuje dvije različite kategorije koje su klasificirane korištenjem granice odluke ili hiperravnine: [6]

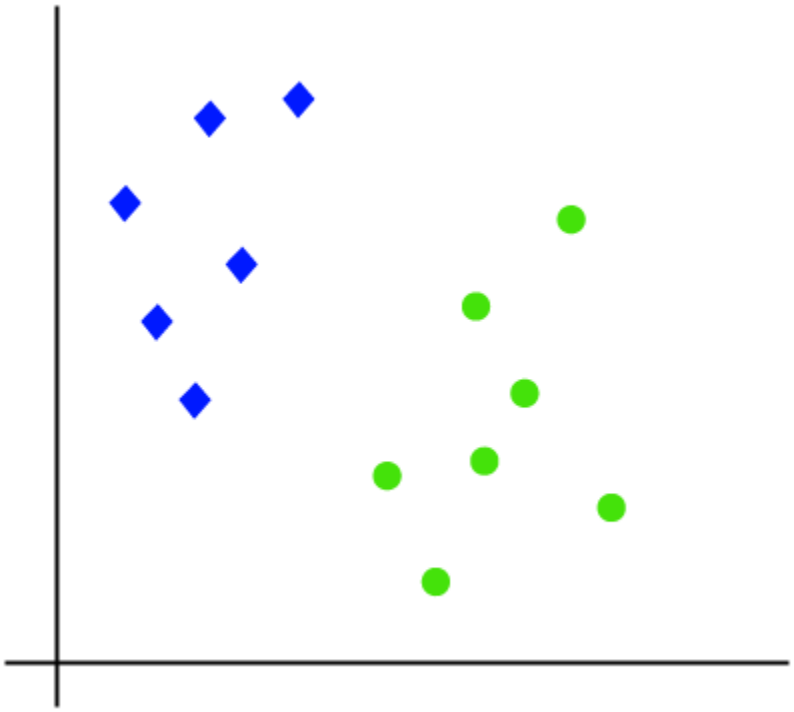


Slika 4. Vizualizacija stroja potpornih vektora [7]

SVM ima dvije vrste:

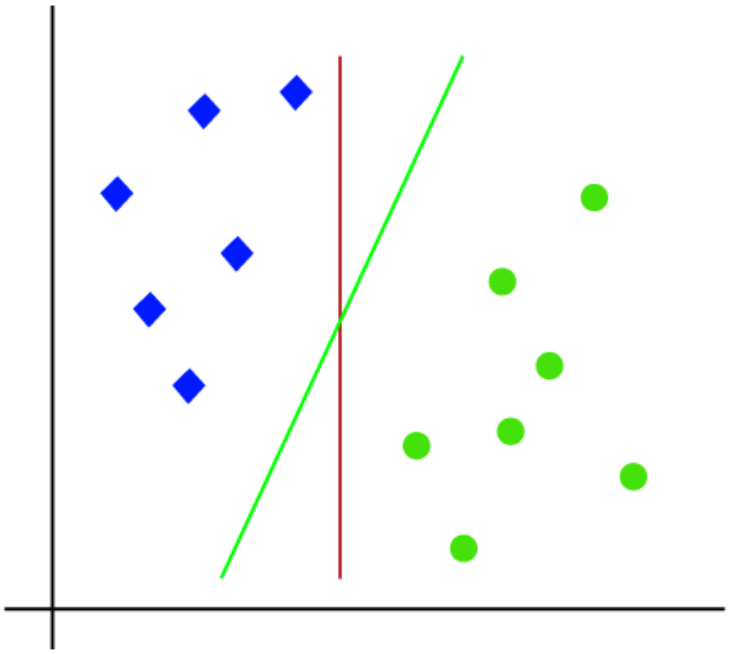
- Linearni SVM: Linearni SVM koristi se za linearno odvojive podatke, što znači da ako se skup podataka može klasificirati u dvije klase pomoću jedne ravne crte, tada se takvi podaci nazivaju linearno odvojivi podaci.
- Nelinearni SVM: Nelinearni SVM se koristi za nelinearno odvojene podatke, što znači da ako se skup podataka ne može klasificirati pomoću pravocrtne crte, tada se takvi podaci nazivaju nelinearnim podacima.

Rad linearnog SVM algoritma može se opisati korištenjem primjera. Pretpostavka je da skup podataka koji ima dvije oznake (zelenu i plavu) i dvije značajke x1 i x2. Cilj je napraviti klasifikator koji može klasificirati par (x1, x2) koordinata u zelenu ili plavu. Sljedeća slika prikazuje skupa podataka:



Slika 5. Graf skupa podataka [8]

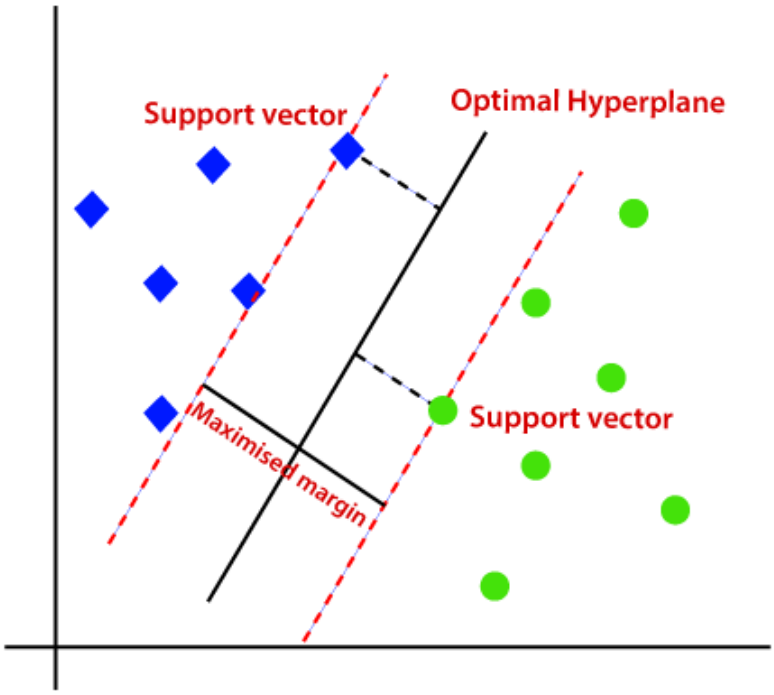
Budući kako je to 2D prostor, moguće je korištenjem ravne linije lako odvojiti ove dvije klase. Ali može postojati više linija koje mogu odvojiti ove klase. Sljedeća slika prikazuje prethodno objašnjeno:



Slika 6. Odvajanje klasa pomoću dvije linije [9]

Dakle, linearni SVM algoritam pomaže pronaći najbolju liniju ili granicu odluke; ova najbolja granica ili regija naziva se hiperravninom. Linearni SVM algoritam pronalazi najbližu točku do

linije iz obje klase. Te se točke nazivaju vektori potpore. Udaljenost između vektora i hiperravnine naziva se margina. A cilj linearnog SVM-a je maksimizirati ovu marginu. Hiperravnina s maksimalnom marginom naziva se optimalnom hiperravninom.

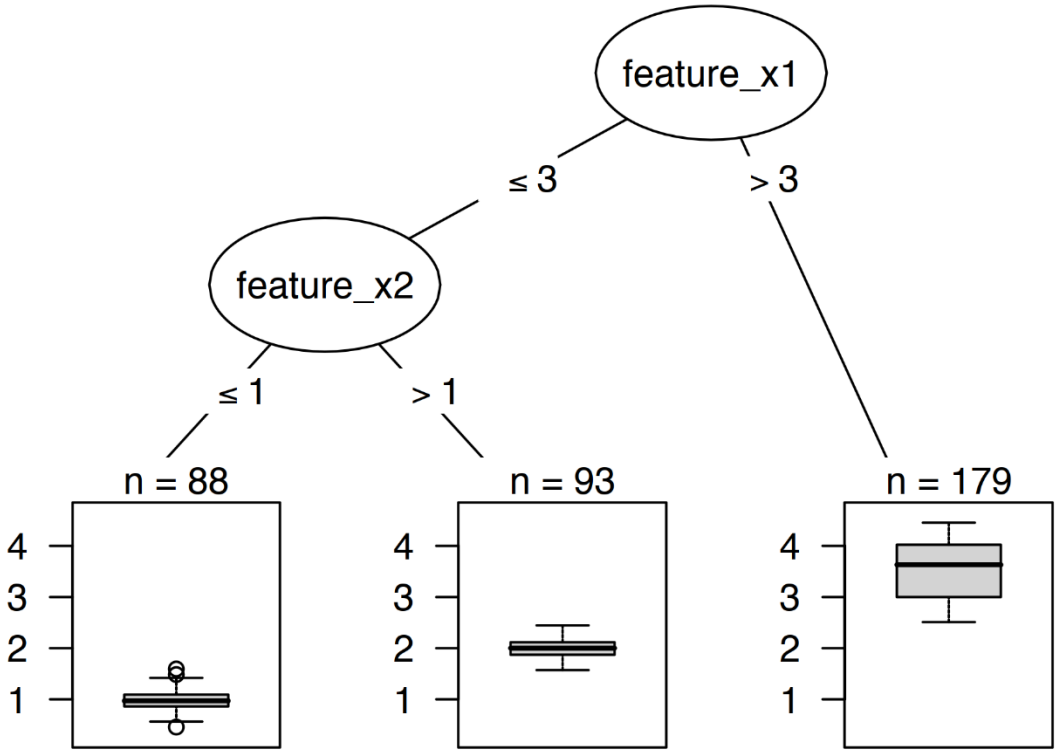


Slika 7. Prikaz hiperravnine i vektora potpore [10]

U slučaju više klasa algoritam će odvojiti rješavanja u više manjih problema.

2.3.4. Klasifikator stabla odluka (engl. Decision tree classifier)

Modeli temeljeni na stablu dijele podatke više puta prema određenim graničnim vrijednostima u značajkama. Podjelom se stvaraju različiti pod skupovi skupa podataka, pri čemu svaka instanca pripada jednom podskupu. Konačni pod skupovi nazivaju se terminalni ili lisni čvorovi, a srednji pod skupovi se nazivaju unutarnji čvorovi ili podijeljeni čvorovi. Za predviđanje ishoda u svakom lisnom čvoru koristi se prosječni ishod podataka obuke u ovom čvoru. Stabla se mogu koristiti za klasifikaciju i regresiju.



Slika 8. Klasifikator stabla odluka [11]

Stablo odluka s umjetnim podacima. Instance s vrijednošću većom od 3 za značajku x1 završavaju u čvoru 5. Sve ostale instance se dodjeljuju čvoru 3 ili čvoru 4, ovisno o tome da li vrijednosti značajke x2 prelaze 1 [11].

3. OPIS PROGRAMSKOG RJEŠENJA

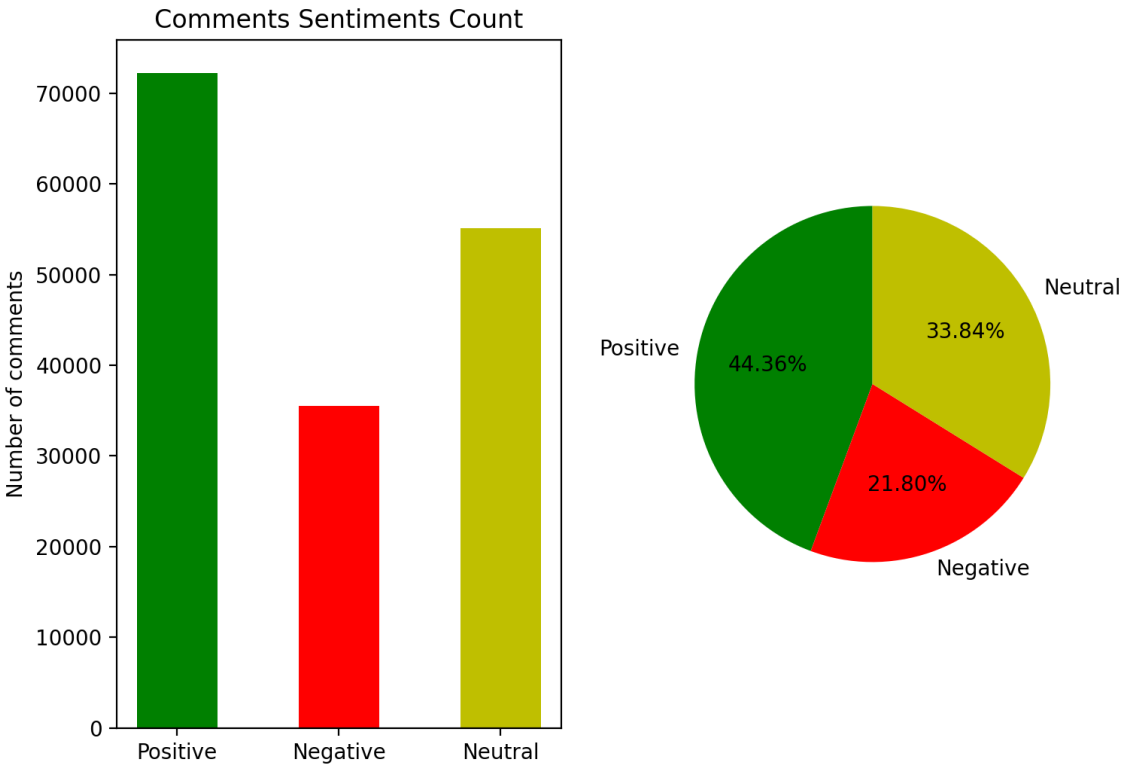
3.1. Skup podataka

Zbog nemogućnosti pronalaska skupa podataka koji sadrži klasifikaciju Youtube komentara, korišten je skup podataka za klasifikaciju Twitter komentara [12]. Skup podataka je zadovoljavajući za korištenje jer je pozitivan komentar na Twitter-u jednak kao i pozitivan komentar na Youtube-u, i tako dalje. Komentari su podijeljeni na pozitivne, negativne i neutralne, na način da klasa 1 označuje pozitivan, klasa -1 negativan te klasa 0 neutralan komentar. Skup podataka sadrži 162853 komentara, gdje je 72240 pozitivnih, 35508 negativnih te 55105 neutralnih komentara.

Komentar	Klasa

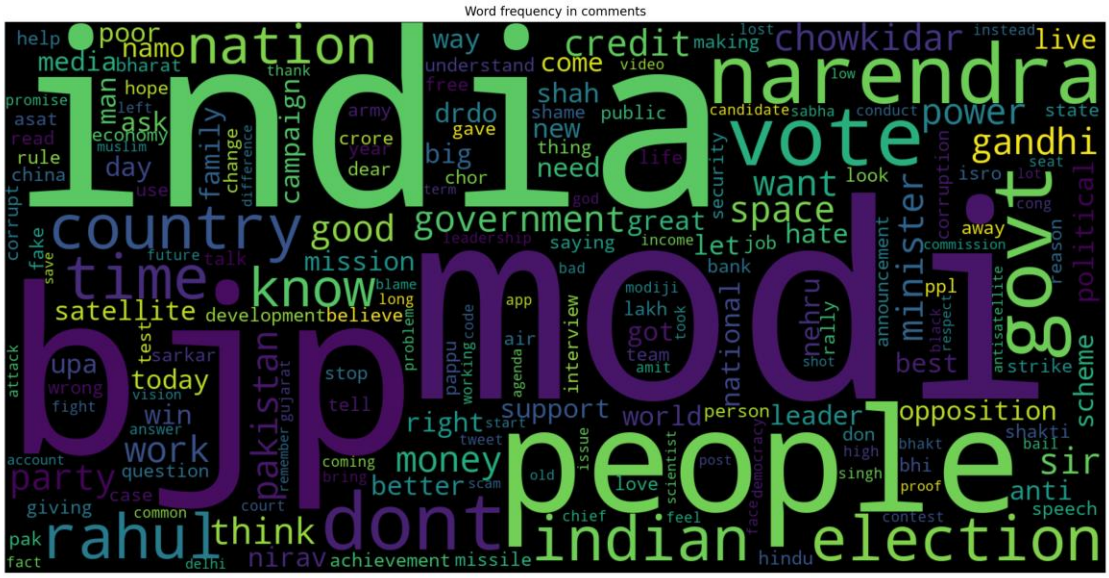
when modi promised “minimum government maximum governance” expected him begin the difficult job reforming the state why does take years get justice state should and not business and should exit psus and temples	-1
talk all the nonsense and continue all the drama will vote for modi	0
answer who among these the most powerful world leader today trump putin modi may	1
development has become mass movement under modi govt with economic social and political empowerment life one and all has witnessed positive paradigm shift this new india"	1
was waiting for this modi will also talk about varanasi	0
just like fake gandhi’ fake modi’ will there	-1

Slika 9. Komentari i odgovarajuće klase



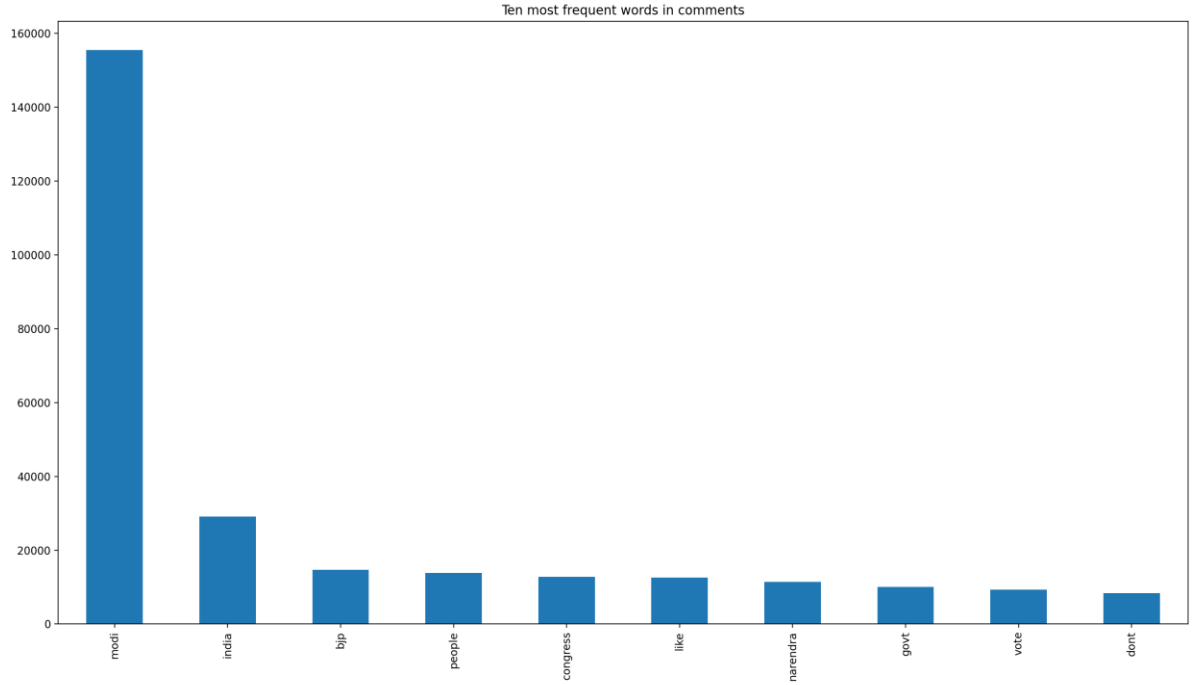
Slika 10. Stupčasti i kružni grafikon za skup podataka

Na temelju grafova može se zaključiti da je broj pozitivnih komentara najveći, nakon njega je broj neutralnih te je broj negativnih najmanji. Prethodna rečenica može se izraziti i u postotcima, što znači da ima 44.36% pozitivnih, 33.84% neutralnih te 21.08% negativnih komentara.



Slika 11. Oblak riječi skupa podataka

Slika 3. predstavlja oblak riječi koji se može napraviti koristeći Python biblioteku wordcloud. Navedena biblioteka je veoma jednostavna i korisna za korištenje. Na slici je veličina riječi ekvivalenta frekvenciji ponavljanja.



Slika 12. Stupčasti grafikon frekvencije riječi u skupu podataka

Grafikon potvrđuje točnost oblaka riječi te prikazuje identičnu stvar na jednostavniji i lakši način.

3.2. Priprema i vizualizacija podataka

Prije korištenja podataka potrebno ih je obraditi i pripremiti kako bi ih algoritmi mogli pravilno koristiti.

```
#spacy is library for natural language processing and 'en_core_web_sm' file contains stopwords which
#are not needed for model training. Some examples are often, the, therefore, another, if, whereafter, your, thereby, does, etc...
english = spacy.load('en_core_web_sm')
stopWords = english.Defaults.stop_words

def removePunctuationFromString(text):
    return text.translate(str.maketrans('', '', string.punctuation))

def removeWordsFromString(text):
    return ' '.join([word for word in text.split() if word.lower() not in stopWords])

def cleanFile(file):
    for index in range(len(file)):
        #remove punctuation from string
        file.at[index, 'comment'] = removePunctuationFromString(str(file.at[index, 'comment']))
        #remove stop words from string
        file.at[index, 'comment'] = removeWordsFromString(str(file.at[index, 'comment']))
        #remove everything other than number or letter, in this case remove emojis
        file.at[index, 'comment'] = re.sub('[^A-Za-z0-9]+', ' ', file.at[index, 'comment'])
    #remove empty lines, first mark them as NaN, then drop them from dataframe
    file['comment'].replace(' ', np.nan, inplace=True)
    file.dropna(subset = ["comment"], inplace=True)
    file['comment'].replace('', np.nan, inplace=True)
    file.dropna(subset = ["comment"], inplace=True)
    return file

def makeValidCSV(fileName):
    file = pd.read_csv(path.join('files\\raw\\', fileName + '.csv'), names=['comment', 'polarity'])
    validFile = cleanFile(file)
    #index is False in order to remove adding line numbers, and float format is set to '%.0f', because by deafult numbers are in a float format
    validFile.to_csv(path.join('files\\cleaned\\', fileName + '.csv'), index=False, float_format='%.0f')
```

Slika 13. Kod za pripremu i čišćenje podataka

Prvo je potrebno ukloniti stop riječi, točnije riječi koje ne pomažu pri određivanju klase komentara, poput veznika, zamjenica i tako dalje. Nakon toga se uklanjaju svi znakovi interpunkcije iz komentara iz jednakog razloga. Pošto se u komentarima mogu nalaziti i emoji znakovi, isti se trebaju ukloniti, pomoću regexa koji uklanja sve što nije slovo ili broj. Na kraju se uklanjaju prazne linije, na način da se označe kao NaN (engl. not a number) i odbace iz skupa podataka. Nakon čišćenja podataka, sve promjene se spremaju u novu datoteku.

```
def loadTrainingData():
    return pd.read_csv('files\\cleaned\\cleanedDataset.csv')

def loadInputData(fileName):
    return pd.read_csv(path.join('files\\cleaned\\', fileName + '.csv'))

#Convert a collection of text documents to a matrix of token counts
#ngram_range of (1, 1) means only unigrams will be selected. Unigram is one word sequence
def transformComments(data):
    cv = CountVectorizer(ngram_range = (1,1))
    return cv.fit_transform(data['comment'].values.astype('U'))

#convert data type of column in dataframe to Variable-length Unicode
def transformPolarity(data):
    return data['polarity'].values.astype('U')
```

Slika 14. Kod za učitavanje podataka i pretvorbu vrijednosti

Za početak je potrebno učitati podatke funkcijom *read_csv*. Nakon toga se obrađuju komentari i klase komentara, tako što im se vrijednost postavlja na Unicode. U slučaju komentara prvo je potrebno funkcijom *CountVectorizer* te parametrom *ngram_range* odrediti odabir jedne riječi. Pod vizualizacijom podataka podrazumijeva se grafička prezentacija informacija i podataka. Služenjem vizualnih elemenata kao što su grafovi, dijagrami i karte, vizualizacija podataka predstavlja jednostavnu i lako dostupnu metodu prikaza trendova, odstupanja i uzoraka u podacima te olakšava razumijevanje istih [13].

```
def getWordFrequencyInComments(data):
    return pd.Series(' '.join(data['comment']).split()).value_counts()

def getTenMostFrequentWordsInComments(data):
    return getWordFrequencyInComments(data)[:10]

def getCommentsCount(data):
    totalNumberOfComments = len(data.index)
    positiveCommentsCount = (data['polarity'] == 1).sum()
    negativeCommentsCount = (data['polarity'] == -1).sum()
    neutralCommentsCount = (data['polarity'] == 0).sum()

    return totalNumberOfComments, positiveCommentsCount, negativeCommentsCount, neutralCommentsCount

def getCommentsPercentage(data):
    totalNumberOfComments, positiveCommentsCount, negativeCommentsCount, neutralCommentsCount = getCommentsCount(data)
    positiveCommentPercentage = positiveCommentsCount / totalNumberOfComments
    negativeCommentPercentage = negativeCommentsCount / totalNumberOfComments
    neutralCommentPercentage = neutralCommentsCount / totalNumberOfComments

    return positiveCommentPercentage, negativeCommentPercentage, neutralCommentPercentage
```

Slika 15. Kod za vizualizaciju podataka

Prije vizualizacije potrebno je pripremiti podatke koji se vizualiziraju. Potrebno je prebrojati koliko ima komentara svake klase, te navedeno također izraziti i u postotcima. Najčešće riječi u komentarima služe kako bi se mogao napraviti oblak komentara prethodno prikazan u poglavlju 3.1.

3.3. Modeli strojnog učenja

Korišteni algoritmi za treniranje su vektor linearne potpore, logistička regresija, klasifikator stabla odluka, multinomial Naive Bayes, complement Naive Bayes te bernoulli Naive Bayes.

```
def trainModel(choice):
    modelOption = {
        1: LinearSVC(),
        2: MultinomialNB(),
        3: ComplementNB(),
        4: BernoulliNB(),
        5: LogisticRegression(),
        6: tree.DecisionTreeClassifier()
    }
    data = loadTrainingData()

    x = transformComments(data)
    y = transformPolarity(data)

    startTime = time.time()
    xTrain, xTest, yTrain, yTest = train_test_split(x, y, test_size=0.20)
    model = modelOption[choice]
    model.fit(xTrain, yTrain)
    yPredict = model.predict(xTest)
    trainingTime = time.time() - startTime
    return yTest, yPredict, trainingTime


def testModel(menuChoice):
    yTest, yPredict, trainingTime = trainModel(menuChoice)
    createModelMetrics(yTest, yPredict, trainingTime)
```

Slika 16. Kod za odabir, izradu i testiranje modela

Na početku se na temelju korisnikovog unosa bira model, nakon čeka se učitavaju i pripremaju podatci. Nakon što su podatci pripremljeni, potrebno je podijeliti podatke na trening i test grupu u omjeru 80/20. *xTrain* su komentari za trening, a *yTrain* klase za trening. *xTest* i *yTest* su komentari i klase komentara za testiranje točnosti. Nakon podjele podataka i izrade modela, model napravi pretpostavke na temelju testnih komentara. Funkcijom *time* se računa vrijeme potrebno za izradu modela i izračun pretpostavki klasa komentara. Nakon svih potrebnih izračuna, podatci se vizualiziraju na način da se prikaže matrica zabune, vrijeme treniranja, točnost, sveukupni broj

komentara, broj pozitivnih komentara, broj negativnih komentara, broj neutralnih komentara, te postotak svih navedenih.

```
def createModelMetrics(yTest, yPredict, trainingTime):
    confusionMatrix = confusion_matrix(yTest, yPredict)
    accuracyScore = accuracy_score(yTest, yPredict)
    disp = ConfusionMatrixDisplay(confusion_matrix=confusionMatrix, display_labels=[
                                  'negative', 'neutral', 'positive'])
    disp.plot()
    plt.show()
    print('Accuracy:', round(accuracyScore, 4) * 100, '%')
    print('Training time:', round(trainingTime, 4), 'seconds')
    getDataMetricsForAlgorithm(yPredict)
```

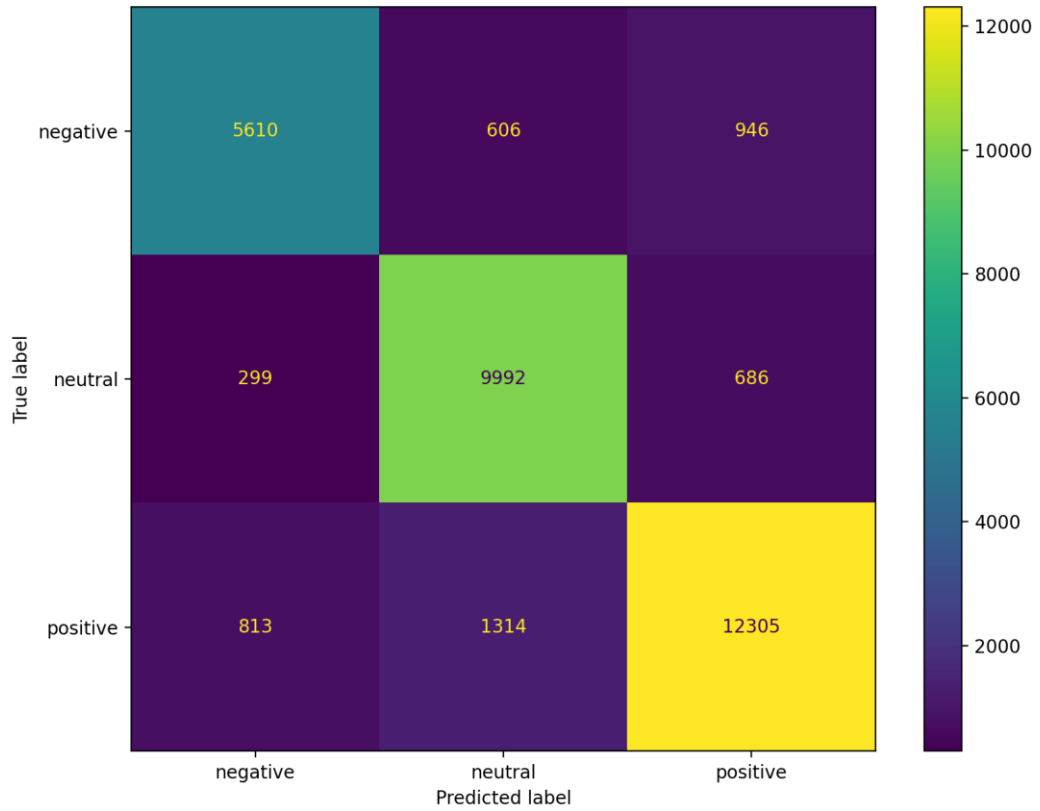
Slika 17. Kod za prikaz metrika rezultata algoritma

Sve izračunate vrijednosti se zaokružuju na 2 decimale. Broj 4 znači da će biti 4 broja sveukupno, točnije 2 prije decimalne točke i 2 poslije.

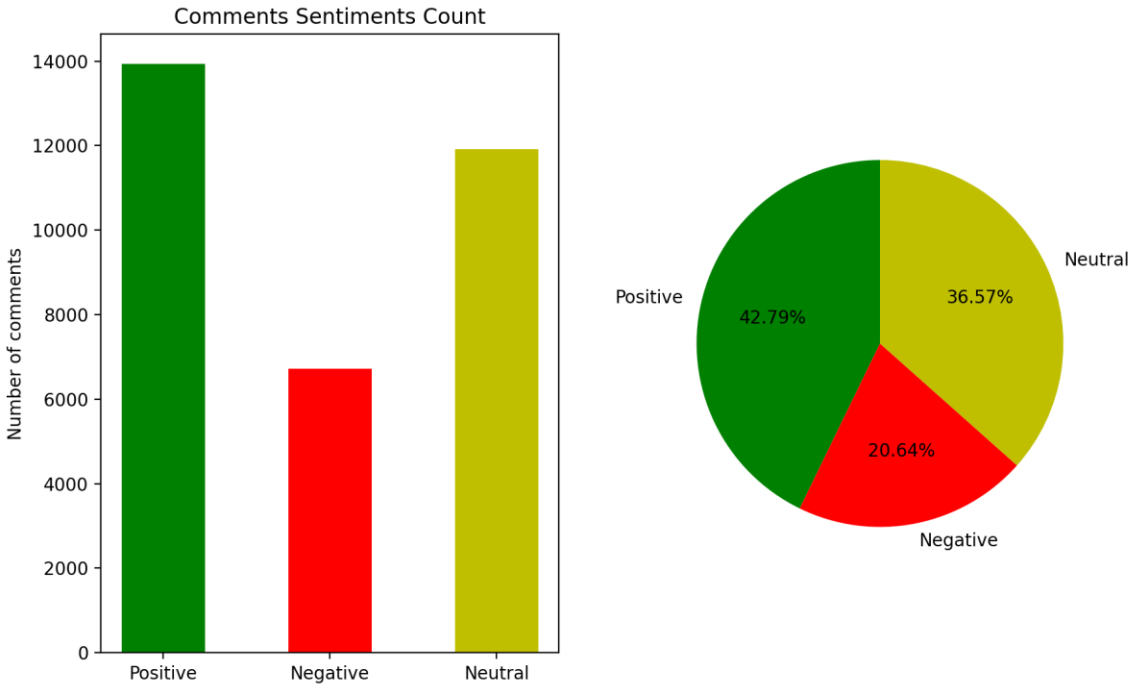
3.4. Rezultati i usporedba algoritama

Kako bi se što točnije opisali podatci za svaki model, prikazana je matrica zabune, preciznost te vrijeme treniranja. Uz navedene metrike rezultati su također i vizualizirani pomoću stupčastog grafa koji prikazuje broj pozitivnih, negativnih i neutralnih komentara te kružnog grafa koji prikazuje postotak istih. U nastavku su detaljno opisani postignuti rezultati za svaki algoritam.

Algoritam vektora linearne potpore ostvario je točnost od 85.68% uz vrijeme treniranja od 53.66 sekundi. 13937 komentara je označeno pozitivno, 6722 negativno te 11912 neutralno. Navedeno se može izraziti i pomoću postotaka, gdje je 42.79 % pozitivnih, 20.64 % negativnih i 36.57 % neutralnih komentara.

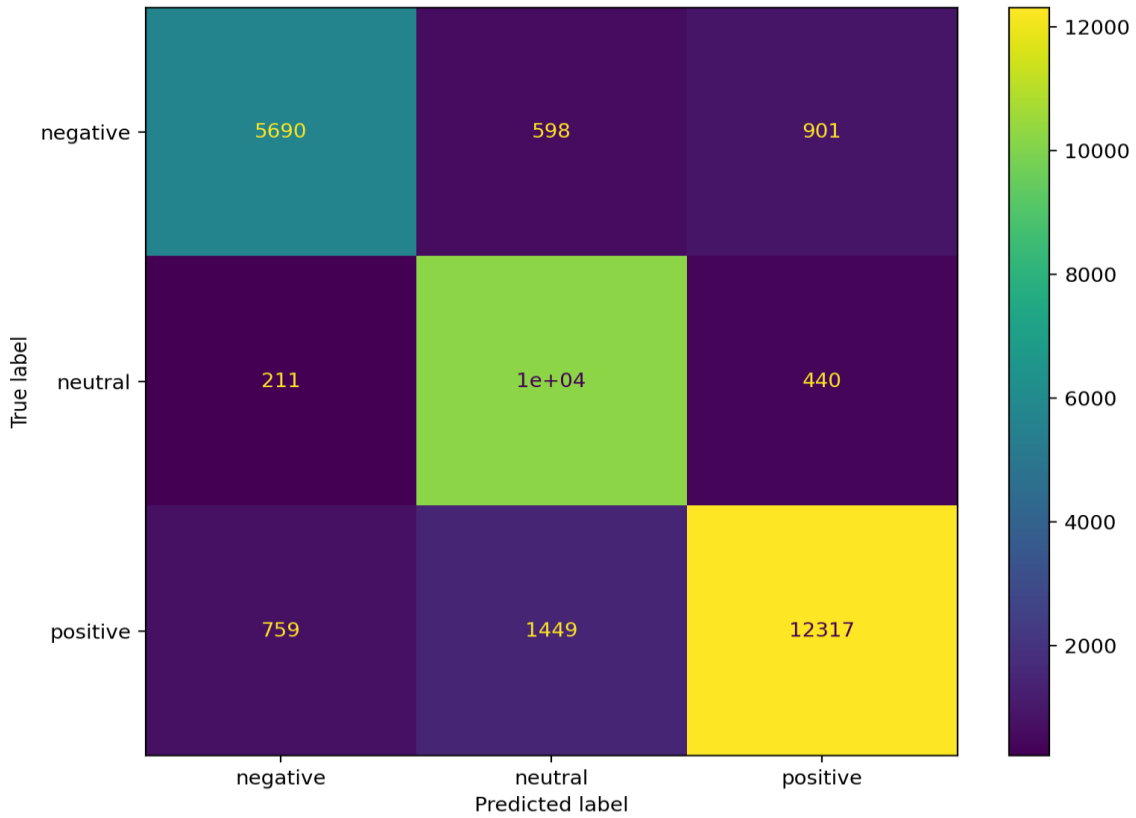


Slika 18. Matrica zabune vektora linearne potpore

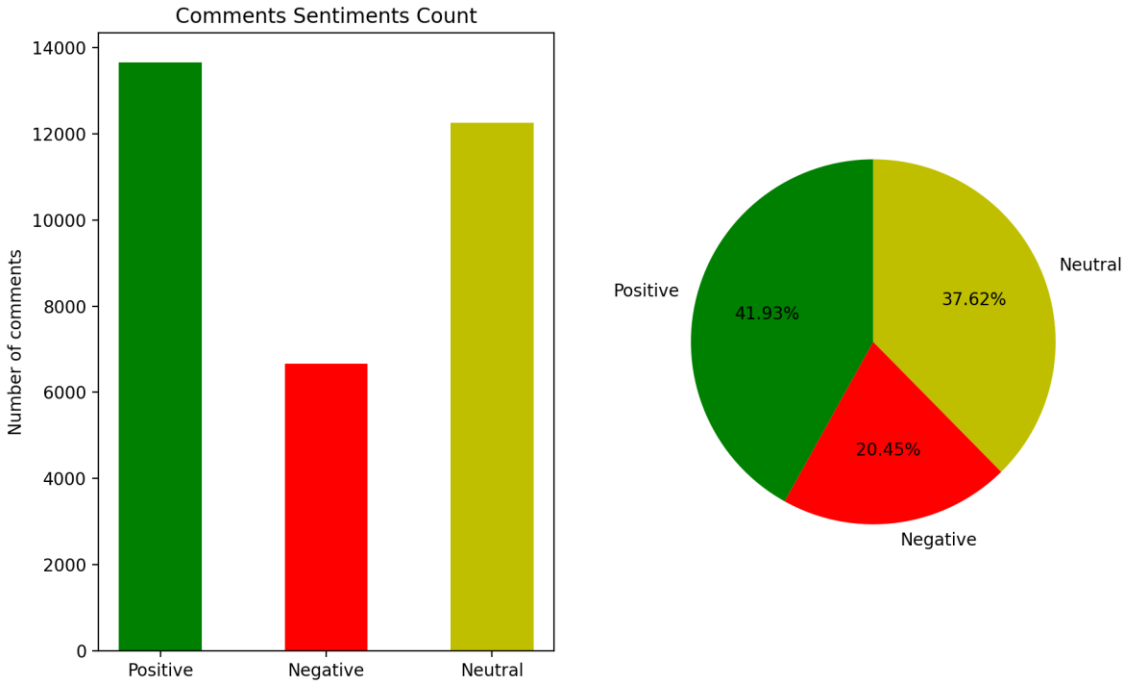


Slika 19. Stupčasti i kružni grafikon za vektor linearne potpore

Algoritam logističke regresije ostvario je točnost od 86.62 % uz vrijeme treniranja od 10.38 sekundi. 13658 komentara je označeno pozitivno, 6660 negativno te 12253 neutralno. Navedeno se može izraziti i pomoću postotaka, gdje je 41.93 % pozitivnih, 20.45 % negativnih i 37.62 % neutralnih komentara.

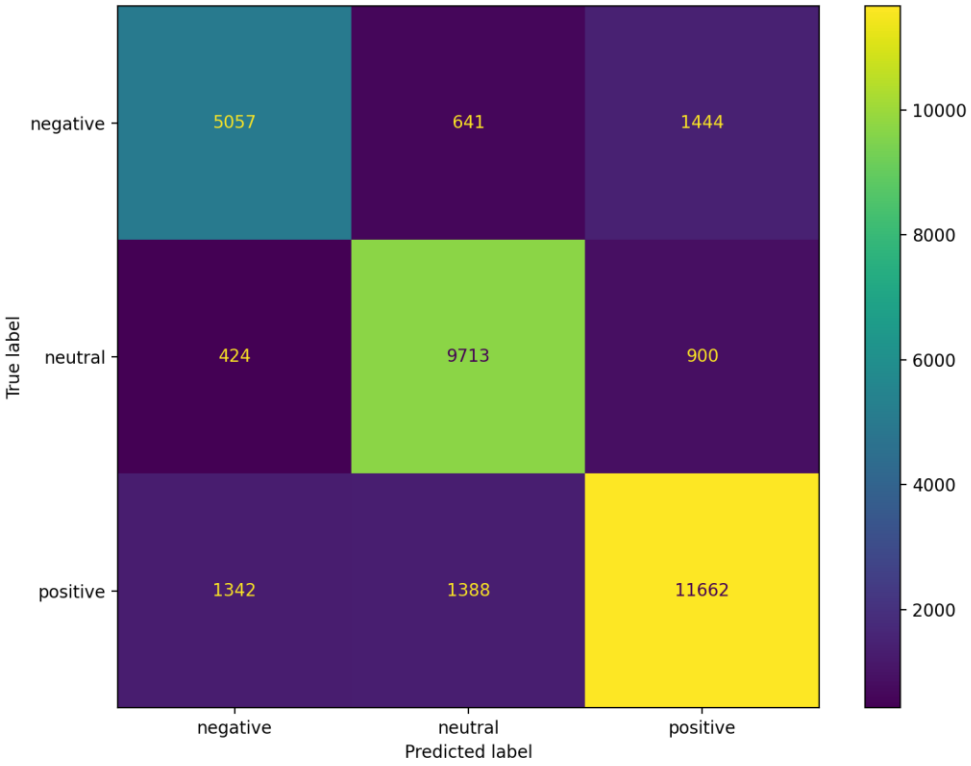


Slika 20. Matrica zabune logističke regresije

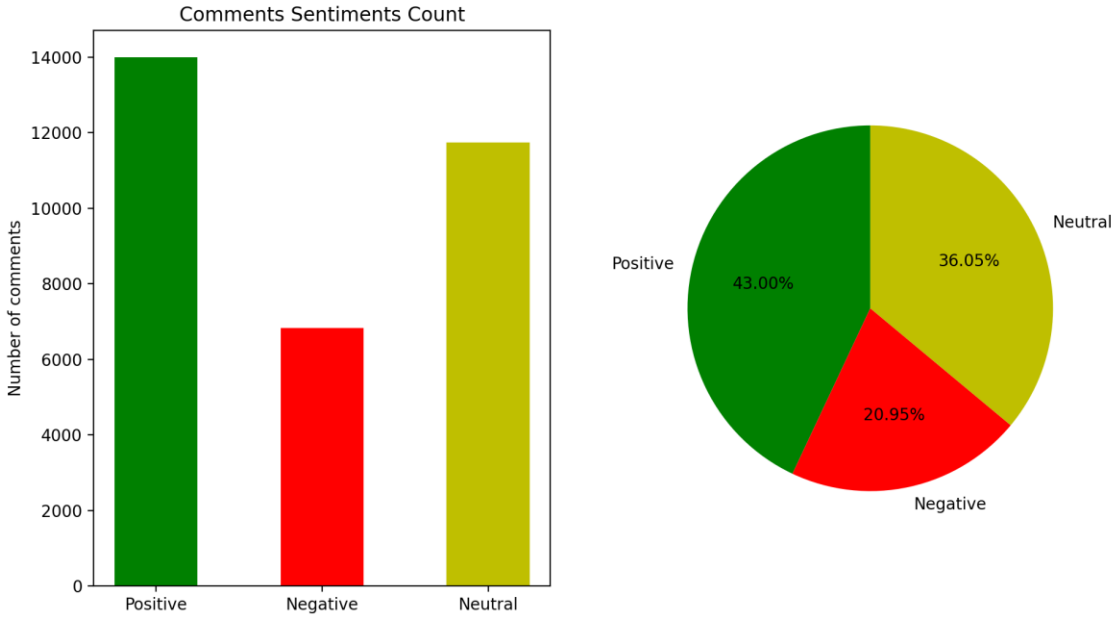


Slika 21. Stupčasti i kružni grafikon za logističku regresiju

Algoritam klasifikator stabla odluka ostvario je točnost od 81.15 % uz vrijeme treniranja od 178.91 sekundi što je skoro 3 minute. 14006 komentara je označeno pozitivno, 6823 negativno te 11742 neutralno. Navedeno se može izraziti i pomoću postotaka, gdje je 43.0 % pozitivnih, 20.95 % negativnih i 36.05 % neutralnih komentara.



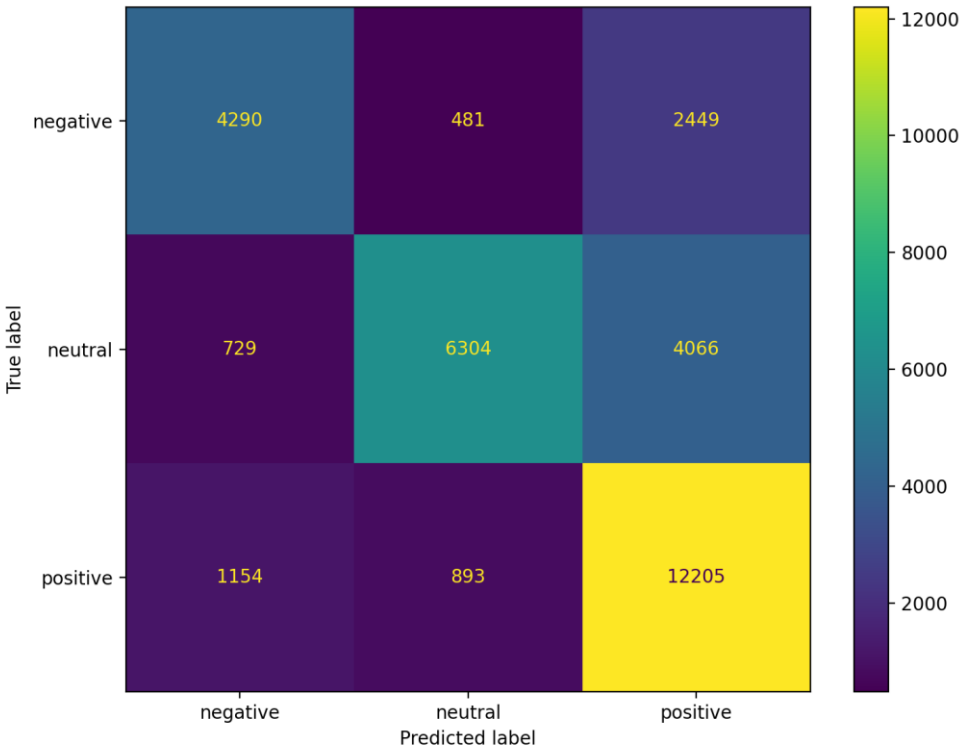
Slika 22. Matrica zabune klasifikatora stabla odluka



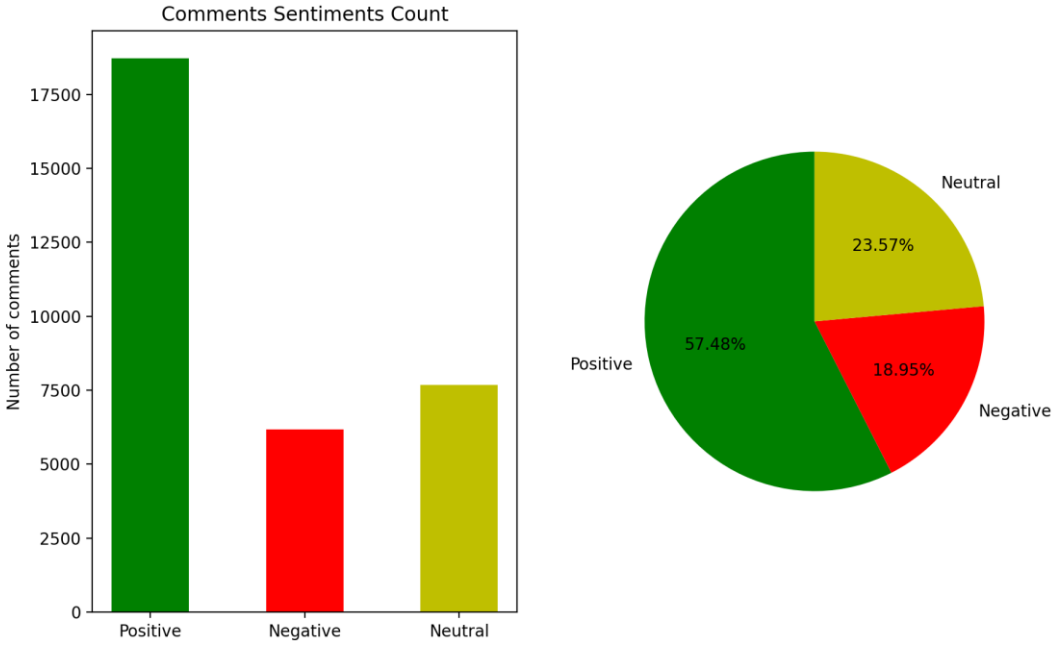
Slika 23. Stupčasti i kružni grafikon za klasifikator stabla odluka

Algoritam multinomial Naive Bayes ostvario je točnost od 70.0 % uz vrijeme treniranja od 0.31 sekundu. 18720 komentara je označeno pozitivno, 6173 negativno te 7678 neutralno. Navedeno

se može izraziti i pomoću postotaka, gdje je 57.47 % pozitivnih, 18.95 % negativnih i 23.57 % neutralnih komentara.

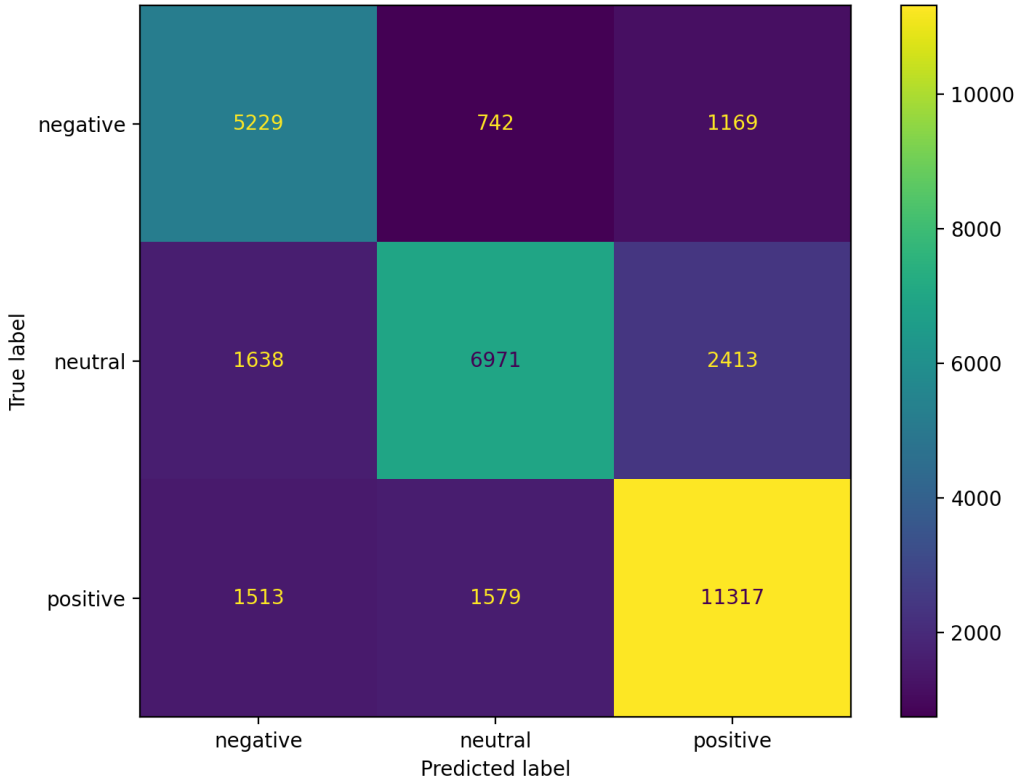


Slika 24. Matrica zabune za multinomial Naive Bayes

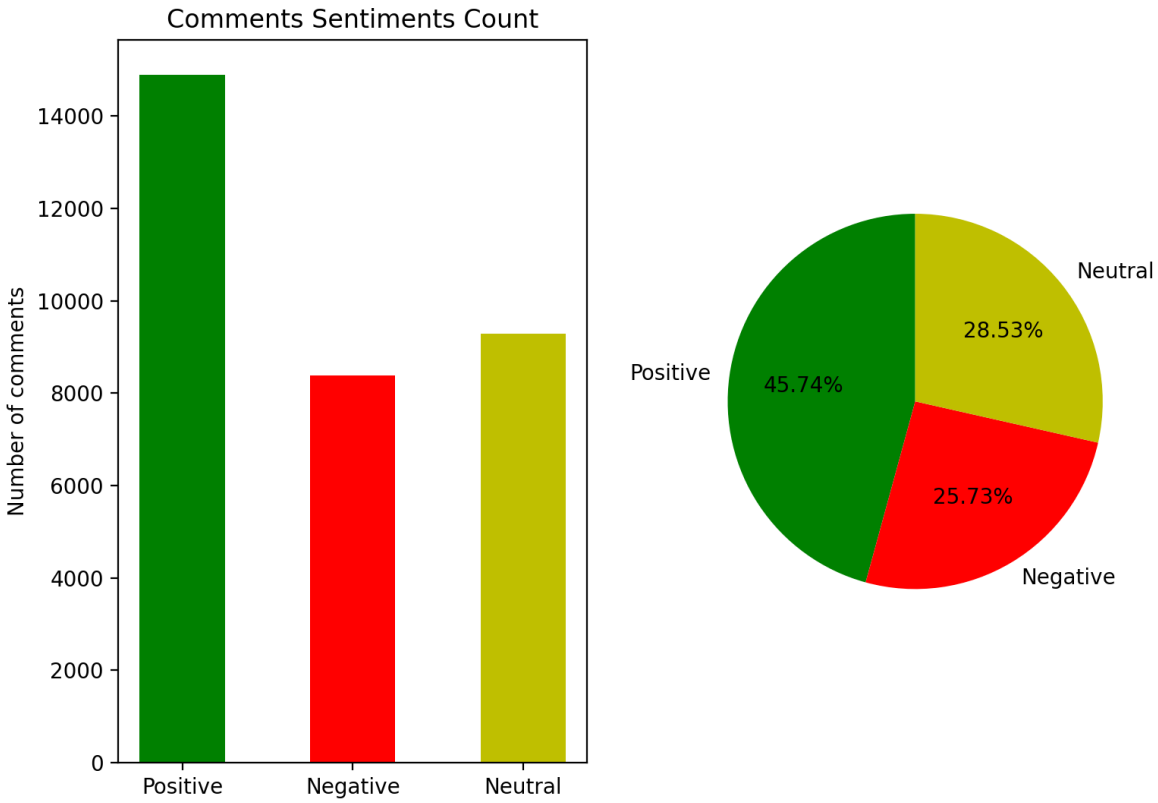


Slika 25. Stupčasti i kružni grafikon za multinomial Naive Bayes

Algoritam complement Naive Bayes ostvario je točnost od 72.2 % uz vrijeme treniranja od 0.27 sekundi. 14899 komentara je označeno pozitivno, 8380 negativno te 9292 neutralno. Navedeno se može izraziti i pomoću postotaka, gdje je 45.74 % pozitivnih, 25.73 % negativnih i 28.53 % neutralnih komentara.

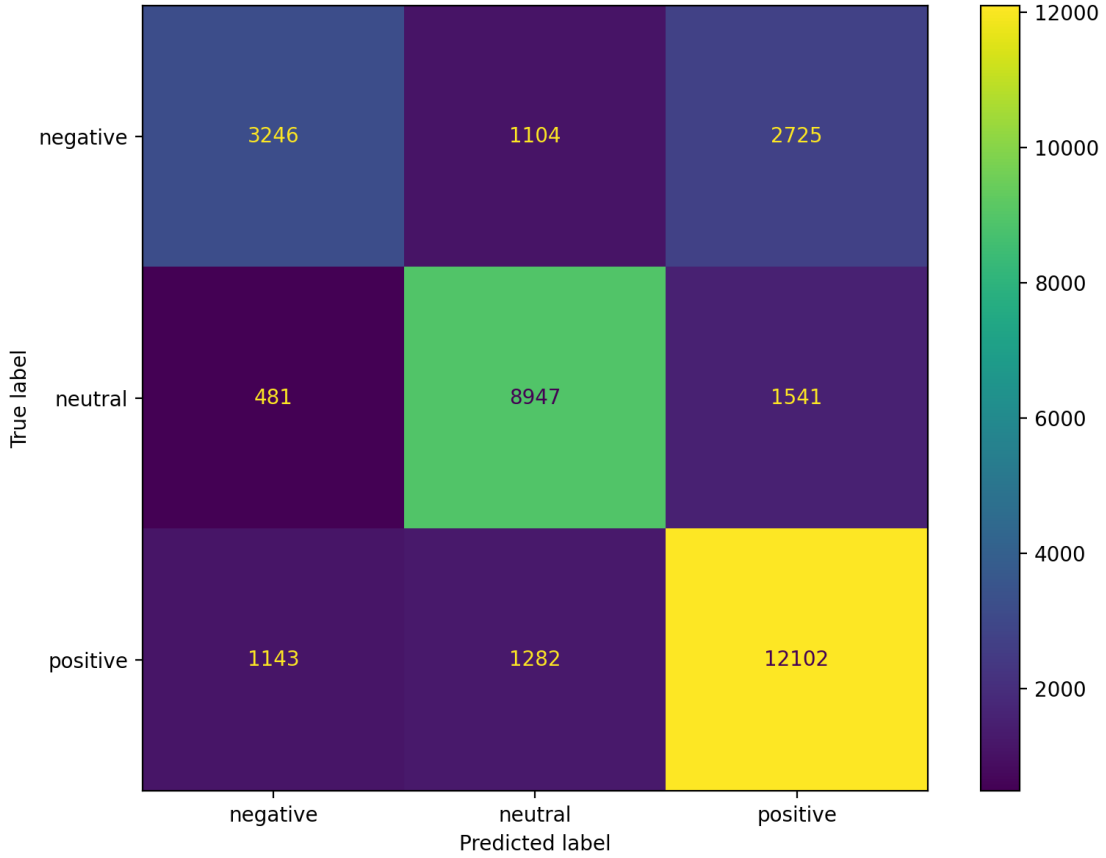


Slika 26. Matrica zabune za complement Naive Bayes

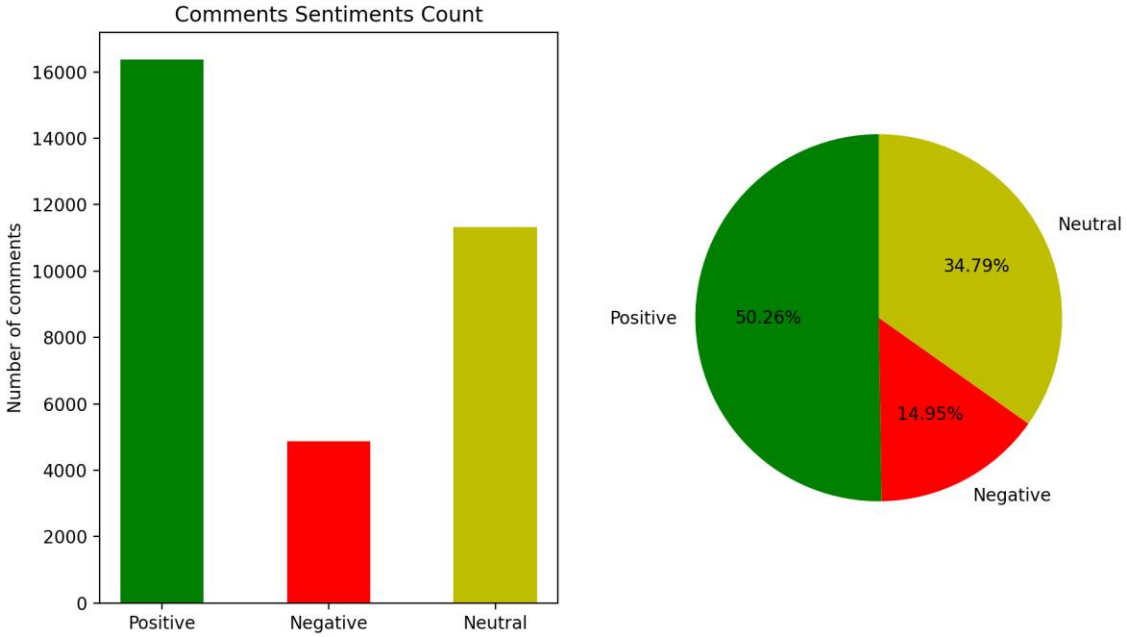


Slika 27. Stupčasti i kružni grafikon za complement Naive Bayes

Algoritam bernoulli Naive Bayes ostvario je točnost od 74.59 % uz vrijeme treniranja od 0.29 sekundi. 16368 komentara je označeno pozitivno, 4870 negativno te 11333 neutralno. Navedeno se može izraziti i pomoću postotaka, gdje je 50.25 % pozitivnih, 14.95 % negativnih i 34.79 % neutralnih komentara.



Slika 28. Matrica zabune za bernoulli Naive Bayes



Slika 29. Stupčasti i kružni grafikon za bernoulli Naive Bayes

Nakon pojedinačnog prikaza rezultata svakog algoritma potrebno je rezultate usporediti. Usporedba se radi na temelju točnosti i vremena treniranja.

Algoritam	Točnost [%]	Vrijeme treniranja [s]
Vektor linearne potpore	85.68	53.66
Logistička regresija	86.62	10.38
Klasifikator stabla odluka	81.15	178.91
Multinomijalni naivni Bayesov	70.0	0.31
Komplementni naivni Bayesov	72.2	0.27
Bernoulliev naivni Bayesov	74.59	0.29

Slika 30. Tablica rezultata svih korištenih algoritama

Na temelju tablice se može donijeti zaključak kako je logistička regresija ostvarila najveću točnost. Naive Byes algoritmi su daleko najbrži od svih, ali i imaju najlošiju točnost. Klasifikator stabla odluka se ne isplati koristi zbog prevelikog vremena treniranja bez obzira na točnost. Na temelju svega prethodno navedenog se može donijeti zaključak da je logistička regresija najbolji izbor, jer ima najefikasniji omjer točnost i vremena treniranja.

3.5. Youtube link

Jedna od bitnih značajki programa je da korisnik može unijeti željeni Youtube link i pogledati klasifikaciju komentara za njega.

```
def customYoutubeVideo():
    system('cls')
    print("You choose Custom Youtube Video.\n")
    youtubeLink = str(input("Enter Youtube Video ID: "))
    # regex to extract Youtube video ID from URL
    regex = re.compile(
        r'(https?://)?(www\.)?(youtube|youtu|youtube-nocookie)\.(com|be)/(watch\?v=|embed/|v/|.+\?v=)?(?P<id>[A-Za-z0-9\-=_]{11})')
    match = regex.match(youtubeLink)
    if not match:
        print('Invalid Youtube Link')
    else:
        # vided ID will be used as file name
        scrapeCommentsOnVideo(match.group('id'), path.join(
            'files\\raw\\', match.group('id') + '.csv'), None)
        makeValidCSV(match.group('id'))
        trainClassifierForYoutubeComments(match.group('id'))
```

Slika 31. Kod za izdvajanje video ID-a i poziv svih potrebnih funkcija

Pošto je za Youtube API [14] potrebno koristiti samo video ID iz linka, napravljen je regex koji će ga izdvojiti. Na primjer, u linku https://www.youtube.com/watch?v=videoID, videoID označuje mjesto gdje se uvijek nalazi video ID, točnije poslije ključne riječi watch?v=. Nakon odvajanja video ID-a, potrebno je preuzeti sve komentare sa željenog linka, a navedeno se postiže korištenjem funkcije scrapeCommentsOnVideo.

```python
def scrapeCommentsOnVideo(videoID, saveToFile, nextPageToken):
    os.environ["OAUTHLIB_INSECURE_TRANSPORT"] = "1"
    api_service_name = "youtube"
    api_version = "v3"
    DEVELOPER_KEY = os.environ.get('DEVELOPER_KEY')
    youtube = googleapiclient.discovery.build(
        api_service_name, api_version, developerKey=DEVELOPER_KEY)

    # Comments are ordered by time, which means that latest comments will be
    # Text format is set to plainText, beacuse it's way easier to obtain wanted data
    # The snippet object contains basic details about the comment thread
    request = youtube.commentThreads().list(
        order="time",
        part="snippet",
        videoId=videoID,
        maxResults=500,
        pageToken=nextPageToken,
        textFormat='plainText'
    )

    response = request.execute()
    nextPageToken = response.get('nextPageToken')
    # append to file, newline is neccessary in order to avoid adding comma after every letter
    # encoding is added beacuse of the inability to read response from API
    with open(saveToFile, 'a', newline='', encoding="utf-8") as file:
        fileWriter = writer(file)
        for index in range(len(response.get('items'))):
            # response return very complex dictionary object, so to get to comments this is the path
            fileWriter.writerow([response.get('items')[index].get('snippet').get(
                'topLevelComment').get('snippet').get('textDisplay')])
        file.close()
    # recursive call if there is a next page of comments
    if(nextPageToken):
        scrapeCommentsOnVideo(videoID, saveToFile, nextPageToken)
```

Slika 32. Kod za preuzimanje komentara sa Youtube linka

Korištena funkcija je dorađena funkcija preuzeta sa službene stranice Youtube API. Za početak je komentare potrebno poredati tako da najnoviji idu prvi, rezultate povećati na 500 kako bi se što brže preuzelo sve komentare, i format postaviti na obični tekst kako bi se moglo što lakše pronaći željene komentare. Parametar pageToken je bitan jer se trebaju preuzeti svi komentari sa stranice, a jednim zahtjevom se može preuzeti samo dio komentara. Koristeći prethodno navedeni parametar funkcija se rekurzivno poziva sve dok se ne preuzmu svi komentari sa linka. Nakon preuzimanja, potrebno je izdvojiti komentare iz odgovora te ih zapisati u datoteku.

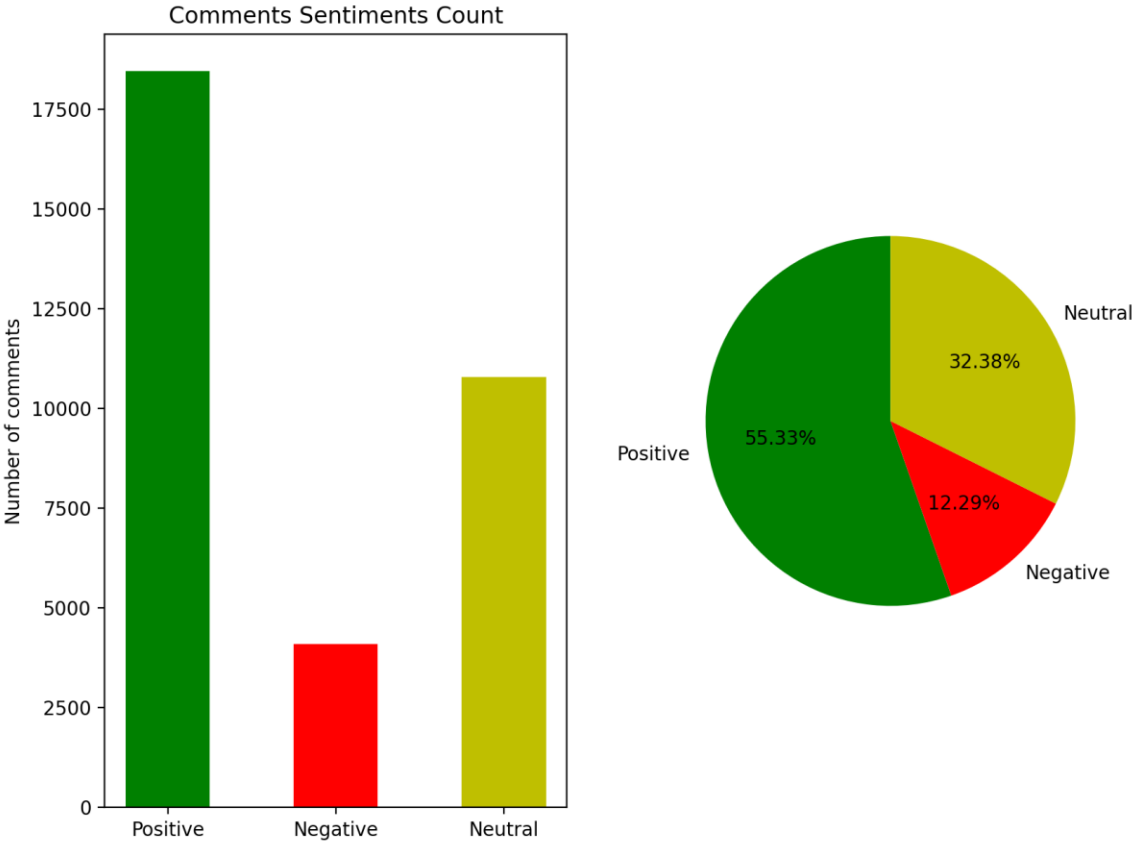
Poslije preuzimanja, potrebno je očistiti i pripremiti podatke u datoteci za korištenje, a navedeno je objašnjeno u poglavlju 3.2.

```
def trainClassifierForYoutubeComments(videoID):
    trainingData = loadTrainingData()
    inputData = loadInputData(videoID)

    trainingComments = transformComments(trainingData)
    inputComments = transformComments(inputData)
    polarities = transformPolarity(trainingData)

    model = LogisticRegression()
    trainingComments.resize(
        (trainingComments.shape[0], inputComments.shape[1]))
    model.fit(trainingComments, polarities)
    prediction = model.predict(inputComments)
    # write predicted polarities in CSV file
    for index in range(len(inputData)):
        inputData.at[index, 'polarity'] = prediction[index]
    inputData.to_csv(path.join('files\\cleaned\\', videoID +
                               '.csv'), index=False, float_format='%.0f')
    filePath = path.join('files\\cleaned\\', videoID + '.csv')
    getAllDataMetrics(filePath)
```

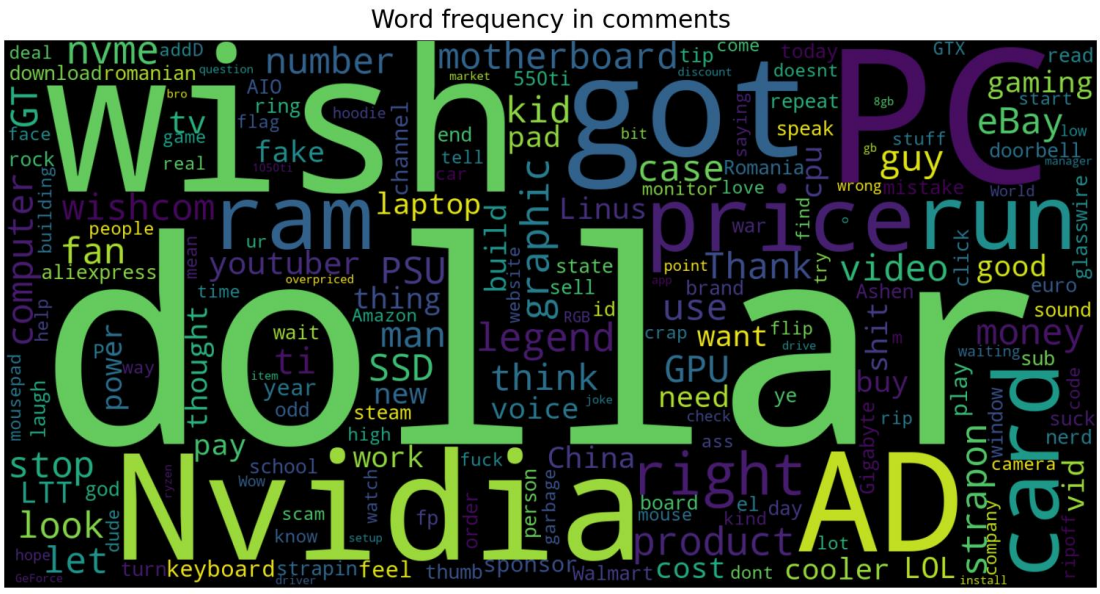
Slika 33. Kod modela za predikciju klasa komentara na Youtube linku

Razlog odabira logističke regresije objašnjen je u poglavlju 3.4. Prvo je potrebno prilagoditi podatke za treniranje na temelju broja komentara koji su preuzeti u prethodnom koraku. Nakon što model napravi pretpostavke vrijednosti, potrebno ih je zapisati u datoteku kako bi se mogle pravilno vizualizirati.

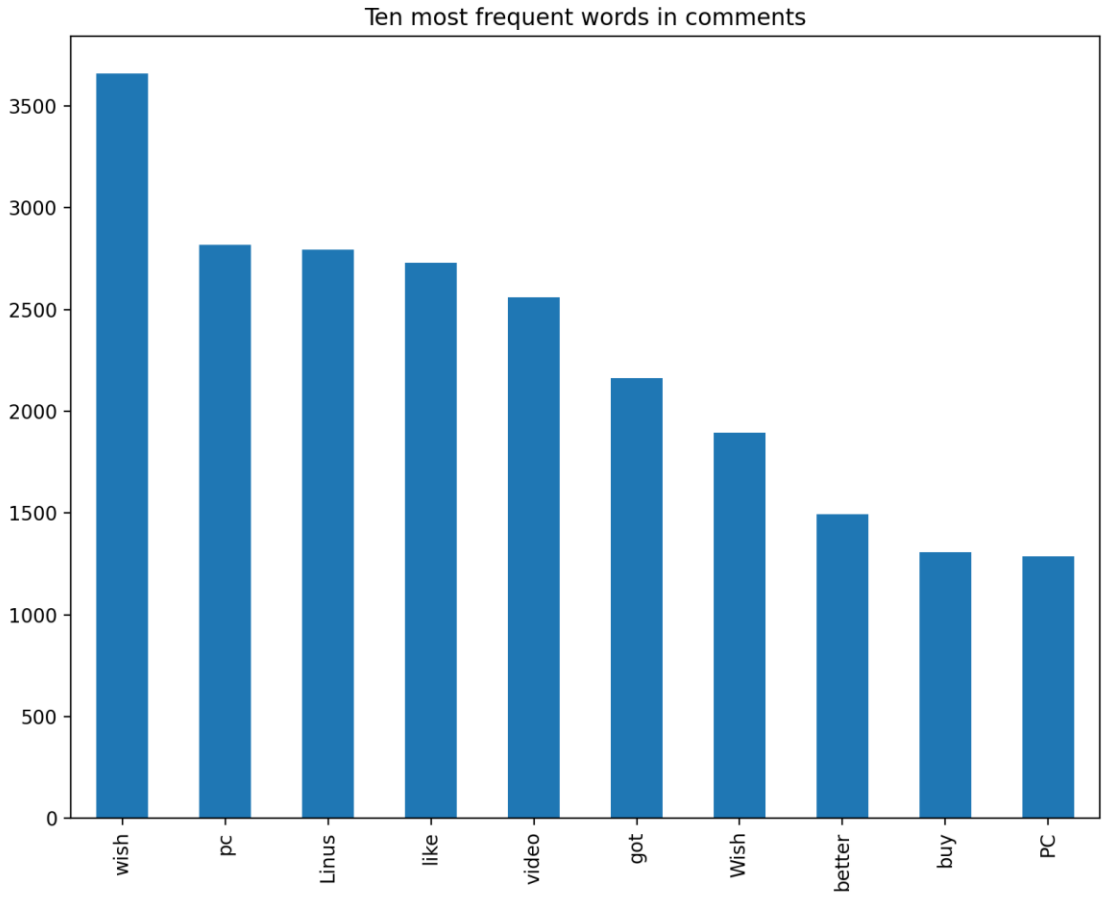
U idućem primjeru prikazani su rezultati klasifikacije komentara na videu *Building a PC... using only Wish.com* od *Linus Tech Tips* kanala [15]. Sveukupno je pronađeno 33364 komentara, gdje je 18462 komentara je označeno pozitivno, 4100 negativno te 10802 neutralno. Navedeno se može izraziti i pomoću postotaka, gdje je 55.34 % pozitivnih, 12.29 % negativnih i 32.38 % neutralnih komentara. Rezultati su zadovoljavajući jer je video dobro prihvaćen od strane publike zbog visokog nivoa kreativnosti. *Linus Tech Tips* kanal je također poznat to visokoj kvaliteti sadržaja, što potvrđuje visok postotak pozitivnih komentara.



Slika 34. Stupčasti i kružni grafikon za Youtube link



Slika 35. Oblak riječi za Youtube link



Slika 36. Stupčasti grafikon frekvencije riječi za Youtube link

4. ZAKLJUČAK

Cilj seminarskog rada bio je proučiti, istražiti i demonstrirati upotrebu raznih algoritama u svrhu klasifikacije Youtube komentara. Rezultati svih algoritama su vizualizirani, objašnjeni i uspoređeni te je donesen zaključak koji algoritam je najbolje koristiti. Logistička regresija prema rezultati ima najbolji omjer točnosti i vremena treniranja. Sve algoritme je jednako zahtjevno implementirati i koristiti. Programsko rješenje se može implementirati u raznim mobilnim i web aplikacijama koje bi puno pomogle i olakšale posao Youtube kanalima kada im je potrebna brza i jednostavna vizualizacija odaziva publike na novi video. Strojno učenje može biti od velike pomoći u mnogim područjima znanosti, jer predstavlja efikasan i brz način rješavanja problema koji bi ljudima oduzeli puno vremena i ne bi bili jednako kvalitetno riješeni. Ovaj seminarski rad je samo jedan od primjera rješenja koja mogu svakodnevno pomoći u obavljanju monotonih zadataka.

LITERATURA

[1] https://www.ibm.com/cloud/learn/machine-learning

[2] https://developers.google.com/machine-learning/guides/text-classification/images/TextClassificationExample.png

[3] https://machinelearningmastery.com/types-of-classification-in-machine-learning

[4] https://peopleanalytics-regression-book.org/multinomial-logistic-regression-for-nominal-category-outcomes.html

[5] https://scikit-learn.org/stable/modules/naive_bayes.html

[6] https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

[7] https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm.png

[8] https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm3.png

[9] https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm4.png

[10] https://static.javatpoint.com/tutorial/machine-learning/images/support-vector-machine-algorithm5.png

[11] https://www.sciencedirect.com/topics/computer-science/decision-tree-classifier

[12] https://www.kaggle.com/cosmos98/twitter-and-reddit-sentimental-analysis-dataset

[13] https://selmirkalender.com/2021/07/07/vizualizacijapodataka/

[14] https://developers.google.com/youtube/v3/docs/commentThreads/list?apix_params=%7B%22textFormat%22%3A%22plainText%22%7D

[15] https://www.youtube.com/watch?v=eQ_8F4nzyiw

POPIS SLIKA