Univerzitet u Sarajevu
Elektrotehnički fakultet
Odsjek za Računarstvo i informatiku
Dubinska analiza podataka
Dokumentacija seminarskog rada

Predicting Stock Market Movement Based on Twitter Data and News Articles Using Sentiment Analysis and Fuzzy Logic

Džigal Džemil
Hodo Midhat
Pirija Ena

Sarajevo, juni 2021

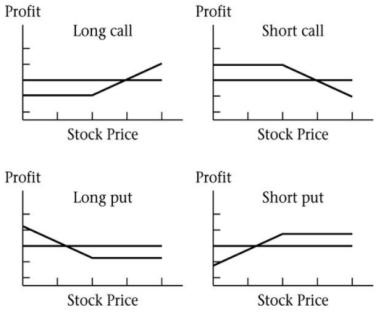
1) **Odabir problema za rješavanje, motivacija, specifikacija problema koji se rješava.**

Džemil:

Investiranje u dionice je dugogodišnja i temeljno ustanovljena struka koja zahtijeva godine studiranja i učenja na greškama kako bi osoba mogla biti iole uspješna u istoj. Generalno, dionice mogu da rastu i analogno tome, da opadaju. Najveći problem je predvidjeti u kojem smjeru će se kretati cijena dionice. Shodno tome, postoje razni mehanizmi "klađenja" za i protiv neke određene dionice. Ta "klađenja" se zovu stock calls i stock puts. Kupovanje stock calls je, pojednostavljeno, klađenje da će do isteka tog call-a cijena dionice porasti do i iznad cijene X. Kupovanje stock puts je, pojednostavljeno, klađenje da će do isteka tog put-a cijena dionice opasti do i ispod cijene X. Jasno je odakle dolazi faktor "klađenje". Shodno tome, za maksimizaciju profita, potrebno je implementirati sistem za predlaganje koji korisniku implicitno predlaže da li treba kupiti short-term call ili put na određenu dionicu.

Ena:

Jasno je iz prethodnog izlaganja da je vrijeme isticanja call i put poziva u uskoj vezi sa mogućim profitom nakon uspjeha call ili put-a. Što je vrijeme isticanja call-a i put-a kraće, to je rizik veći zbog prirodne fluktuacije svake dionice ali profiti mogu biti veliki. Analogno tome duga vremena trajanja rezultiraju skupljim call i put-ovima ali većom vjerovatnoćom za profit koji je, u ovom slučaju, manji nego za kraće call-ove i put-ove. Očigledno je da sistem koji predviđa smjer kretanja cijene dionice može primarno pomoći za kupovinu call-ova i put-ova sa kraćim vremenskim trajanjem.



Midhat:

Jedna dionica koja objedinjuje stanje na svjetskoj berzi u odnosu na prvih 500 kompanija po cijeni njihovih dionica. Ovaj indeks, historijski, dobro opisuje kretanje čitave berze, primarno u SAD-u. Tako da ako možemo formirati neki model koji određuje short-term kretanje cijene ovog indeksa, možemo kupovati call-ove i put-ove za ovaj indeks. Shodno tome, potrebno je prikupiti

informacije koje generalno opisuju dionice. Jedan koristan izvor ovih informacija je Twitter. Pokazano je da twitter postovi su u korelaciji sa kretanjem cijena SNP500 indeksa.

Sve ukazuje na to da je moguće riješiti problem predviđanja kretanja cijena dionica na berzi tako što ćemo iskorisiti Twitter objave u sistemu kako bi se moglo odrediti kretanje berze, shodno tome i kretanje SNP500 indeksa. Dalje je jednostavno inferirati šta treba i kada treba kupiti. Ovakav sistem je više prikladan Sistemima podrške odlučivanju i možemo ga posmatrati kao kompromis između klasičnih ML problema čija rješenja su eksplicitna i opštih, sugestivnih problema ML-a čija rješenja su generalni prijedlozi šta treba uraditi (iz razloga što dobijamo sugestiju u kojem smjeru će se kretati cijene dionica odakle možemo inferirati koju odluku trebamo donijeti kao akter, odnosno investitor).

Džemil, Ena, Midhat:
Zaključujemo da bi ovaj sistem bio najviše koristan običnim investitorima, amaterima. Također, i veliki investitori u hedge fund kompanijama mogu koristiti sistem za svrhe poteza “velike kocke” koji se često dešavaju u ovim kompanijama, s obzirom na to da je uvijek u pitanju kompanija čiji kapital je reda milijardi USD.

2) **Odabir naučnog rada koji je povezan sa temom i analiza istog.**

Džemil, Ena, Midhat:
Rad koji smo odabrali za analizu i iz kojeg smo preuzeli mnoge ideje za implementaciju svojeg rada je Predicting Stock Market Movement Based on Twitter Data and News Articles Using Sentiment Analysis and Fuzzy Logic, Roshni et. al.

Midhat:
Uvod i Related Work:
U uvodu autori opisuju veliki uticaj socijalnih mreža na razne grane industrije. Jedna od tih grana je ulaganje u dionice. Autori ističu činjenicu da objave na socijalnim mrežama mogu formirati “kolektivno mišljenje” o stanju na berzi koje se može koristiti kako bi se kretanje iste moglo predviđati. Također spominju da kretanje Dow Jones indeksa može biti predviđeno sa tačnosti 87.6% koristeći samo Twitter podatke. Napominju da veliki uticaj na cijene dionica imaju poznate ličnosti, muzičari, poduzetnici, generalno bogati ljudi. Ovo ima smisla jer je poznato da ljudi sa uticajem mogu da sa tim svojim uticajem značajno utiču na opadanje odnosno rast cijena

određenih dionica i resursa. Ovo se očituje u posljednjem rastu i padu cijena kriptovaluta za čije haotično ponašanje se "glavnim krivcem" smatra poduzetnik Elon Musk.

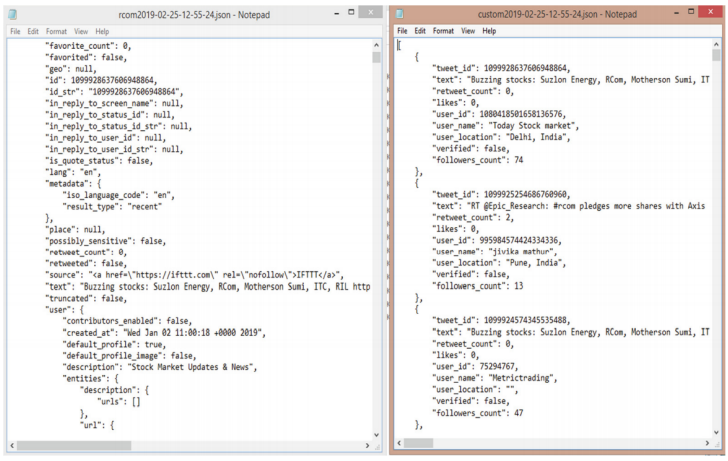
Dalje, autori opisuju sentimentalnu analizu u smislu rješavanja problema ternarne klasifikacije. Opisuju da ova analiza može biti iskorištena za formiranje robustnog ML modela sa većom tačnosti i najavljuju grafike performansi određenih modela koje su iskoristili. U dijelu Related Work opisuju razne prijašnje autore koji su iskoristili Random Forest klasifikatore, linearnu regresiju i SVM za postizanje sličnih rezultata, ali sa manjom tačnošću.

Ena:

Data Acquisition i Data Preprocessing:

U Data Acquisition dijelu autori opisuju način prikupljanja podataka koristeći Twitter-ove API-eve. Koristili su poznati paket tweepy i učitavali su mnogo tweet-ova u JSON formatu. Ovim načinom su formirali bazu podataka koja predstavlja bazu znanja njihovog sistema. Naravno, ove podatke je potrebno obraditi. Autori su to opisali u dijelu Data Preprocessing u kojem navode da je potrebno ukloniti razne "problematične" simbole kao što su URL-ovi, EMOJI simboli i slični simboli koje smo i mi morali uklanjati pri radu sa tekstom na ovom predmetu. Dalje, tokenizirali

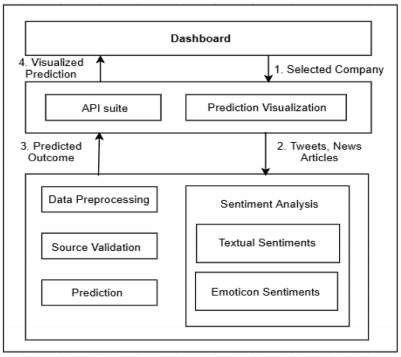
su tekst tweet-ova i, koristeći regularne ekspresije, izvršili su pattern matching kako bi uklonili hashtag-ove i spominjanje određenih korisnika, zamijenivši ih kao (#Nestle u Nestle i @DonaldTrumph u user). Ovo ima za cilj maksimalno poopštenje konteksta teksta tweet-a koji se analizira.



Džemil:

Sentiment Analysis i Implementation:

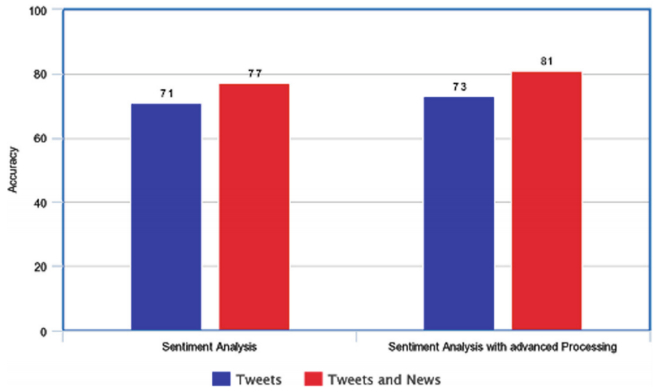
Autori su izvršili sentimentalnu analizu koristeći NLTK biblioteku. Konkretno iskoristili su Naive Bayes Classifier unutar NLTK-a kako bi izvršili analizu. Dalje, izvršili su shallow parsing da odrede glavne dijelove rečenica kako bi se umanjio efekat outlier riječi. U dijelu preprocessing-a su uklonili određene emoji-e ali s obzirom na to da za ovu analizu treba i posljedica, odnosno labels za svaki tweet, određeni emoji-i se koriste kako bi se odredio sentiment tog tweet-a. Također, prioritet su imali tweet-ovi koji imaju veliki broj retweet-ova. Obrazložili su da tweet-ovi koji su više puta retweet-ovani imaju veći uticaj od drugih, što ima smisla jer se tim povećava doseg tog tweet-a pa i uticaj tog tweet-a na cijenu dionica, kao i tumačenje u kojem smjeru treba ići cijena dionice. Tweet-ovi korisnika sa manje od 100 follower-a se odbacuju jer nemaju dovoljno veliki doseg širokih narodnih masa. Predložili su izgled arhitekture sistema za konačnu kalkulaciju sentimenta odnosno odluke u kojem smjeru će se kretati cijena dionica na berzi.



Što se tiče same implementacije, autori opisuju korake njihovog pristupa. Počinju sa akvizicijom podataka kako je opisano u dijelu prije ovog. Preprocesiraju te podatke, također kako je opisano u dijelu prije ovog. Dalje, izvršavaju sentimentalnu analizu koristeći tehniku sa emoji-ima koji

opisuju sentiment tog tweet-a i formiraju inicijalni zaključak o sentimentu tweet-a. Ovdje dolaze do zaključka da samo jedan izvor neće omogućiti maksimalnu tačnost. Zaključuju da trebaju uključiti i podatke iz novina kako bi povećali tačnost. Ovim formiraju hibridni NLP sistem sa dvije baze znanja (tweet-ovi i novinski članci) i koriste ih kako bi zaključili sentiment ovih podataka.

Džemil, Midhat i Ena:
Results and Discussion:
Kako je opisano u prošlom odjeljku, rezultati ukazuju na to da uključivanje novinskih članaka znatno povećava tačnost predikcije sentimentalne analize.



Ukazuju na značajno povećavanje tačnosti, u prosjeku oko 7-10%, u ovisnosti od toga koji podaci se koriste za analizu. U dijelu diskusije (zaključak i budući rad) autori ukazuju na moć RNN-ova za povećavanje tačnosti. Shodno njihovoj osobini da razmatraju eksponencijalno više različitih scenarija u odnosu na klasične ML algoritme i metode. Ovu metodu ostavljaju za dalji rad pri čemu sugerišu da je moguće postići tačnosti oko 81%. Ne opisuju kako se dolazi do te tačnosti, ali uz pomoć RNN-ova nije teško zaključiti kako (uz dovoljno velike mreže). Također, predlažu fuzzy sisteme regulacije za adaptaciju parametara RNN-ova kako bi se povećala tačnost istih. Također, zbog korištenja tweepy paketa, predlažu da uz dodatnu promjenu data acquisition koraka je moguće predvidjeti kretanje cijene dionica konkretnih kompanija, a ne samo SNP 500 indeksa i sličnih indeksa metrika svjetske berze.

3) **Prikupljanje i odabir podataka za sistem specificiran u 1. Priprema i preprocesiranje podataka (priložiti dokumentaciju i kodove za ovaj korak). Odabir algoritama mašinskog i dubokog učenja za rješavanje problema. Tabelarni prikaz algoritama sa naznakom ko je od članova tima zadužen za detaljniju analizu i implementaciju pojedinog algoritma.**

Za podatke je korišten sljedeći dataset: 📈 Stock Market Tweet | Sentiment Analysis lexicon | Kaggle

Što se tiče preprocesiranja podataka korištena su sljedeća pomagala:

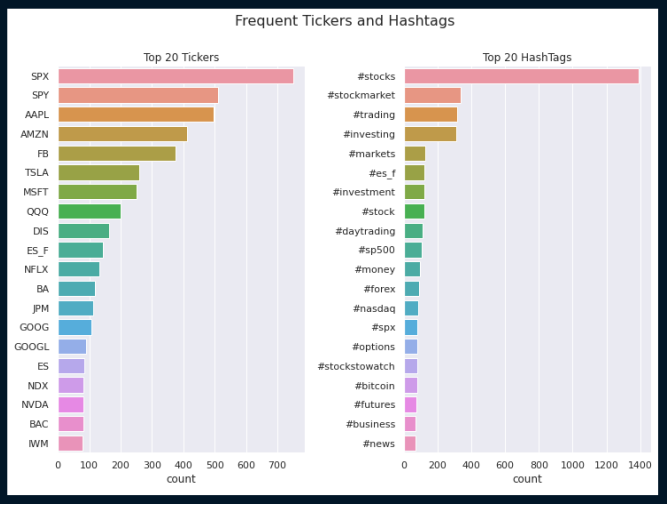
```
import warnings
warnings.simplefilter('ignore')
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

from pprint import pprint
from datetime import datetime
import collections
import re

import nltk
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
from nltk.corpus import wordnet, stopwords
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize

from wordcloud import WordCloud
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /home/chilichipsba/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /home/chilichipsba/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /home/chilichipsba/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

```
ticker_pattern = re.compile(r'(^\$[A-Z]+|^\$ES_F)')
ht_pattern = re.compile(r'#\w+')

ticker_dic = collections.defaultdict(int)
ht_dic = collections.defaultdict(int)

for text in data['text']:
    for word in text.split():
        if ticker_pattern.fullmatch(word) is not None:
            ticker_dic[word[1:]] += 1

        word = word.lower()
        if ht_pattern.fullmatch(word) is not None:
            ht_dic[word] += 1
```

Također, prikazali smo najpopularnije termine koristeći sljedeći kod:

```
[7]
ticker_df = pd.DataFrame.from_dict(
    ticker_dic, orient='index').rename(columns={0:'count'})\
    .sort_values('count', ascending=False).head(20)

ht_df = pd.DataFrame.from_dict(
    ht_dic, orient='index').rename(columns={0:'count'})\
    .sort_values('count', ascending=False).head(20)

fig, ax = plt.subplots(1, 2, figsize=(12,8))
plt.suptitle('Frequent Tickers and Hashtags', fontsize=16)
plt.subplots_adjust(wspace=0.4)

sns.barplot(x=ticker_df['count'], y=ticker_df.index, orient='h', ax=ax[0])
ax[0].set_title('Top 20 Tickers')

sns.barplot(x=ht_df['count'], y=ht_df.index, orient='h', ax=ax[1])
ax[1].set_title('Top 20 HashTags')

plt.show()
```



Te, zamijenili smo određene ključne riječi sa uniformnim formama tih riječi

```
charonly = re.compile(r'[^a-zA-Z\s]')
handle_pattern = re.compile(r'@\w+')
emoji_pattern = re.compile("["
                           u"\U0001F600-\U0001F64F"  # emoticons
                           u"\U0001F300-\U0001F5FF"  # symbols & pictographs
                           u"\U0001F680-\U0001F6FF"  # transport & map symbols
                           u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
                           u"\U00002702-\U00002780"
                           u"\U000024C2-\U0001F251"
                           "]+", flags=re.UNICODE)
url_pattern = re.compile(
    'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+')
pic_pattern = re.compile('pic\.twitter\.com/.{10}')
special_code = re.compile(r'(&amp;|&gt;|&lt;)')
tag_pattern = re.compile(r'<.*/>')

STOPWORDS = set(stopwords.words('english')).union(
    {'rt', 'retweet', 'RT', 'Retweet', 'RETWEET'})

lemmatizer = WordNetLemmatizer()

def hashtag(phrase):
    return ht_pattern.sub(' ', phrase)

def remove_ticker(phrase):
    return ticker_pattern.sub('', phrase)

def specialcode(phrase):
    return special_code.sub(' ', phrase)

def emoji(phrase):
    return emoji_pattern.sub(' ', phrase)

def url(phrase):
    return url_pattern.sub('', phrase)

def pic(phrase):
    return pic_pattern.sub('', phrase)

def html_tag(phrase):
    return tag_pattern.sub(' ', phrase)

def handle(phrase):
    return handle_pattern.sub('', phrase)
```

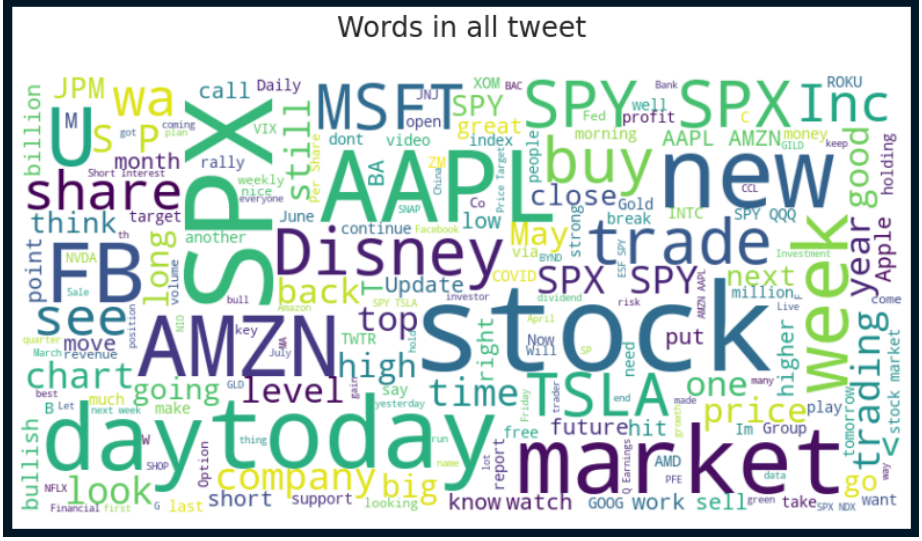
```
def decontracted(phrase):
    # specific
    phrase = re.sub(r"won\'t", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # DIS, ticker symbol of Disney, is interpreted as the plural of "DI"
    # in WordCloud, so I converted it to Disney
    phrase = re.sub('DIS', 'Disney', phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"(he|He)\'s", "he is", phrase)
    phrase = re.sub(r"(she|She)\'s", "she is", phrase)
    phrase = re.sub(r"(it|It)\'s", "it is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"(\'ve|has)", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

def onlychar(phrase):
    return charonly.sub('', phrase)

def remove_stopwords(phrase):
    return " ".join([word for word in str(phrase).split()\
                     if word not in STOPWORDS])

def tokenize_stem(phrase):
    tokens = word_tokenize(phrase)
    stem_words =[]
    for token in tokens:
        word = lemmatizer.lemmatize(token)
        stem_words.append(word)
    buf = ' '.join(stem_words)
    return buf
```

Te pokrenuli te filtere

```
def arrange_text(ds):
    ds['text2'] = ds['text'].apply(emoji)
    ds['text2'] = ds['text2'].apply(handle)
    ds['text2'] = ds['text2'].apply(specialcode)
    ds['text2'] = ds['text2'].apply(hashtag)
    ds['text2'] = ds['text2'].apply(url)
    ds['text2'] = ds['text2'].apply(pic)
    ds['text2'] = ds['text2'].apply(html_tag)
    ds['text2'] = ds['text2'].apply(onlychar)
    ds['text2'] = ds['text2'].apply(decontracted)
    ds['text2'] = ds['text2'].apply(onlychar)
    ds['text2'] = ds['text2'].apply(tokenize_stem)
    ds['text2'] = ds['text2'].apply(remove_stopwords)
```

Prikazali smo sve riječi u tweet-ovima

```
words = ' '.join([text for text in data['text2']])
wordcloud = WordCloud(
    width=800, height=400, background_color='white', max_font_size=110)\
    .generate(words)

plt.figure(figsize=(14, 7))
plt.imshow(wordcloud, interpolation="bilinear")
plt.title('Words in all tweet\n', fontsize=24)
plt.axis('off')
plt.show()
```



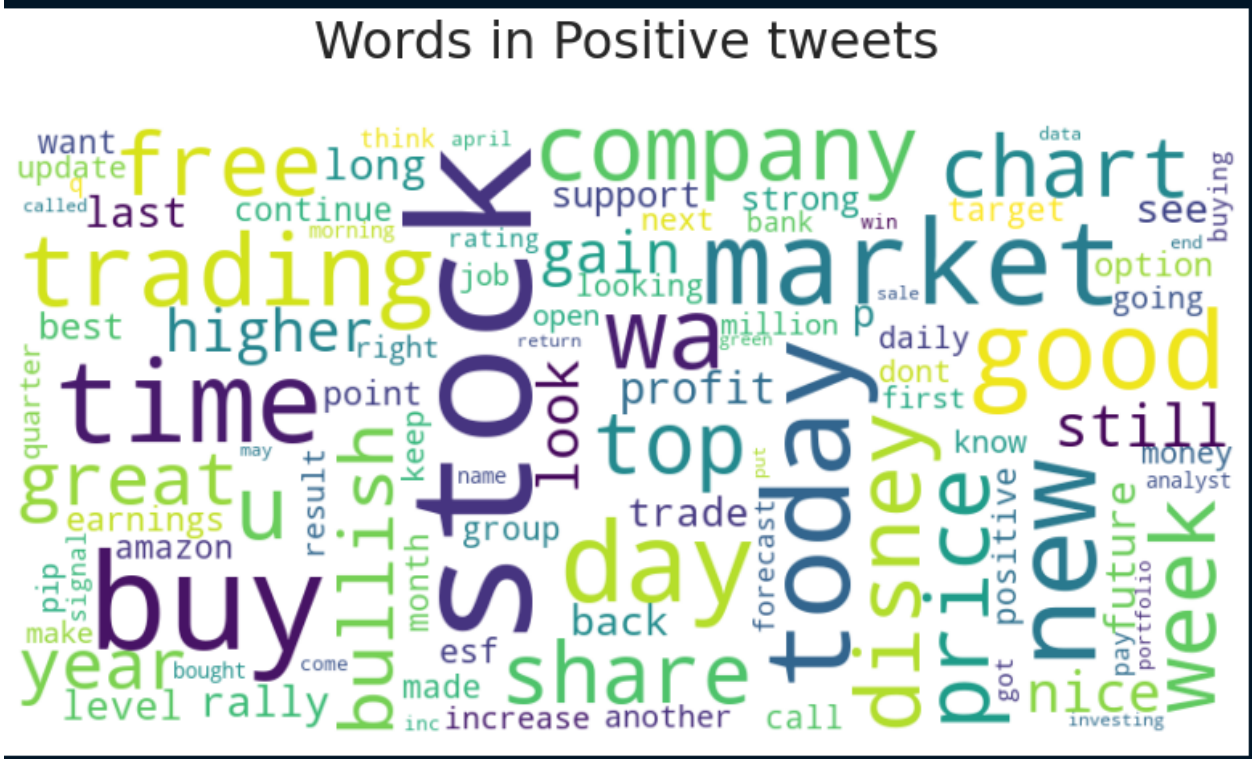
Dalje, napisali smo funkciju koja će formirati word cloud po sentimentima koje želimo:

```
def wordcloud_by_sentiment(sentiment):
    not_ticker = []

    for text in data[data['sentiment']==sentiment]['text2']:
        for word in text.split():
            if word.upper() not in ticker_dic:
                not_ticker.append(word.lower())

    words = ' '.join([word for word in not_ticker])
    wordcloud = WordCloud(
        width=800, height=400, background_color='white', max_font_size=110, max_words=100).\
        generate(words)
    plt.figure(figsize=(14, 7))
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.title('Words in '+ sentiment.capitalize() + ' tweets\n', fontsize=32)
    plt.axis('off')
    plt.show()
```

Te svako od nas ponaosob je formirao jedan od wordcloud-a za različite sentimente.

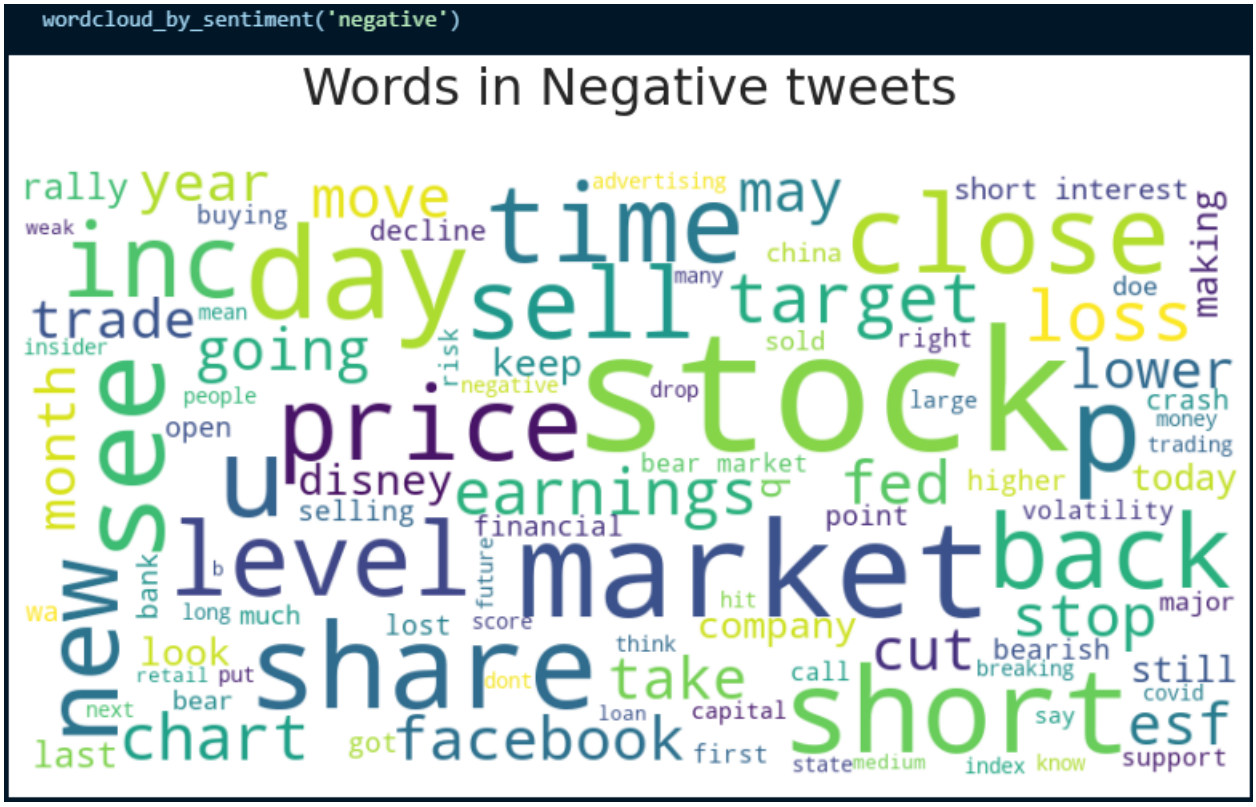
Ena:

```
wordcloud_by_sentiment('positive')
```



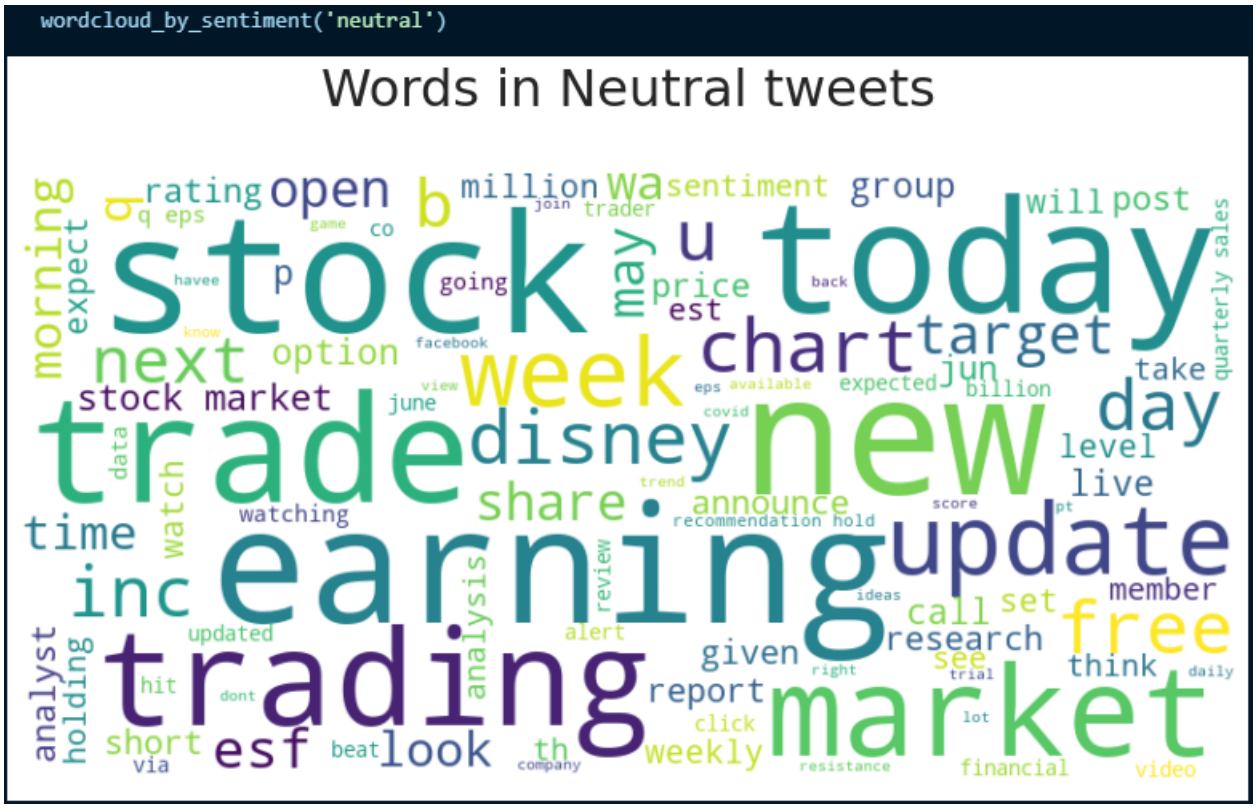
Midhat:

```
wordcloud_by_sentiment('negative')
```



Džemil:

```
wordcloud_by_sentiment('neutral')
```



Tabelarni prikaz algoritama sa naznakom ko je od članova tima zadužen za detaljnu analizu i implementaciju algoritma:

Student/metoda ML-a	Log. regression	Naive Bayes	SVM
Ena	**X**		
Džemil			**X**
Midhat		**X**	

Ena:

```
#Logistic regression
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression()
classifier.fit(X_train, y_train)
score = classifier.score(X_test, y_test)


print("Accuracy:", score)
```

```
Accuracy: 0.7776
```

Džemil:

```
#SVM
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
clf_svm = make_pipeline(StandardScaler(with_mean=False), SVC(gamma='auto'))
clf_svm.fit(X_train, y_train)
clf_svm.score(X_test, y_test)
```

```
0.8144
```

Midhat:

```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import CategoricalNB
from sklearn.naive_bayes import BernoulliNB

clf = BernoulliNB()
clf.fit(X_train, y_train)
score_clf = classifier.score(X_test, y_test)
print(score_clf)
```

```
0.7776
```

4) **Detaljna analiza i Implementacija algoritama. Svaki član tima treba detaljno analizirati jedan algoritam-metodu i implementirati ga na osnovu tabele iz 3. Analiza znači sažeti opis metode sa teorijskog aspekta sa adekvatnom referencom na detaljni opis, analiza prihvatljivosti te metode za problem koji se rješava, implementacija metode – opis koji jezik ili okruženje se koristilo. Potrebno je priložiti kodove.**

Kodovi su predloženi u prošlom odjeljku. Korišten je SKLEARN paket jer u sebi sadrži sve metode koje smo odabrali da implementiramo. Implementirane metode izvršavaju svoje treniranje na sličan način. Glavna razlika je u tome koji modul je korišten. Razliku između metoda smo odabrali objasniti u teoretskoj formi zato jer priloženi kodovi su izuzetno slični.

Ena:

Modelom logističke regresije opisujemo vezu između prediktora koji mogu biti neprekidni, binarni, kategorički, i kategoričke zavisne promjenljive. Na primjer zavisna promjenljiva može biti binarna - na osnovu nekih prediktora predviđamo da li će se nešto desiti ili ne. Ocjenjujemo zapravo vjerovatnoće pripadanja svakoj kategoriji za dat skup prediktora. U zavisnosti od tipa zavisne promjenljive imamo:

Binarnu logističku regresiju - zavisna promjenljiva je binarna. Na primjer: proći ili pasti test, odgovor da ili ne u anketi, visok ili nizak krvni pritisak

Nominalna logistička regresija - zavisna promjenljiva ima 3 ili više kategorija koje se ne mogu po vrednosti upoređivati. Na primjer: boje (crna, crvena, plava,…), internet pretraživač (Google, Bing, Yahoo!,…) ili u našem slučaju, pozitivan, neutralan i negativan sentiment.

Ordinalna logistička regresija - zavisna promjenljiva ima 3 ili više kategorija koje se prirodno mogu upoređivati, ali rangiranje ne mora da znači da su “rastojanja” između njih jednaka. Na primjer: ocjena (5-10), zdravstveno stanje (dobro, stabilno, ozbiljno, kritično) itd. Logistička regresija koristi kada zavisna promjenljiva uzima samo neki konačan skup vrijednosti. *Izvor: predavanja sa MU, RIM1*
Ovo je slučaj kod nas i također, poznato je da ovaj oblik regresije je više robustan metod za klasifikaciju podataka kao što su tekst, što čini ovu metodu prihvatljivom u našem slučaju. Algoritam koji je implementiran je iz sklearn python paketa i kod je:

```
#Logistic regression
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression()
classifier.fit(X_train, y_train)
score = classifier.score(X_test, y_test)


print("Accuracy:", score)
```

```
Accuracy: 0.7776
```

Odakle se vidi da se inicijalizira objekat klase LogisticRegression koji je naš klasifikator. Pripremljene podatke (opisane u preprocessing dijelu) prosljeđujemo metodi fit ove klase. Na kraju ispitujemo tačnost naše metode preko test skupa pozivom metode score. Ispisujemo konačnu tačnost koja je jednaka 77.76%.

Midhat:

Naive Bayes je probabilistički klasifikator. Ovo znači da glavni mehanizam rada je usko povezan sa računanjem vjerovatnoće i uslovne vjerovatnoće događaja. Konkretno u našem slučaju, vjerovatnoće određene klase u odnosu na ulazne podatke. Naivni Bayes klasifikator se zasniva na Naive Bayes teoremi koja podrazumijeva gotovo potpuno neovisnost podatka. Pokazalo se da iako postoji određena ovisnost u podacima pri sentimentalnoj analizi, zaključujemo da ta ovisnost i korelacija nije dovoljno velika da utiče na tačnost NBC-a u tolikoj mjeri da metoda ne bude iskorištena. Odabrali smo metodu ne samo zbog toga,

već i zbog velike brzine treniranja i inferiranja testnog skupa. Generalno metode koje smo koristili su mnogo brže od njihovih pandana iz opšte, AI klase algoritama čija brzina treniranja je neuporedivo manja od ovih metoda, što konkretno za predviđanje koje je trenutno potrebno klijentu, se neće moći tolerisati za iole više korisnika i više podataka, što je konačni cilj ovakvog sistema u komercijalnom smislu.

Po Bayes-ovoj teoremi, uslovna vjerovatnoća se računa po sljedećoj formuli:

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\, p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

Te se klasifikacija vrši prema sljedećoj formuli:

$$\hat{y} = \underset{k \in \{1,\ldots,K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Gdje je Ck klasa koju uzimamo i analiziramo. Ponaosob je potrebno analizirati i izračunati vjerovatnoću pripadnosti svakoj klasi i uzeti maksimum za rezultat. Pošto imamo samo tri klase, ovo je poprilično jednostavno uraditi i nisu potrebni veliki računarski resursi da bi se izvršila klasifikacija.

Potencijalni minus je što skup podataka postaje veći, to je veći proračun potreban kako bi se izvršila klasifikacija. Isto tako sa više podataka raste mogućnost da određeni podaci budu u jakoj korelaciji, koja smeta ovom klasifikatoru. Ovo su informacije koje je potrebno imati na umu pri implementaciji ovog klasifikatora. *Izvor: predavanja sa MU, RIM1*

Što se tiče implementacije, koristeći sklearn pakete za NaiveBayes smo testirali i odredili da BernoulliNB je najbolji klasifikator od tri ponuđena. Nakon inicijalizacije BernoulliNB klase bez parametara, prosljeđujemo trening podatke fit metodi ove klase. Nakon toga, koristeći score metodu zaključujemo da je tačnost 77.76%

```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.naive_bayes import CategoricalNB
from sklearn.naive_bayes import BernoulliNB

clf = BernoulliNB()
clf.fit(X_train, y_train)
score_clf = clf.score(X_test, y_test)
print(score_clf)
```

```
0.7776
```

Džemil:

SVM je namijenjen rješavanju binarnih klasifikacijskih problema ali također se može primjeniti i na multi-class klasifikacijski problem, što je nama potrebno u ovom slučaju. Konkretno, potrebna nam je troklasna klasifikacija za naše predstavljene klase (pozitivan, negativan i neutralan sentiment).

Poznato je da je SVM klasifikator nadogradnja MM koji zahtjeva potpuno razdvajanje klasa. Očigledno je da kod nas to nije slučaj jer određeni sentimenti su tranzientni kroz sve moguće klase (npr riječ stock). Očigledno je da je potrebno koristiti SVM klasifikator.

Pošto za SVM treniranje i testiranje ovisi isključivo od skalarnog proizvoda instanci koje se nalaze unutar i na margini, kao i skalarnog proizvoda instanci koje se nalaze na i unutar margine sa nepoznatim testnim skupom, ovaj metod je vrlo jednostavno implementirati sa obzirom na to koje metode su prethodno korištene (podaci su spremni, kao i rezultati za poređenje). *Izvor: predavanja sa MU, RIM1*

$$max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j \boldsymbol{x_i \cdot x_j}$$

$$f(x^*) = w \cdot u + b = \sum_{i,j=1}^{l} \alpha_i y_i \boldsymbol{x_i \cdot x^*} + b$$

Metod koji je korišten za implementaciju je iz paketa sklearn, konkretno SVC klasa svm modula unutar ovog paketa. Pošto je za SVM potrebno odrediti skaliranje podataka, napravljen je pipeline koristeći pipeline modul sklearn paketa u pythonu. Nakon formiranja pipeline-a, slično kao prethodne dvije metode, isti trenirng skup je dodijeljen fit metodi klasifikatora SVM-a sa default podacima koji su se pokazali dovoljnim, s obzirom na to da SVM klasifikator posjeduje optimizatore hiperparametara u tom slučaju koji će odabrati između ostalog i optimalnu funkciju kernela.

```
#SVM
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
clf_svm = make_pipeline(StandardScaler(with_mean=False), SVC(gamma='auto'))
clf_svm.fit(X_train, y_train)
clf_svm.score(X_test, y_test)

0.8144
```

Istrenirani model se evaluira preko testnog skupa i vidi se da je tačnost modela 81.44% što je i najveća tačnost dosad. Model izuzetno brzo dostavlja rezultat i brzo trenira, što ga čini mnogo više iskoristivim od možda RNN mreža za čije treniranje je potrebno mnogo vremena, a i vrijeme inferencije istih nije zavidno u odnosu na metode ML-a.

5) **Analiza dobijenih rezultata na podacima iz 2 upotrebom raznih metrika evaluacije. Analiza primjenjivosti sistema.**

Rezultati koji su dobijeni su prihvatljivi s obzirom na to koje metode su iskorištene. Najvažnija metrika za evaluaciju je metrika tačnosti.

Metrika tačnosti ukazuje na to da se tačnost naših metoda kreće od 77-81%, što autori navode kao maksimalnu tačnost koju su i oni postigli.

Još jedna metrika koja je razmatrana je neovisnost klasa. Pri sentimentalnoj analizi je primijećeno da određene riječi su koliko-toliko isključivo vezane za određene klase.

Pri testiranju, ako su intenzivno korištene ove riječi, došlo je do pada tačnosti. Zaključujemo da ovisnost klasa o određenim riječima je umjereno visoka i to je nešto što se treba razmotriti ti povećavanju skupa podataka za treniranje ili testiranje.

Metrike za True Positive i True Negative su jedine koje imaju smisla biti iskorištene u ovom slučaju. False Negative i False Positive imaju smisla za dvoklasnu klasifikaciju, tako da se u ovom slučaju odbacuje.