Truc Nguyen
11033379

CSCE 5290: Natural Language Processing
Project Increment 2

Project Proposal

This project is my personal attempt to predict the stock markets. It is not easy to predict the stock markets with a high degree of accuracy because there are so many factors affecting the prediction. Fundamentals of the market, human behaviors, and physical and psychological factors are among factors which can add noise to the model. That being said, correctly identify the market movement will result in lucrative rewards.

This project will try to predict stock price/movement of a group of multiple stocks using sentiment analysis of social media.

Project Proposal Description:

I. Project Title and Team Members:

Project Title: Predicting Stock Markets with the help of sentiment analysis.
Team Members: I, Truc Nguyen, will be a sole member of the project
Source code and all related documentations will be uploaded to:
https://github.com/trucntx0550/NLP

II. Goals and Objectives:

• Motivation: The popularity of social media and investment platforms has attracted more and more retail investors to participate in the stock market. It has, for the time being, changed the way the markets behave. Investing is no longer just the game of WallStreet and Hedge fund's managers. Retail investors, with the enabling of social platforms, can gather together to manipulate a company's stock. The recent example of WallStreetBets has proved the influence of social media to the market movement. Correctly analyze the posts/tweets can lead to an improvement in predicting the market.

• Significance: This project aims to expand the sentiment analysis dictionary for the financial sector introduced by D. Shah et al. (2018).

• Objectives: Find the correlation between the sentiment on social media platforms and a particular stock movement. This can be later expanded to a group of stocks.

• Features:

- Stock price will be pulled from Yahoo Finance for stock price prediction
- Data from Twitter, Stockwits, Reddit, etc. will be used for sentiment analysis.

Truc Nguyen
11033379

III. Background

1. Related Work

This project is inspired by Khedr and Yaseen (2017). They introduced a model to predict future trends of the stock market. This model successfully achieved a small error ratio. They used KNN and naïve Bayes algorithm to improve the accuracy of prediction.
Das et al. (2018) work study the correlation between stock market and twitter data. Their classifying model takes in historical data to improve the accuracy of the predictions with the assistance of Twitter data.
The impact of COVID-19 has fundamentally changed the way investors view the market. Lee (2020) uses big data collected from Google Trends data as well as Daily News Sentiment Index to explore the impact of COVID19 sentiment on the market movements. This study investigated the correlation between 11 indices and investors' sentiment during COVID. They have found a significant connection between the sentiment and different industries and classified them into correlation groups.

2. Model

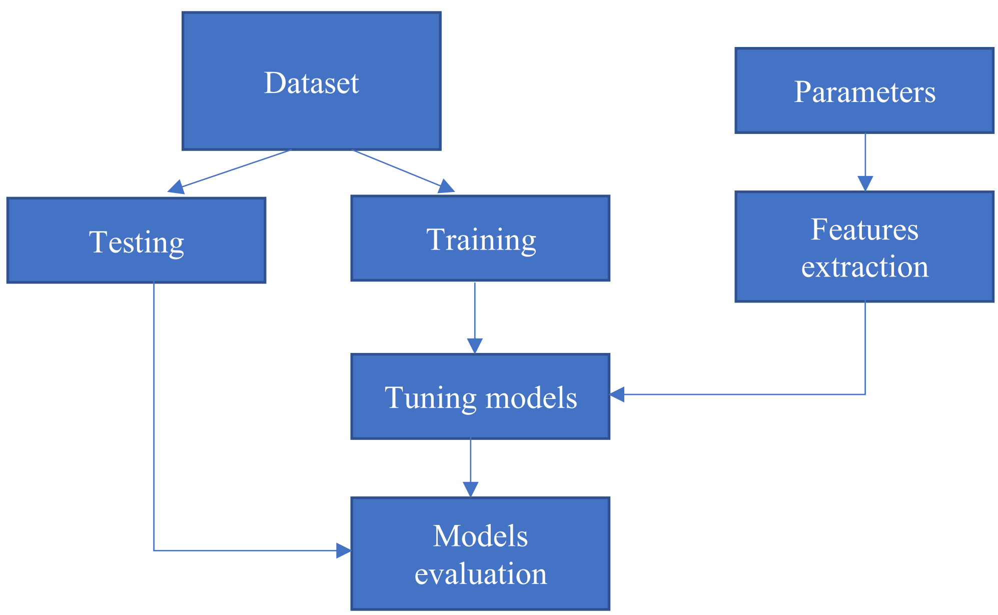


Figure 1 - Architecture Diagram

Truc Nguyen
11033379

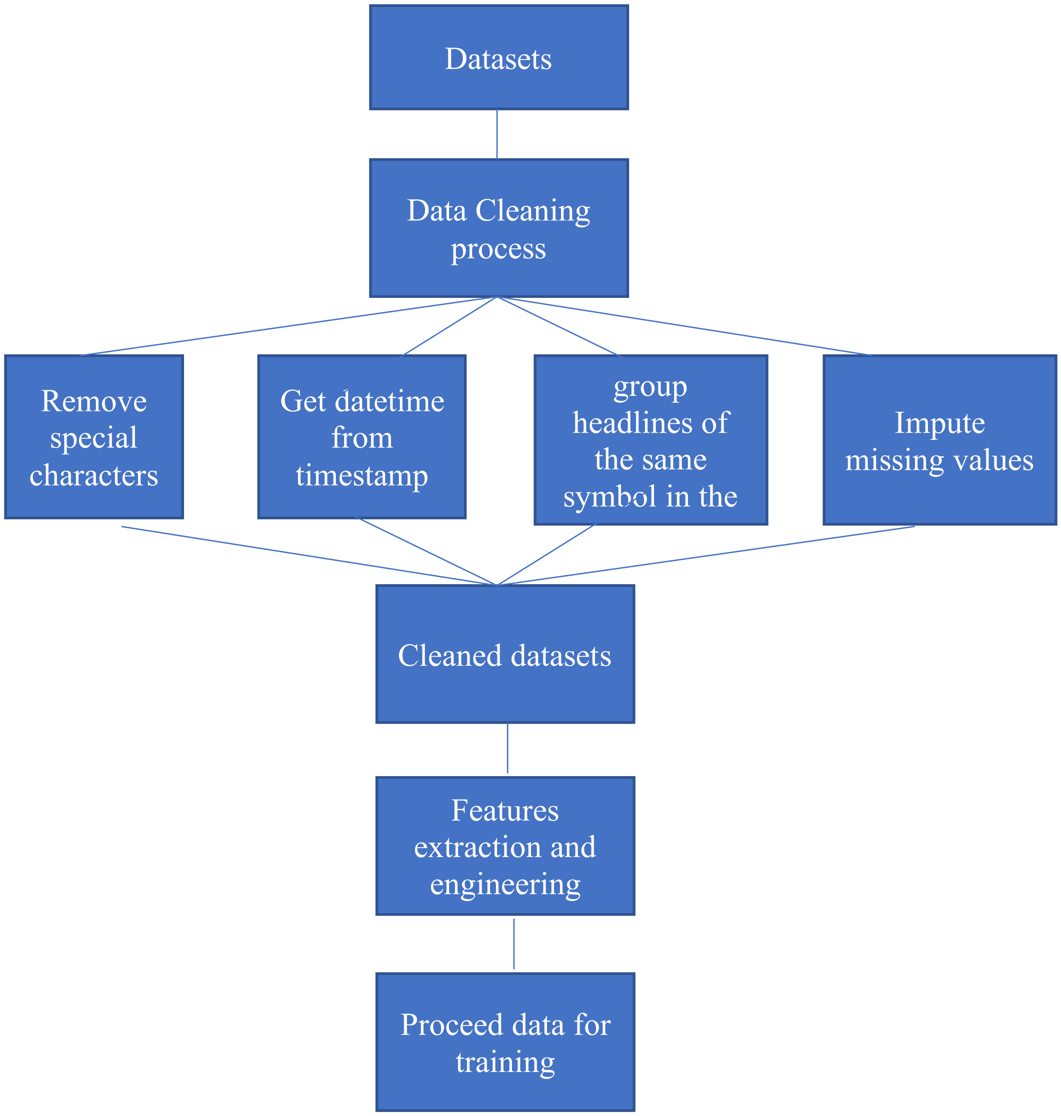


Figure 1 - Workflow

Truc Nguyen
11033379

3. Datasets:

I used two datasets for this project:

- Headline news dataset was obtained from Kaggle. It combined stock news from 2008 to 2016.
- The second dataset contains stock prices of companies in the first dataset during the same period. This dataset is pulled from Yahoo Finance.

Datasets description:

- Headline news dataset - `stockerbot-export.csv`
  - 28264 rows x 8 columns:
    - `id`
    - `text`- headlines news of stocks
    - `timestamp` – datetime of the news
    - `source` – website/source of the headlines
    - `symbols` – stocks symbol extracted from the news
    - `company_names` – company names based on stock symbols
    - `url` – link to the headline
    - `verified` – Boolean - if the source is verified

[4]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28264 entries, 0 to 28263
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             28264 non-null  int64
 1   text           28264 non-null  object
 2   timestamp      28264 non-null  object
 3   source         28264 non-null  object
 4   symbols        28264 non-null  object
 5   company_names  28263 non-null  object
 6   url            21895 non-null  object
 7   verified       28264 non-null  bool
dtypes: bool(1), int64(1), object(6)
memory usage: 1.5+ MB
b'Skipping line 731: expected 8 fields, saw 13\nSkipping line 2836: expected 8 fields, saw 15\nSkipping line
```

```
df.sample(10).head(5)
```

text	timestamp	source	symbols	company_names	url	verified
tocks on watch $uuuu $UUU $UEC $URG $WWR $CCJ...	Wed Jul 18 12:56:46 +0000 2018	SwingingForward	LLY	Eli Lilly and Company	NaN	False
$0.45 EPS Expected for Taiwan Semiconductor Ma...	Wed Jul 18 13:11:31 +0000 2018	whatsonthorold2	TSM	Taiwan Semiconductor Manufacturing Company Lim...	https://www.whatsonthorold.com/2018/07/18/0-45...	False
Analyst Peter Keith discusses Best Buy come-ba...	Mon Jul 16 14:51:00 +0000 2018	PiperJaffrayCo	BBY	Best Buy Co.	https://www.mprnews.org/story/2018/07/12/best-...	False
M&T Bank MTB Releases Quarterly	Wed Jul 18 13:26:18 +0000 2018	TheMarketsDaily	MTB	M&T Bank Corporation	http://zpr.io/6XcRj	False

-
-

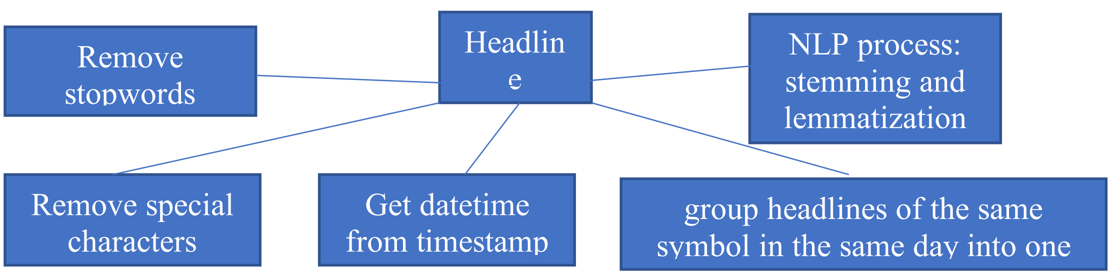
Truc Nguyen
11033379

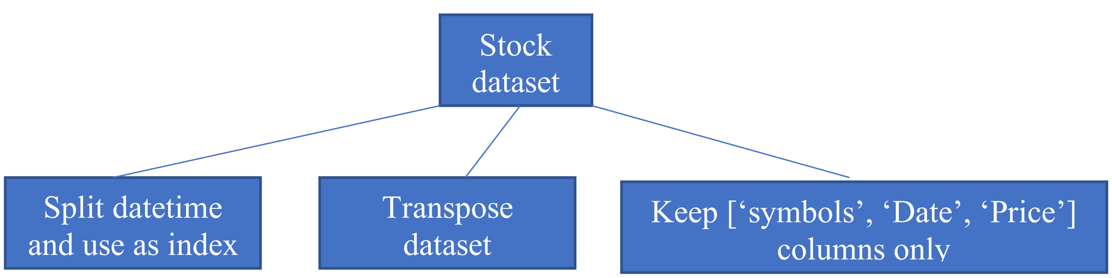
- Stock Prices dataset - `stocks_cleaned.csv`
  - 4152 rows × 3 columns:
    - `symbols`
    - `Date`
    - `Price`

	symbols	Date	Price
0	A	2018-07-09	61.5467
519	A	2018-07-10	62.083
1038	A	2018-07-11	61.2931
1557	A	2018-07-12	61.8684
2076	A	2018-07-13	61.8002
...	...	...	...
2074	ZTS	2018-07-12	83.7446
2593	ZTS	2018-07-13	84.4011
3112	ZTS	2018-07-16	82.8726
3631	ZTS	2018-07-17	84.0385
4150	ZTS	2018-07-18	84.2149

4152 rows × 3 columns

Detail design of Features with diagram





4. Detail design of Features

   I have decided to keep all the features in the original dataset since the headlines may have equally effectiveness on the target value.

Truc Nguyen
11033379

I used TimeSeriesSplit from sklearn.model_selection to split the combined dataset into training and testing using with `test_size=0.2, random_state=50`

Stop words and special characters are remove from the dataset using regular expressions

```
# Removing special characters

import re
import copy

#df_cleaned = pd.DataFrame(columns=['timestamp','text'])
df_cleaned = df[['timestamp','text', 'symbols']]

spec_cha = "(@\[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)|^rt|http.+?"
#spec_cha = '[^A-Za-z0-9]+'


df_cleaned['text'] = df_cleaned['text'].replace(to_replace=spec_cha, regex=True
df_cleaned['text'].reset_index(drop=True)
#df_cleaned = [df_cleaned['text'].replace(spec_cha, ' ')]
```

```
[ ] # Remove Stop words
    freq = pd.Series(' '.join(df_cleaned['text']).lower().split()).value_counts()[:20]
    freq

    stop_words = set(stopwords.words("english"))
    stop_words = stop_words.union(freq.index.tolist())
    extra_words = ['amp', 'rt']
    stop_words = stop_words.union(extra_words)
```

Processing language to keep English headlines only

Truc Nguyen
11033379

```
# Processing language

from langdetect import detect_langs

#check for valid string only to detect languages
TextValid=[]

for i in range(len(df_cleaned)):
    TextValid.append(bool(re.match('^(?=.*[a-zA-Z])', df_cleaned.iloc[i,0])))

df_cleaned['valid'] = TextValid
#print(len(df_cleaned[df_cleaned['valid']==False]))
#print(len(df_cleaned[df_cleaned['valid']==True]))

# Detect languages for each text
languages = []

# Loop over the sentences in the data and detect their language
for row in range(len(df_cleaned)):
    languages.append(detect_langs(df_cleaned.iloc[row, 0]))

languages = [str(lang).split(':')[0][1:] for lang in languages]

# Assign the list to a new feature
df_cleaned['language'] = languages

# count the languages in the data
df_cleaned['language'].value_counts()

# keep EN Only
df_cleaned = df_cleaned[df_cleaned['language']=='en']
```

NLP processing

Truc Nguyen
11033379

```
[ ] corpus = []
    for i in df_cleaned.index:
        #Remove punctuations
        text = re.sub('[^a-zA-Z]', ' ', df_cleaned['text'][i])

        #Convert to lowercase
        text = text.lower()

        #remove tags
        text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)

        # remove special characters and digits
        text = re.sub("(\\d|\\W)+"," ",text)
        text = text.replace("\n","")

        ##Convert to list from string
        text = text.split()

        ##Stemming
        ps=PorterStemmer()    #Lemmatisation
        lem = WordNetLemmatizer()
        text = [lem.lemmatize(word) for word in text if not word in
                stop_words]
        df_cleaned['keywords'] = pd.Series(text)
        text = " ".join(text)
        corpus.append(text)

    pd.Series(corpus).sample(20).head(20)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:24: DeprecationWarning: The default
2435     join binance fastest growing exchange world re...
7638                                           ibkr etfc
19484                                                tel
10146     analyst adopt bullish outlook helmerich payne hp
2279     blue apron aprn v netshoes cayman net financia...
5789     get access premium paid group fraction price j...
15235    energicrypto energi earndrop live early partic...
```

Get the datetime from timestamp then sort by `'symbols', 'datetime'`

```
# get datetime from timestamp
df_cleaned['datetime'] = pd.to_datetime(df['timestamp']).apply(lambda x: x.date())

# sort by 'symbols', 'datetime'
df_cleaned = df_cleaned.sort_values(['symbols', 'datetime'])
```

Headlines of a same date are combined into a paragraph to create BagofWords to use for sentiment extraction

Truc Nguyen
11033379

```
[ ] # grouping text that have the same symbol to one row
    # then group by date
    indx=0
    get_tweet=""
    for i in range(0,len(df_cleaned)-1):
        get_date = df_cleaned['datetime'].iloc[i]
        next_date = df_cleaned['datetime'].iloc[i+1]

        get_symbols = df_cleaned['symbols'].iloc[i]
        next_symbols = df_cleaned['symbols'].iloc[i+1]

        if(str(get_symbols) == str(next_symbols)):
          if(str(get_date) != str(next_date)):
            get_tweet = df_cleaned['text'].iloc[i]

            temp_df = pd.DataFrame([[get_date, get_tweet, get_symbols]]
                                   , columns = ['Date','text','symbols'])
            df_cleaned1 = pd.concat([df_cleaned1, temp_df], axis = 0).reset_index(drop = True)

            get_tweet=" "
          else:
            get_tweet = get_tweet + df_cleaned['text'].iloc[i]+" "
        else:
          #if (str(get_date) != str(next_date)):
          temp_df = pd.DataFrame([[get_date, get_tweet, get_symbols]]
                                 , columns = ['Date','text','symbols'])
          df_cleaned1 = pd.concat([df_cleaned1, temp_df], axis = 0).reset_index(drop = True)
          get_tweet=" "
```

```
[ ] df_cleaned1
```

	datetime	text	symbols	Date
0	NaN	a pa ion du football tait d j connue	A	2018-07-16
1	NaN	A repeat of 2002 Walmart may be looking to...	A	2018-07-17
2	NaN	ACE OUT A CGI HE PAY ALL DATA YELP ...	A	2018-07-18

Get stock prices from Yahoo Finance

```
                    A        AAL       AAOI  ...       ZION  ZNGA        ZTS
Date                                         ...
2018-07-09  61.546658  38.477581  45.330002  ...  48.723942  4.24  85.723824
2018-07-10  62.082951  38.291603  45.790001  ...  48.263161  4.19  84.479469
2018-07-11  61.293144  35.198517  45.759998  ...  48.001163  4.26  82.794197
2018-07-12  61.868443  35.560684  48.799999  ...  47.477150  4.40  83.744614
2018-07-13  61.800186  36.333954  47.779999  ...  46.763412  4.34  84.401062
```

Combine the datasets

Truc Nguyen
11033379

```
[ ] # append 'Price' from Stock Dataset to News Dataset
    for i in range (0,len(df_cleaned1)):
        for j in range (0,len(data_Tr)):
            get_tweet_date = df_cleaned1['Date'].iloc[i]
            get_stock_date = (data_Tr['Date'].iloc[j]).date() # get rid of 00:00:00

            get_tweet_symbol = df_cleaned1['symbols'].iloc[i]
            get_stock_symbol = data_Tr['symbols'].iloc[j]



            if(str(get_tweet_symbol) == str(get_stock_symbol) and
               (str(get_stock_date) == str(get_tweet_date))):
                #print(get_stock_date," ",get_tweet_date)
              #if(str(get_stock_date) == str(get_tweet_date)):

                # ccdata.set_value(i,'Prices',int(read_stock_p.Close[j]))
                df_cleaned1['Price'].iloc[i] = int(data_Tr['Price'][j])
```

```
# fill missing 'Price' with the most recent price
for i in range(len(df_cleaned1)):
  if df_cleaned1['Price'].iloc[i] == '':
    df_cleaned1['Price'].iloc[i] = df_cleaned1['Price'].iloc[i-1]
```

5. Implementation
   First step is the Sentiment analysis using `TextBlob` to classify the headlines to three categories `Positive,` `Negative,` and `Neutral.`

```
[ ] # Percentage of each Emotions overall symbols

    df_neutral   = df_cleaned['text'][df_cleaned['Emotion'] == '0']
    df_positive  = df_cleaned['text'][df_cleaned['Emotion'] == '1']
    df_negative  = df_cleaned['text'][df_cleaned['Emotion'] == '2']
```

```
[ ] print(f'Percentage Positive: {len(df_positive)/len(df_cleaned)}')
    print(f'Percentage Negetive: {len(df_negative)/len(df_cleaned)}')
    print(f'Percentage Neutral: {len(df_neutral)/len(df_cleaned)}')

    Percentage Positive: 0.2759310283530353
    Percentage Negetive: 0.12278908541462981
    Percentage Neutral: 0.6012798862323349
```

Truc Nguyen
11033379

```
[ ] # convert 'Price' to integer
    combined_data['Price'] = combined_data['Price'].apply(np.int64)

    # adding columns for sentiment analysis
    combined_data['Emotion'] = ''
    combined_data['Negative'] = ''
    combined_data['Neutral'] = ''
    combined_data['Positive'] = ''
```

```
[ ] # Sentiment Analysis with vader
    import nltk
    nltk.download('vader_lexicon')

    [nltk_data] Downloading package vader_lexicon to /root/nltk_data...
    True
```

```
[ ] from nltk.sentiment.vader import SentimentIntensityAnalyzer
    from nltk.sentiment.vader import SentimentIntensityAnalyzer
    import unicodedata
    sentiment_i_a = SentimentIntensityAnalyzer()
    for indexx, row in combined_data.T.iteritems():
        try:
            sentence_i = unicodedata.normalize('NFKD', combined_data.loc[indexx, 'text'])
            sentence_sentiment = sentiment_i_a.polarity_scores(sentence_i)
            combined_data['Emotion'].iloc[indexx] = sentence_sentiment['compound']
            combined_data['Negative'].iloc[indexx] = sentence_sentiment['neg']
            combined_data['Neutral'].iloc[indexx] = sentence_sentiment['neu']
            combined_data['Positive'].iloc[indexx] = sentence_sentiment['compound']

        except TypeError:
            print (stocks_dataf.loc[indexx, 'text'])
            print (indexx)
```

```
[24] from sklearn.feature_extraction.text import CountVectorizer
     from sklearn.ensemble import RandomForestClassifier
     from sklearn.metrics import classification_report,confusion_matrix,accuracy_score

     ## implement BAG OF WORDS
     countvector=CountVectorizer(ngram_range=(2,2))
     traindataset=countvector.fit_transform(headlines)
```

Different models are used to detect emotion in the headline: The `LogisticRegression`, `SupportVectorClassifier`, `DecisionTree`, `KNeighborsClassifier`

These models are then tuning using below methods:

- `NgramModels` with n = 2 and n = 3
- `NgramModels` with `k in [1,3,5,7,10]`
- `TFIDFModels` with `min_df = 5, max_df =0.8`
- `KNN_TFIDF` with `min_df = 5, max_df =0.8 and k in [1,3,5,7,10]`

Truc Nguyen
11033379

```
## implement RandomForest Classifier
randomclassifier=RandomForestClassifier(n_estimators=200,criterion='entropy')
randomclassifier.fit(traindataset,train['Label'])
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=200,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
## Predict for the Test Dataset
test_transform= []
for row in range(0,len(test.index)):
    test_transform.append(' '.join(str(x) for x in test.iloc[row,2:27]))
test_dataset = countvector.transform(test_transform)
predictions = randomclassifier.predict(test_dataset)
```

6. Results

```
matrix = confusion_matrix(test['Label'],predictions)
print(matrix)
score = accuracy_score(test['Label'],predictions)
print(score)
report = classification_report(test['Label'],predictions)
print(report)
```

```
[[139  47]
 [  8 184]]
0.8544973544973545
              precision    recall  f1-score   support

           0       0.95      0.75      0.83       186
           1       0.80      0.96      0.87       192

    accuracy                           0.85       378
   macro avg       0.87      0.85      0.85       378
weighted avg       0.87      0.85      0.85       378
```

The RandomForest model accuracy was 85% in predicting if stock will increase or decrease based on a headline.

The `LogisticRegression`, `SupportVectorClassifier`, `DecisionTree`, `KNeighborsClassifier` also achieve high accuracy with TFIDF to be the most accuracy. Of all the models attempted, `SupportVectorClassifier` and `DecisionTree` are the best models

Truc Nguyen
11033379

FeatureExtraction	Metric	LogisticRegression	SupportVectorClassifier	DecisionTree	KNeighborsClassifier
2-grams	Accuracy Training %	79.17	78.91	82.66	80.77
	Accuracy Testing %	76.92	76.94	77.32	74.24
3-grams	Accuracy Training %	73.60	73.65	74.12	73.08
	Accuracy Testing %	71.99	72.15	72.15	70.69
TFIDF	Accuracy Training %	95.73	97.83	1.0	1.0
	Accuracy Testing %	93.35	95.99	95.96	82.46

7. Project Management
   Implementation status report:
   - Work completed:
     - Cleaned the dataset
     - Feature engineering to select appropriate features for the model
     - Implemented RandomForest Classification to predict whether the index will increase (1) or decrease (0) based on headlines
     - `LogisticRegression`, `SupportVectorClassifier`, `DecisionTree`, `KNeighborsClassifier` also used to detect emotion in the headline with high accuracy
   - Issues/Concerns:
     - Attempted to predict stock prices based on headlines but due to time constraint, I wasn't able to get the desirable outcomes.

Truc Nguyen
11033379

References

https://github.com/you915/Sentiment-Analysis-of-Twitter-Data-for-predicting-Apple-stock-price

Pagolu, Challa, Panda, Majhi, Sentiment Analysis of Twitter Data for Predicting Stock Market Movements. International conference on Signal Processing, Communication, Power and Embedded System (SCOPES)-2016

A. Pak and P. Paroubek, Twitter as a corpus for sentiment analysis and opinion mining, in Proceedings of the Seventh International Conference on Language Resources and Evaluation, 2010, pp. 13201326

D. Shah, H. Isah and F. Zulkernine, "Predicting the Effects of News Sentiments on the Stock Market," 2018 IEEE International Conference on Big Data (Big Data), 2018, pp. 4705-4708, doi: 10.1109/BigData.2018.8621884.

László Nemes & Attila Kiss (2021) Prediction of stock values changes using sentiment analysis of stock news headlines, Journal of Information and Telecommunication, 5:3, 375-394, DOI: 10.1080/24751839.2021.1874252

Khedr, A. E., & Yaseen, N. (2017). Predicting stock market behavior using data mining technique and news sentiment analysis. International Journal of Intelligent Systems and Applications, 9(7), 22. https://doi.org/10.5815/ijisa

Das, S., Behera, R. K., & Rath, S. K. (2018). Real-time sentiment analysis of twitter streaming data for stock prediction. Procedia Computer Science, 132, 956–964. https://doi.org/10.1016/j.procs.2018.05.111

Lee, H. S. (2020). Exploring the initial impact of COVID-19 sentiment on US stock market using big data. Sustainability, 12(16), 6648. https://doi.org/10.3390/su12166648