Section 1. Introduction

1.1 Domain Specific Area

Sentimental impact on Financial Markets

Financial markets were traditionally analyzed on the financial results of companies but there is a recent change in trend due to technological advancement over the last decade, which let the general public invest in stocks conveniently and efficiently for day trading. However, This ease of investing and involvement of public raises a question that "Do financial markets and stock investment reflects the rational behavior of investors ".

According to the theory laid by Kahneman & Tversky, There is evidence that the investment decisions in digitalized world are more often guided by personal perceptions derived from social media.

VIX is a fear index that measures the 30-day expected volatility of the US stock market. It is used as an indicator for investor to judge the sign of risks for stock investments. Correlation between VIX and movement in S&P 500 is depicted in charts below:

VIX trend for the month of May is provided above for comparison purpose, we can notice that the S&P 500 went down from 4 May onwards and at the same time VIX percentage was highest. This confirms the relationship between rational behavior of investors and investment decisions made based on sentiments.

Text classification techniques to analyze the rational behavior of investors

Since, we have real-time information available in huge volume from social media platforms such as Twitter, Facebook, Reddit etc. Therefore, we can use NLP - Natural Language Processing for sentimental analysis of comments/tweets to predict positive and negative impact on stocks prices.

Machine learning could help companies and individuals dealing in stock investment, and some of these are as follows:

- Quickly analyze hundreds of tweets/comments on social media using NLP sentimental analysis libraries such as NLTK, Spacy etc.
- Judge the trends for future investments using text classification and machine learning.
- Companies can have targeted advertising specifically for the users commenting and interested in stocks e.g. Apple, Google etc.
- Q & A system could be established based on machine learning to answers public queries on stocks.

It is roughly estimated to have 4.41 Billion worldwide social media users by 2025. Since, This project is related to NLP, therefore, text classification models will be build to perform sentimental analysis of stock market related tweets to judge their impact (positive or negative) on stock prices and financial market.

Resource Reference:

1. *https://corporatefinanceinstitute.com/resources/knowledge/trading-investing/market-sentiment/*
2. *https://finance.yahoo.com/quote/%5EVIX/*
3. *https://www.statista.com/statistics*

~ 394 word count

1.2 Objective

Background

Emotional analysis is a way of measuring the different emotional aspects of human expression in their speech, comments or body language. Emotional analysis from text is related to the field of sentimental analysis. The aim of sentimental analysis being subset of NLP is to judge the positivity, negativity, or neutrality from the text [1]. The use of computers and AI improvements also got the attention of Financial sector to help them make right and timely decisions based on market sentiments. GramStop is the recent example in which share price was almost up by 100% based on the speculative trend started by Reddit community [2]. In another example, Elon Musk being entrepreneurs have almost 96 M follower on Twitter, therefore, his small comment on something could have wide range impact. This was also evident in cash of GameStop stock, in which his tweet "Gamestonk!!" resulted in price hike of 31% from $144 to $348 next day [3].

Objective and methodology

There are almost 3,300 companies listed on NASDAQ and all susceptible towards rational behaviour of public. It is impossible to have a team of people analysing the public tweets or comments posted on social media. Therefore, the objective of the project is to get Twitter dataset for stock and build a text classifier to get the sentiment (Positive, Negative, or Neutral) analysed based on machine learning. This will help financial sector and more specifically for investors to make well-informed and timely decision for the stock investment. The steps to be followed are as follows:

- **Step 1 — Dataset Download:** Dataset will be taken from Kaggle or any other source. There is another option to download the tweets directly from twitter using data extraction tools such as selenium.
- **Step 2 — Data Cleaning:** News/Messages or tweets are usually presented in an unstructured formate, therefore, preprocessing of the data will be performed using regex, tokenization, removing stop words, removing punctuations, stemming and lemmatization.
- **Step 3 — Data Analysis and Feature presentation:** Preprocessed data will be analysed using simple statistical tool using term frequency and Word cloud will also be used to represent the words used most in the tweets posted. Moreover, features presented in the dataset will be extracted and presented via stacked chart to get the overall feel of the datasets.
- **Step 4 — Feature Extraction:** Feature extraction technique will be finalized at this stage and dataset will get divided into training and test subsets. These subsets are required for training and testing of models
- **Step 5 — Training Model:** I have selected Logistic Regression as text classifier to get the baseline performance, and it will be compared against the Naive Bayes classifier under classification approach.
- **Step 6 — Evaluating Models:** The performance of the models will be evaluated using Precision, Recall, Accuracy, F-Measure, Confusion Matrix etc. This could either be conducted while training the model or at later stage of the project/

Resource Reference:

1. *https://devblogs.microsoft.com/cse/2015/11/29/emotion-detection-and-recognition-from-text-using-deep-learning/*
2. *https://www.forexlive.com/news/!/speculation-gone-wild-or-a-new-era-in-the-stock-market-20210127*
3. *https://www.wsj.com/articles/how-the-stock-market-works-now-elon-musk-tweets-gamestop-millions-buy-11613147654*

~ 478 word count

1.3 Datasets

Most challenging part was to get the dataset from online for twitter posts or comments which are specifically related to financial markets or stocks. The dataset available online are either small in size or not annotated for sentiments [negative, positive or neutral]. Therefore, I have selected dataset from 4 different sources to make one combined dataset for model training and testing purposes.

Datasets Description

Dataset	Source	Data Label	Description
Dataset-1	Kaggle	Sentiment Analysis lexicon	Dataset related to stocks with 9,917 record. However, 1,300 records are manually classified and reviewed by an independent reviewer for sentiments accuracy.
Dataset-2	Kaggle	Stock data EDA and Prediction	Dataset related to stock with 5,791 record which are annotated between negative and positive sentiments.
Dataset-3	Kaggle	Huge crash in the Stock market 2022	Dataset for financial market crash and stocks and with 33,946 records annotated between negative and positive sentiments.
Dataset-4	Kaggle	Sentiment140 dataset with 1.6 million tweets	It is a general Twitter dataset with 1.6 million tweets annotated for positive and negative sentiments.

Since, there is no single source available for financial or stock tweets related dataset. Therefore, dataset 1-3 *(Stock related datasets)* is combined with a general dataset having 1.6 millions tweets. This approach will fine tune the model for better text classification and will let the model accurately classify tweets text even if it is related to financial market or stocks.

Data types

Dataset	Data Types
Dataset-1	- ID -> Contains id for posted tweet - Date and time -> Date and time of the tweet. - Tweet -> Tweet/text written by the user. - Sentiment -> Tweet is classified as positive or negative.
Dataset-2	- Text-> Tweet/text written and posted by the user. - Sentiment -> Tweet annotated for being positive or negative.
Dataset-3	- id-> id of the posted tweet. - text-> Tweet/text written by the user. - username -> Tweet name for user. - hashtags -> hashtags used for the post. - created_at -> date on which tweet posted. - user follower count -> Count of the followers for users posted tweet. - replycount-> Count for the reply on tweet. - retweetcount-> Count for tweet retweeted. - likescount-> Count for likes on the tweet
Dataset-4	- target> Tweet annotated for Positive, Negative, or Neutral. - id-> id of the posted tweet. - date -> date and time for tweet posted. - query -> Flag for the tweet. - user -> user for the tweet. - text -> text of the tweet.

Dataset size and links

Dataset	Data Size	Online Link
Dataset-1	175 MB	https://www.kaggle.com/datasets/utkarshxy/stock-markettweets-lexicon-data
Dataset-2	479.97 KB	https://www.kaggle.com/code/adeyoyintemidayo/stock-data-eda-and-prediction/data
Dataset-3	14.22 MiB	https://www.kaggle.com/datasets/tejasurya/huge-stock-market-crash-2022
Dataset-4	238.8 MB	https://www.kaggle.com/datasets/kazanova/sentiment140

~ 468 word count

1.4 Evaluation Methodology

There are different metrics to judge the performers of text classifiers Such as accuracy , precision, and recall metrics. Using this mattress will let us improve our model production before we rollout such models for production.

Relying just on the accuracy and without evaluating the model with the help of evaluation metrics could lead to unseen problems and end up in poor prediction implementation.

Accuracy, confusion metrics and ROC metrics are some popular metrics widely used to evaluate classification problems. Therefore, I will these matrices to evaluate the performance of models in this project along with precision, recall and F1 measure.

Accuracy measures how correctly the classifier is predicting the sentiment based on inputs. Accuracy is the correct matrix when the target classes are well-balanced, but it is not a good choice for the unbalanced classes, therefore in order to get the full picture we should consider other mattresses such as recall and precision.

Accuracy = (TP + TN) / (TP+TN+FP+FN)

Confusion matrix is a performance measurement tool for classification problems where there are two or more classes it is shown in the table format with a combination of predicted and actual values.

> Confusion matrix a table representation to check the performance of model in matrix form | | Predicted NO | Predicted YES | ----------| ----------- | ----------- | | Actual NO |True Negative (TN) | False Positive (FP)| | Actual YES |False Negative (FN) |True Positive (TP)|

True Negative (TN), True Positive(TP), False Negative(FN) and False Positive (FP) for twitter sentiments are further explained below:

- **True Positive:** The number of tweets correctly identified by model as Positive, which is in fact positive
- **True Negative:** The number of tweets correctly identified by model as Negative, which is in fact Negative
- **False Positive:** Number of tweets selected as having sentiment positive, but in fact these have negative sentiment
- **False Negative:** Number of tweets identified as negative, but in fact these are having positive sentiments

Precision explains how many correct predictions made by the model turn out to be positive. It is important for cases where false positives are of more concern than false negatives. Especially in our case where wrong sentiment could lead to wrong investment decisions.

Recall explains that how many positive cases are predicted by the model correctly. This measure is significant where false negatives are of more concern than false positives. E.g. in medical cases where raising false alarm is not a bing concern because actual cases should not be gone as undetected.

F-1 Measure merges both precision and recall, and it archives the maximum threshold when recall equals precision.

UC-ROC—The Receiver Operator Characteristic (ROC) plots the True positive rate against the False positive rates in a curve graph. Areas under the curve measure the capability of the classifier between positive and negative classes.

Resource Reference:

1. *https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62*
2. *https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/*

~ 460 word count

Section 2. Implementation

2.1 Preprocessing

In [2]:
```
#Installing required packages

!pip3 install numpy
!pip3 install pandas
!pip3 install regex
!pip3 install nltk
!pip3 install plotly
!pip3 install nbformat
!pip3 install matplotlib
!pip3 install scipy
!pip3 install sklearn
!pip3 install textblob
!pip3 install wordcloud
!pip3 install seaborn
!pip3 install nbconvert
```

```
Requirement already satisfied: numpy in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (1.21.5)
Requirement already satisfied: pandas in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (1.4.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from pandas) (2021.3)
Requirement already satisfied: numpy>=1.18.5 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from pandas) (1.21.5)
Requirement already satisfied: six>=1.5 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Requirement already satisfied: regex in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (2022.3.15)
Requirement already satisfied: nltk in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (3.7)
Requirement already satisfied: regex>=2021.8.3 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nltk) (2022.3.15)
Requirement already satisfied: click in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nltk) (8.0.4)
Requirement already satisfied: tqdm in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nltk) (4.64.0)
Requirement already satisfied: joblib in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nltk) (1.1.0)
Requirement already satisfied: plotly in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (5.6.0)
Requirement already satisfied: six in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from plotly) (1.16.0)
Requirement already satisfied: tenacity>=6.2.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from plotly) (8.0.1)
Requirement already satisfied: nbformat in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (5.3.0)
Requirement already satisfied: jupyter-core in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbformat) (4.9.2)
Requirement already satisfied: traitlets>=4.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbformat) (5.1.1)
Requirement already satisfied: fastjsonschema in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbformat) (2.15.1)
Requirement already satisfied: jsonschema>=2.6 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbformat) (4.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from jsonschema>=2.6->nbformat) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from jsonschema>=2.6->nbformat) (21.4.0)
Requirement already satisfied: matplotlib in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (3.5.1)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: cycler>=0.10 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: packaging>=20.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: numpy>=1.17 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib) (1.21.5)
Requirement already satisfied: pillow>=6.2.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib) (9.0.1)
Requirement already satisfied: six>=1.5 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: scipy in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (1.7.3)
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from scipy) (1.21.5)
Requirement already satisfied: sklearn in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (0.0)
Requirement already satisfied: scikit-learn in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from sklearn) (1.0.2)
Requirement already satisfied: joblib>=0.11 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn) (1.1.0)
Requirement already satisfied: numpy>=1.14.6 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn) (1.21.5)
Requirement already satisfied: scipy>=1.1.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from scikit-learn->sklearn) (2.2.0)
Requirement already satisfied: textblob in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (0.17.1)
Requirement already satisfied: nltk>=3.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from textblob) (3.7)
Requirement already satisfied: regex>=2021.8.3 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nltk>=3.1->textblob) (2022.3.15)
Requirement already satisfied: tqdm in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nltk>=3.1->textblob) (4.64.0)
Requirement already satisfied: joblib in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nltk>=3.1->textblob) (1.1.0)
Requirement already satisfied: click in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nltk>=3.1->textblob) (8.0.4)
Collecting wordcloud
  Using cached wordcloud-1.8.1.tar.gz (220 kB)
Requirement already satisfied: numpy>=1.6.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from wordcloud) (1.21.5)
Requirement already satisfied: pillow in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from wordcloud) (9.0.1)
Requirement already satisfied: matplotlib in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from wordcloud) (3.5.1)
Requirement already satisfied: packaging>=20.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (1.3.2)
Requirement already satisfied: python-dateutil>=2.7 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: cycler>=0.10 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (4.25.0)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib->wordcloud) (3.0.4)
Requirement already satisfied: six>=1.5 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)
Building wheels for collected packages: wordcloud
  Building wheel for wordcloud (setup.py) ... done
  Created wheel for wordcloud: filename=wordcloud-1.8.1-cp39-cp39-macosx_10_9_x86_64.whl size=154058 sha256=ca258e7bd50a014f9940669b37f2208fca34ba0d0c33974a4de3dd4e659719e9
  Stored in directory: /Users/ahmad/Library/Caches/pip/wheels/f9/7a/dd/06ef8b5dfe5483f6204133c08eeb16c287cc2c05e290ae2fc0
Successfully built wordcloud
Installing collected packages: wordcloud
Successfully installed wordcloud-1.8.1
Requirement already satisfied: seaborn in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (0.11.2)
Requirement already satisfied: matplotlib>=2.2 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from seaborn) (3.5.1)
Requirement already satisfied: numpy>=1.15 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from seaborn) (1.21.5)
Requirement already satisfied: pandas>=0.23 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from seaborn) (1.4.2)
Requirement already satisfied: scipy>=1.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from seaborn) (1.7.3)
Requirement already satisfied: pyparsing>=2.2.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (3.0.4)
Requirement already satisfied: python-dateutil>=2.7 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (2.8.2)
Requirement already satisfied: fonttools>=4.22.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (4.25.0)
Requirement already satisfied: packaging>=20.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (21.3)
Requirement already satisfied: cycler>=0.10 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (9.0.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from matplotlib>=2.2->seaborn) (1.3.2)
Requirement already satisfied: pytz>=2020.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from pandas>=0.23->seaborn) (2021.3)
Requirement already satisfied: six>=1.5 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn) (1.16.0)
Requirement already satisfied: nbconvert in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (6.4.4)
Requirement already satisfied: testpath in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (0.5.0)
Requirement already satisfied: traitlets>=5.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (5.1.1)
Requirement already satisfied: nbformat>=4.4 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (5.3.0)
Requirement already satisfied: entrypoints>=0.2.2 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (0.4)
Requirement already satisfied: bleach in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (4.1.0)
Requirement already satisfied: beautifulsoup4 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (4.11.1)
Requirement already satisfied: defusedxml in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (0.7.1)
Requirement already satisfied: jinja2>=2.4 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (2.11.3)
Requirement already satisfied: jupyterlab-pygments in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (0.1.2)
Requirement already satisfied: jupyter-core in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (4.9.2)
Requirement already satisfied: pandocfilters>=1.4.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (1.5.0)
Requirement already satisfied: pygments>=2.4.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (2.11.2)
Requirement already satisfied: mistune<2,>=0.8.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (0.8.4)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbconvert) (0.5.13)
Requirement already satisfied: MarkupSafe>=0.23 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from jinja2>=2.4->nbconvert) (2.0.1)
Requirement already satisfied: nest-asyncio in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (1.5.5)
Requirement already satisfied: jupyter-client>=6.1.5 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbclient<0.6.0,>=0.5.0->nbconvert) (6.1.12)
Requirement already satisfied: pyzmq>=13 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (22.3.0)
Requirement already satisfied: python-dateutil>=2.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (2.8.2)
Requirement already satisfied: tornado>=4.1 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (6.1)
Requirement already satisfied: fastjsonschema in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbformat>=4.4->nbconvert) (2.15.1)
Requirement already satisfied: jsonschema>=2.6 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from nbformat>=4.4->nbconvert) (4.4.0)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from jsonschema>=2.6->nbformat>=4.4->nbconvert) (0.18.0)
Requirement already satisfied: attrs>=17.4.0 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from jsonschema>=2.6->nbformat>=4.4->nbconvert) (21.4.0)
Requirement already satisfied: six>=1.5 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from python-dateutil>=2.1->jupyter-client>=6.1.5->nbclient<0.6.0,>=0.5.0->nbconvert) (1.16.0)
Requirement already satisfied: soupsieve>1.2 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from beautifulsoup4->nbconvert) (2.3.1)
Requirement already satisfied: packaging in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from bleach->nbconvert) (21.3)
Requirement already satisfied: webencodings in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from bleach->nbconvert) (0.5.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /Users/ahmad/opt/anaconda3/lib/python3.9/site-packages (from packaging->bleach->nbconvert) (3.0.4)
```

In [4]:
```python
#Importing required libraries

import pandas as pd
import numpy as np
import regex as re
import nltk
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import string
from nltk.stem import PorterStemmer
import plotly.express as px
%matplotlib inline
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
from nltk.probability import FreqDist
import scipy as sp
from textblob import TextBlob
nltk.download('twitter_samples')
from wordcloud import WordCloud
nltk.download('wordnet')
nltk.download('omw-1.4')
from nltk.stem import WordNetLemmatizer
import seaborn as sns
```

```
[nltk_data] Downloading package stopwords to /Users/ahmad/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /Users/ahmad/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package twitter_samples to
[nltk_data]     /Users/ahmad/nltk_data...
[nltk_data]   Package twitter_samples is already up-to-date!
[nltk_data] Downloading package wordnet to /Users/ahmad/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /Users/ahmad/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
```

In [5]:
```python
# Loading the csv files for stock specific dataset into pandas dataframes
# Since, There is no single extensive dataset available and annotated for trading stocks,
# therefore, combining three different datasets to fine tune model and improve its accuracy.

#1300 tweets manually annotated with negative and positive and reviewed related to investing and stock
dataset1 = pd.read_csv('tweets_labelled_dataset1.csv', sep=';')
# Labelled dataset for 5000 entries related to investing and stock
dataset2 = pd.read_csv('tweets_labelled_dataset2.csv')
# Labelled dataset with 33,946 entires related to investing and stock
dataset3 = pd.read_csv('tweets_labelled_dataset3.csv')

# General purpose dataset annotated from  between positive, negative and neutral.
# Dataset is annotated based on emotions ":( or :)" having 1.6 Million tweets.
# It is general purpose dataset not specifically related to stocks.
dataset4 = pd.read_csv('Tweet_general_dataset4.csv', encoding = "ISO-8859-1", names=["sentiment", "id", "date_c
```

Dataset-1 evaluation and dropping null Values

In [6]:
```python
#Dataset-1 has 1300 tweets manually annotated and independently reviewed as per the description of dataset prov
dataset1.count()
```

Out[6]:
```
id            5000
created_at    5000
text          5000
sentiment     1300
dtype: int64
```

In [7]:
```python
#Count of null Values
dataset1.isnull().sum()
```

Out[7]:
```
id               0
created_at       0
text             0
sentiment     3700
dtype: int64
```

In [8]:
```python
# Dropping rows with where at least one element is missing in row.
# Only Sentiment column was check for the missing values
dataset1 = dataset1.dropna(axis=0, subset=['sentiment'])
```

Dropping columns and changing the columns heading to make dataset unique with each other for combination

In [9]:
```python
dataset1['Label'] = 'dataset1'
dataset1 = dataset1.drop(['id', 'created_at'], axis=1)
dataset1 = dataset1.rename(columns={"text": "Text", "sentiment": "Sentiment"})
```

In [10]:
```python
dataset1.head(5)
```

Out[10]:

	Text	Sentiment	Label
0	RT @RobertBeadles: Yo\nEnter to WIN 1,000 Mon...	positive	dataset1
1	#SriLanka surcharge on fuel removed!\n\nThe ...	negative	dataset1
2	Net issuance increases to fund fiscal programs...	positive	dataset1
3	RT @bentboolean: How much of Amazon's traffic ...	positive	dataset1
4	$AMD Ryzen 4000 desktop CPUs looking 'great' a...	positive	dataset1

Dataset-2 evaluation and dropping null Values and columns not required

In [12]:
```python
# Dataset-2 has 5791 stock related tweets annotated for positive, negative and neutral
# as per the description of dataset provided on Kaggle.
dataset2.count()
```

Out[12]:
```
Text         5791
Sentiment    5791
dtype: int64
```

Since, there is no missing rows, therefore, no rows are dropped. However, sentiments are provided in digits. Therefore, these are changes to positive, negative or neutral based on values

In [13]:
```python
# Providing labels to the dataset because it will be eventually combined with other datasets.
dataset2['Label'] = 'dataset2'
dataset2.head(10)
```

Out[13]:

	Text	Sentiment	Label
0	Kickers on my watchlist XIDE TIT SOQ PNK CPW B...	1	dataset2
1	user: AAP MOVIE. 55% return for the FEA/GEED i...	1	dataset2
2	user I'd be afraid to short AMZN - they are lo...	1	dataset2
3	MNTA Over 12.00	1	dataset2
4	OI Over 21.37	1	dataset2
5	PGNX Over 3.04	1	dataset2
6	AAP - user if so then the current downtrend wi...	-1	dataset2
7	Monday's relative weakness. NYX WIN TIE TAP IC...	-1	dataset2
8	GOOG - ower trend line channel test & volume s...	1	dataset2
9	AAP will watch tomorrow for ONG entry.	1	dataset2

In [14]:
```python
#Function to change the sentiments to positive, negative and neutral based on sentiment values
def get_sentiment(value):
    if value < 0:
        return 'negative'
    elif value == 0:
        return 'neutral'
    else:
        return 'positive'
```

In [15]:
```python
#Applying get_sentiment function to dataset2
dataset2['Sentiment'] = dataset2['Sentiment'].apply(get_sentiment)
```

In [16]:
```python
dataset2.head(5)
```

Out[16]:

	Text	Sentiment	Label
0	Kickers on my watchlist XIDE TIT SOQ PNK CPW B...	positive	dataset2
1	user: AAP MOVIE. 55% return for the FEA/GEED i...	positive	dataset2
2	user I'd be afraid to short AMZN - they are lo...	positive	dataset2
3	MNTA Over 12.00	positive	dataset2
4	OI Over 21.37	positive	dataset2

Dataset-3 evaluation and dropping null Values and columns not required

In [17]:
```python
#Checking counts for the data fields
dataset3.count()
```

Out[17]:
```
id                       33946
text                     33946
text_sentiment           33946
username                 33946
hashtags                 33945
created_at               33946
user followers count     33946
replycount               33946
retweetcount             33946
likecount                33946
quotecount               33946
language                 33946
media                        0
retweetedTweet               0
quotedtweet                  0
inReplyToTweetId          4948
inReplyToUser             4948
mentionedUsers            5533
dtype: int64
```

Since, there is no missing value in text_sentiment and text and both records are matching with each other by having 33,946 entries each. Therefore, we will only drop irrelevant columns from the dataset

In [18]:
```python
# Renaming the columns to match the dataset1 and dataset2
dataset3 = dataset3.rename(columns={"text": "Text", "text_sentiment": "Sentiment"})
dataset3['Label'] = 'dataset3'
# Droping the columns which are not required
dataset3 = dataset3.drop(['id', 'username', 'hashtags', 'created_at', 'user followers count', 'replycount' ,
               'retweetcount', 'likecount', 'quotecount', 'quotecount', 'language', 'media' ,
               'retweetedTweet', 'quotedtweet', 'inReplyToTweetId', 'inReplyToUser', 'mentionedUsers'   ], axi
```

In [19]:
```python
dataset3.head(5)
```

Out[19]:

	Text	Sentiment	Label
0	When will the #NYSE #stockmarketcrash happen?	Neutral	dataset3
1	Aaj ka gyan:\n\nIf a company isn't a quality c...	Negative	dataset3
2	The stock market needs to crash hard to make i...	Negative	dataset3
3	Those who are "Buying on DIP" will very soon b...	Neutral	dataset3
4	@rdrhwke I wish our so-called President were t...	Positive	dataset3

Dataset-4 evaluation and dropping null Values and columns not required

In [20]:
```python
# Checking the number of records in dataset4 to see if there is any missing record.
dataset4.count()
```

Out[20]:
```
sentiment       1600000
id              1600000
date_created    1600000
flag            1600000
user            1600000
text            1600000
dtype: int64
```

There is no missing value in this dataset. However, extra coulumns will be drop to make it inline with other dataset

In [21]:
```python
# Providing label to the dataset because it will be eventually combined with other datasets.
dataset4['Label'] = 'dataset4'
# Dropping the columns which are not required
dataset4 = dataset4.drop(['id', 'date_created', 'flag', 'user'], axis=1)
```

In [22]:
```python
# Polarity of the tweet (0 = negative, 2 = neutral, 4 = positive) based on description provided in Kaggle
def general_df_sentiment(value):
    if value < 2:
        return 'negative'
    elif value > 2 :
        return 'positive'
    else:
        return 'neutral'
```

In [23]:
```python
#Changing 0, 2 or 4 to nagative, neutral or positive

dataset4['sentiment'] = dataset4['sentiment'].apply(general_df_sentiment)

# Renaming the columns to match the dataset1, dataset2 and dataset3
dataset4 = dataset4.rename(columns={"text": "Text", "sentiment": "Sentiment"})
```

In [24]:
```python
# Changing the order of column to match with other dataframes
dataset4 = dataset4[['Text', 'Sentiment', 'Label']]
```

In [26]:
```python
dataset4.head(5)
```

Out[26]:

	Text	Sentiment	Label
0	@switchfoot http://twitpic.com/2y1zl - Awww, t...	negative	dataset4
1	is upset that he can't update his Facebook by ...	negative	dataset4
2	@Kenichan I dived many times for the ball. Man...	negative	dataset4
3	my whole body feels itchy and like its on fire	negative	dataset4
4	@nationwideclass no, it's not behaving at all....	negative	dataset4

Combining all 4 datset for preprocessing

In [27]:
```python
# Combining all datasets into one dataset
combined_dataset = pd.concat([dataset1,dataset2,dataset3,dataset4])
```

In [28]:
```python
#Re-indexing after combining the dataset into one combined dataset
combined_dataset = combined_dataset.reset_index(drop=True)
```

In [29]:
```python
combined_dataset
```

Out[29]:

	Text	Sentiment	Label
0	RT @RobertBeadles: Yo\nEnter to WIN 1,000 Mon...	positive	dataset1
1	#SriLanka surcharge on fuel removed!\n\nThe ...	negative	dataset1
2	Net issuance increases to fund fiscal programs...	positive	dataset1
3	RT @bentboolean: How much of Amazon's traffic ...	positive	dataset1
4	$AMD Ryzen 4000 desktop CPUs looking 'great' a...	positive	dataset1
...	...	...	...
1641032	Just woke up. Having no school is the best fee...	positive	dataset4
1641033	TheWDB.com - Very cool to hear old Walt interv...	positive	dataset4
1641034	Are you ready for your MoJo Makeover? Ask me f...	positive	dataset4
1641035	Happy 38th Birthday to my boo of alll time!!! ...	positive	dataset4
1641036	happy #charitytuesday @theNSPCC @SparksCharity...	positive	dataset4

1641037 rows × 3 columns

There are 1,641,037 record appearing in combined datasets after dropping the record having null values either in text or sentiment attributes.

Data Cleaning

- Checking null values and count of combined datasets

In [30]:
```python
# Function to get the complete text for tweets as we have to check this again and again in the code to follow

def print_dataset(ds,counts:int,txt:str):
    j=0
    for i in ds.head(counts)[txt]:
        j+=1
        print(j,"=>",i)
        print()
```

In [31]:
```python
#Since, we have already dropped null values above, therefore, there is no null value in combined dataset
combined_dataset.isnull().sum()
```

Out[31]:
```
Text         0
Sentiment    0
Label        0
dtype: int64
```

In [32]:
```python
# Count of all columns is matching exactly with 1,641,037 records.
combined_dataset.count()
```

Out[32]:
```
Text         1641037
Sentiment    1641037
Label        1641037
dtype: int64
```

- Checking for duplicates entries in the combined datset

In [34]:
```python
# We have merged the 4 different datasets of tweets with each other.
# Therefore, There is possibility of having one tweet two or more times in the combined dataset.
# Moreover, there is also possibility of having duplicate record in original datasets.
# Which could also be verified by the checking the number of duplicate rows in the dataframe.

duplicates = combined_dataset.duplicated(subset=['Text'], keep='first')


duplicates.groupby(duplicates).count()
```

Out[34]:
```
False    1621705
True       19332
dtype: int64
```

In [35]:
```python
# Duplicated tweets and their corresponding Sentiment or Labers are removed.
combined_dataset = combined_dataset.drop_duplicates(['Text'], keep='first')
```

In [36]:
```python
#Re-indexing after dropping the duplicated rows
combined_dataset = combined_dataset.reset_index(drop=True)
```

In [37]:
```python
# Original combined dataset  [1,641,037] less Duplicated values [19,332] = 1,621,705
combined_dataset.count()
```

Out[37]:
```
Text         1621705
Sentiment    1621705
Label        1621705
dtype: int64
```

In [38]:
```python
combined_dataset
```

Out[38]:

	Text	Sentiment	Label
0	RT @RobertBeadles: Yo💥\nEnter to WIN 1,000 Mon...	positive	dataset1
1	#SriLanka surcharge on fuel removed!\n⛽📉\nThe ...	negative	dataset1
2	Net issuance increases to fund fiscal programs...	positive	dataset1
3	RT @bentboolean: How much of Amazon's traffic ...	positive	dataset1
4	$AMD Ryzen 4000 desktop CPUs looking 'great' a...	positive	dataset1
...	...	...	...
1621700	Just woke up. Having no school is the best fee...	positive	dataset4
1621701	TheWDB.com - Very cool to hear old Walt interv...	positive	dataset4
1621702	Are you ready for your MoJo Makeover? Ask me f...	positive	dataset4
1621703	Happy 38th Birthday to my boo of alll time!!! ...	positive	dataset4
1621704	happy #charitytuesday @theNSPCC @SparksCharity...	positive	dataset4

1621705 rows × 3 columns

In [39]:
```python
#Converting the body of the text to string format
combined_dataset['Text'] = (combined_dataset['Text'].copy()).astype("string")
```

In [40]:
```python
# Printing first 10 record of combined dataset for better understanding before performing further cleaning proc
print_dataset(combined_dataset,10,'Text')
```

```
1 => RT @RobertBeadles: Yo💥
Enter to WIN 1,000 Monarch Tokens✅

US Stock Market Crashes &amp; what we can LEARN from them PT3!

RETWEET, WATCH video…

2 => #SriLanka surcharge on fuel removed!
⛽📉
The surcharge of Rs.26 imposed on diesel and petrol has been revoked with effect from midnight on June 23 says
Power, Energy and Transport Minister Mahinda. Amaraweera -Adaderana-

#lka #FuelPrices #taxes #economy #stocks #StockMarket

3 => Net issuance increases to fund fiscal programs &gt; yields spike higher &gt; risk off: #stocks and #EMFX c
orrect lower &gt; #Fed comes in with #YCC &gt; stocks to new all time highs with 20% unemployment and -25% GDP.
Fin.

4 => RT @bentboolean: How much of Amazon's traffic is served by Fastly? Help us find out by running this tool f
rom your IP address: https://t.co…

5 => $AMD Ryzen 4000 desktop CPUs looking 'great' and on track to launch in 2020 https://t.co/y7yYvXOVYJ #madtw
eets #stocks #cnbc #AMD

6 => RT @QuantTrend: Reduce your portfolio RISK! GOLD is a perfect tail HEDGE!

Central banks balance sheet expansion &amp; large fiscal deficits &amp;…

7 => $863.69 Million in Sales Expected for Spirit AeroSystems Holdings, Inc. $SPR This Quarter https://t.co/zoq
BvspVSj #stocks

8 => RT @ArjunKharpal: #Apple has cut the prices of the iPhone 11 range by about 12-13% in China. It's an uncom
mon move.

These discounts are n…

9 => RT @SMA_alpha: The #CDC U.S. New Case data has a 2 day lag, but saw another encouraging decline #WHO Globa
l New Case data still flat at the…

10 => Where to Look for Dependable Dividends
Read More &gt; https://t.co/qKvNFF2ih5
#etf #investing #stocks #business #news
```

In [42]:
```python
# Checking datatypes of the columns
combined_dataset.dtypes
```

Out[42]:
```
Text         string
Sentiment    object
Label        object
dtype: object
```

- Cleaning dataset using regrex

In [43]:
```python
#There are links which should be removed and there are hashtags which should be removed.
def remove_unwanted_text(text):
    text = re.sub(r'http\S+', '', text) # remove links
    text = re.sub(r'@[A-Za-z0-9]+', ' ', text) # remove @mentions
    text = re.sub(r'\$\w*', '', text) # remove $ symbols and
    text = re.sub(r'^RT[\s]+', '', text) # remove RT old style retweet text
    text = re.sub(r'[0-9]+', '', text) #Remove digits
    text = re.sub(r'#', ' ', text) # remove hashtags
    text = re.sub(r'_', ' ', text) # remove Underscore
    #Returning clean text
    return text

#Testing the function
sentance = '''Good muffins``cost $3.88\nin '%%!the New York. #Midterm Please buy me
... two 's of them having green apple colour stock.\n\nThanks. @UOL :) $AAPL _, https://london.ac.uk/ {}'''
sentance = remove_unwanted_text(sentance)

sentance
```

Out[43]:
```
"Good muffins``cost .\nin '%%!the New York.  Midterm Please buy me\n... two 's of them having green apple colou
r stock.\n\nThanks.   :)   ,  {}"
```

In [46]:
```python
# apply the function to the text column
combined_dataset['Text'] = combined_dataset['Text'].apply(remove_unwanted_text)
```

In [47]:
```python
combined_dataset
```

Out[47]:

	Text	Sentiment	Label
0	: Yo💥\nEnter to WIN , Monarch Tokens✅\n\nUS St...	positive	dataset1
1	SriLanka surcharge on fuel removed!\n⛽📉\nThe ...	negative	dataset1
2	Net issuance increases to fund fiscal programs...	positive	dataset1
3	: How much of Amazon's traffic is served by Fa...	positive	dataset1
4	Ryzen desktop CPUs looking 'great' and on tr...	positive	dataset1
...	...	...	...
1621700	Just woke up. Having no school is the best fee...	positive	dataset4
1621701	TheWDB.com - Very cool to hear old Walt interv...	positive	dataset4
1621702	Are you ready for your MoJo Makeover? Ask me f...	positive	dataset4
1621703	Happy th Birthday to my boo of alll time!!! Tu...	positive	dataset4
1621704	happy charitytuesday	positive	dataset4

1621705 rows × 3 columns

- Removing Emotion Emojies, Stopwords and punctuations

In [48]:
```python
emojis = set([
    ':-)', ':)', ';)', ':o)', ':]', ':3', ':c)', ':>', '=]', '8)', '=)', ':}',
    ':^)', ':-D', ':D', '8-D', '8D', 'x-D', 'xD', 'X-D', 'XD', '=-D', '=D',
    '=-3', '=3', ':-))', ":'-)", ":')", ':*', ':^*', '>:P', ':-P', ':P', 'X-P',
    'x-p', 'xp', 'XP', ':-p', ':p', '=p', ':-b', ':b', '>:)', '>;)', '>:-)',
    '<3':L', ':-/', '>:/', ':S', '>:[', ':@', ':-(', ':[', ':-||', '=L', ':<',
    ':-[', ':-<', '=\\', '=/', '>:(', ':(', '>.<', ":'-(", ":'(", ':\\', ':-c',
    ':c', ':{', '>:\\', ';('
    ])
```

In [49]:
```python
# We will now remove the stop words from the dataset to make it more focused on the context
# We can use spacy, nltk, sklearn or genism for this purpose. However,
# i have used nltk because it result is very close to spacy
# [https://towardsdatascience.com/text-pre-processing-stop-words-removal-using-different-libraries-f20bac19929a

stop_words = (stopwords.words('english'))
punctuation = '''!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'s``'0123456789''' #Removing punctuations and digits
def remove_stopwords_punch(text):
    # Getting word token for the text parsed in
    word_tokens = word_tokenize(text)
    #Check for the stop_words and removing from the text
    filtered_stop_words = [w for w in word_tokens if not w.lower() in stop_words]
    #Checking for the punctuation and removing from the text
    filtered_punch_words = [w for w in filtered_stop_words if not w in punctuation]
    #Checking for the emojis and removing from the text
    filtered_emoj_words = [w for w in filtered_punch_words if not w in punctuation]
    #Joining the words again for input to dataframe column
    return(' '.join(filtered_emoj_words))

#Testing the function
remove_stopwords_punch(sentance)
```

Out[49]:
```
'Good muffins cost New York Midterm Please buy ... two green apple colour stock Thanks'
```

In [50]:
```python
#Removing stopwords and punctuation from the dataset.
# apply the stop word function to the body column in dataframe
combined_dataset['Text'] = combined_dataset['Text'].apply(remove_stopwords_punch)
```

In [51]:
```python
print_dataset(combined_dataset,5,'Text')
```

```
1 => Yo💥 Enter WIN Monarch Tokens✅ US Stock Market Crashes amp LEARN PT RETWEET WATCH video…

2 => SriLanka surcharge fuel removed ⛽📉 surcharge Rs imposed diesel petrol revoked effect midnight June says
Power Energy Transport Minister Mahinda Amaraweera -Adaderana- lka FuelPrices taxes economy stocks StockMarket

3 => Net issuance increases fund fiscal programs gt yields spike higher gt risk stocks EMFX correct lower gt Fe
d comes YCC gt stocks new time highs unemployment GDP Fin

4 => much Amazon traffic served Fastly Help us find running tool IP address

5 => Ryzen desktop CPUs looking ' great track launch madtweets stocks cnbc AMD
```

- There are still some emojies and flags or other symbols left in the datafram, which are also removed using the DeEmojify function

In [53]:
```python
# Function to remove emojies still left in the text
# [https://stackoverflow.com/questions/53322870/unable-to-remove-some-emojis-from-tweets-in-python]

def deEmojify(text):
    regrex_pattern = re.compile(pattern = "["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"  # chinese char
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                           "]+", flags = re.UNICODE)
    return regrex_pattern.sub(r'', text)
```

In [54]:
```python
#Applying deEmojify function on the text
combined_dataset['Text'] = combined_dataset['Text'].apply(deEmojify)
```

In [55]:
```python
print_dataset(combined_dataset,5,'Text')
```

```
1 => Yo Enter WIN Monarch Tokens US Stock Market Crashes amp LEARN PT RETWEET WATCH video…

2 => SriLanka surcharge fuel removed  surcharge Rs imposed diesel petrol revoked effect midnight June says Powe
r Energy Transport Minister Mahinda Amaraweera -Adaderana- lka FuelPrices taxes economy stocks StockMarket

3 => Net issuance increases fund fiscal programs gt yields spike higher gt risk stocks EMFX correct lower gt Fe
d comes YCC gt stocks new time highs unemployment GDP Fin

4 => much Amazon traffic served Fastly Help us find running tool IP address

5 => Ryzen desktop CPUs looking ' great track launch madtweets stocks cnbc AMD
```

- Making the text string to lower case to make it consistant

In [56]:
```python
# Making the tweet text to lower case for consistency.
combined_dataset['Text']  = combined_dataset.Text.str.lower()
# Making Sentiments to lower case for consistency.
combined_dataset['Sentiment']  = combined_dataset.Sentiment.str.lower()
```

- Stemming and lemmatizing tweets text

In [57]:
```python
#Stemming and lemmatizing tweets  to their root.
stemming = PorterStemmer()
lemmatizer = WordNetLemmatizer()
#Function to stem the sentences of the dataframe
def stem_tweets(text):
     word_tokens = word_tokenize(text) # Getting word token for the text parsed in
     stem_words = [stemming.stem(w) for w in word_tokens]
     lem_words = [lemmatizer.lemmatize(w) for w in stem_words]
     return(' '.join(lem_words))

#Testing the function
sentance = "Programmers program with programming languages corpora"
stem_tweets(sentance)
```

Out[57]:
```
'programm program with program languag corpus'
```

In [58]:
```python
#Applying stem_tweets function on the text
combined_dataset['Text'] = combined_dataset['Text'].apply(stem_tweets)
```

- Checking for duplicates after Pre-processing and cleaning the dataset

In [59]:
```python
# Identifying duplicates after pre-cleaning because there might be some records became unique after cleaning th
duplicates = combined_dataset.duplicated(subset=['Text'], keep='first')
duplicates.groupby(duplicates).count()
```

Out[59]:
```
False    1527616
True       94089
dtype: int64
```

In [62]:
```python
# Duplicated tweets and their corresponding Sentiment or Labels are removed.
combined_dataset = combined_dataset.drop_duplicates(['Text'], keep='first')

#Reindexing after dropping the duplicated rows
combined_dataset = combined_dataset.reset_index(drop=True)
```

In [63]:
```python
combined_dataset
```

Out[63]:

	Text	Sentiment	Label
0	yo enter win monarch token u stock market cras...	positive	dataset1
1	srilanka surcharg fuel remov surcharg r impos ...	negative	dataset1
2	net issuanc increas fund fiscal program gt yie...	positive	dataset1
3	much amazon traffic serv fastli help u find ru...	positive	dataset1
4	ryzen desktop cpu look ' great track launch ma...	positive	dataset1
...	...	...	...
1527611	woke school best feel ever	positive	dataset4
1527612	thewdb.com cool hear old walt interview â «	positive	dataset4
1527613	readi mojo makeov ask detail	positive	dataset4
1527614	happi th birthday boo alll time tupac amaru sh...	positive	dataset4
1527615	happi charitytuesday	positive	dataset4

1527616 rows × 3 columns

1,527,616 records are left after preprocessing and cleaning the text. Now data is in shape for further analysis and training the models

- Checking Label listing before performing feature extraction and analysis

In [64]:
```python
# Getting the sentiments from the tweets
combined_dataset.Sentiment.value_counts()
```

Out[64]:
```
negative    763999
positive    752531
neutral      11086
Name: Sentiment, dtype: int64
```

Since, our datatset for "neutral comments" is not big enough for the models to work properly because all the weight will be given towards either postive or negative side. Therefore, rows with neutral sentiment are drop before to feature extraction and analysis

In [65]:
```python
combined_dataset = combined_dataset.drop(combined_dataset.index[combined_dataset['Sentiment'] == "neutral"])
combined_dataset = combined_dataset.reset_index(drop=True)
```

2.2 | Dataframe Outlook and Features Analysis

- Checking the types of tweets in our dataset

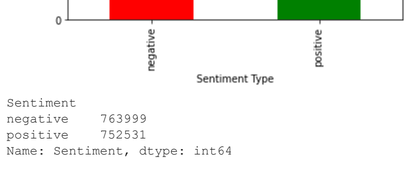
In [68]:
```python
#Checking the number of posted based on stock type
df_tweet = combined_dataset.groupby('Sentiment')['Sentiment'].count()
df_tweet = df_tweet.sort_values(ascending = False) # Keeping the values in asending order

bar_colors = list('rgbkymc')  #red, green, blue, black, etc.

df_tweet.plot(x = df_tweet[0], y= df_tweet[1], kind = 'bar', stacked=True, color=bar_colors)

# To Avoid scientific numnber on axis
plt.gcf().axes[0].yaxis.get_major_formatter().set_scientific(False)

plt.title('Tweets posted')
plt.ylabel('Number of Tweets')
plt.xlabel('Sentiment Type')

plt.show()

df_tweet
```

Out[68]:
```
Sentiment
negative    763999
positive    752531
Name: Sentiment, dtype: int64
```

- Checking word frequency in our "Text" for tweets

In [69]:
```python
#Checking the frequency of words in dataset
#Using itertools for efficiency purposes
import itertools
def word_frequency(text):
    word_tokens = word_tokenize(text) # Getting word token for the text parsedin
    return word_tokens #Returning word tokens

# It is showing repeating sentances in our dataset.
# Based on results it seems many tweets are posted by bot.
# These tweets are not removed while performing preprocessing using duplicated method of pandas

f = list(itertools.chain.from_iterable(combined_dataset['Text'].apply(word_frequency)))

#These dots are kept as it text field of dataset for better learning by the model
ignore = {'...','..',}

result = FreqDist(f)
```

In [70]:
```python
result.plot(30, cumulative=False)
```



Out[70]:
```
<AxesSubplot:xlabel='Samples', ylabel='Counts'>
```

".", ".." or "..." kept in the dataset because it might be useful for the model to get better understanding of the text. We can notice that word shows affections and emphasis the situation (i.e. love, work , today, feel or want etc) are used a lot to express the sentiments in twitter text.

- Feature extraction using count Vectorizer to get the distribution of words for positive and negative sentiments

In [71]:
```python
# I have end up having error of utilizing too much ram to get the term frequency over complete dataset using SK
# This error was resolved using native python functions such as .sum() and numpy np.asarray().squeeze() to conv
# However, above mentioned solution was not perfect for those cases where complete sparse matrix
# (including all combined tweets) were converted to numpy array using method ".toarray()"
# The error was "Memory Error: Unable to allocate 2.79 TiB for an array with shape (1527616, 250973) and data t
# on selecting combined dataset.
# This error is coming because the dataset  is too much for Ram memory to execute operations
# Therefore, Dataset was divided into 100 subsets and middle subset was selected from feature extraction and pr

split_tweet_dataset = np.array_split(combined_dataset, 100)

shrink_data = split_tweet_dataset[50] # Taking the mid dataset to get the combined values from all datasets
```

In [72]:
```python
from sklearn.feature_extraction.text import CountVectorizer
tweet_text=shrink_data['Text'] #Extracting tweets text from the data frame
vec=CountVectorizer(stop_words='english') #  CountVectorization  to convert a collection of text documents to a
tweets_encoded=vec.fit_transform(tweet_text)  #Creating a document-term matrix
```

In [73]:
```python
# Count vector created a sparse matrix
tweets_encoded
```

Out[73]:
```
<15165x14008 sparse matrix of type '<class 'numpy.int64'>'
        with 100608 stored elements in Compressed Sparse Row format>
```

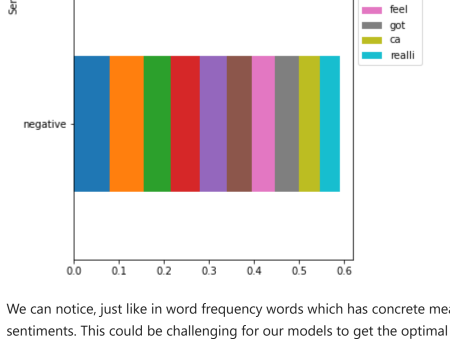
In [74]:
```python
#Converting sparse matrix to numpy array to make it less resource hunger

sum_sparse_matrix = tweets_encoded.sum(axis=0) #Sum of element over axis [0 is used for axis]
arr = np.asarray(sum_sparse_matrix).squeeze() # Removing axis of length a
tweets_encoded_arr = np.asarray(arr).reshape((arr.size,)) #Reshapping array to it size
```

In [75]:
```python
# Matrix converted to array using .toarray() method to make them less resource hunger
term_count=zip(tweets_encoded_arr, vec.get_feature_names_out())
# Sorted list of term counts
sorted_term_count=sorted(list(term_count), reverse=True)
```

In [76]:
```python
top_terms_counts=[tc[0] for tc in sorted_term_count[:10]] #Getting top 10 terms counts
top_terms=[t[1] for t in sorted_term_count[:10]]  #Getting top 10 terms
term_df=pd.DataFrame(tweets_encoded.toarray(), columns=vec.get_feature_names_out())
term_df['Sentiment']=combined_dataset['Sentiment']
```

In [78]:
```python
# Plotting the top 10 terms used in the dataset
term_df[top_terms+['Sentiment']].groupby('Sentiment').mean().plot(kind='barh', stacked=True, figsize=(5, 10))
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5))
```

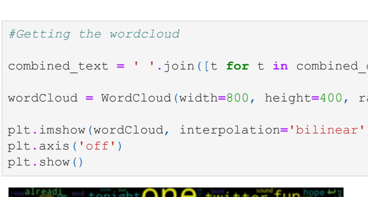
Out[78]:
```
<matplotlib.legend.Legend at 0x7faa0ae1b5e0>
```



We can notice, just like in word frequency words which has concrete meaning of doing something are used both in positive and negative sentiments. This could be challenging for our models to get the optimal accuracy because same words are almost used in similar ratio in both types of sentiments. However, It is expected that naive bayes could be performing well because it usually improve its judgment based on naive assumptions.

- Word cloud in dataset

In [80]:
```python
#Getting the wordcloud

combined_text = ' '.join([t for t in combined_dataset['Text']])

wordCloud = WordCloud(width=800, height=400, random_state=35, max_font_size=110).generate(combined_text)

plt.imshow(wordCloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



2.3 | Splitting Dataset [Training: 75 % , Test: 25%]

- I have used SKlearn to split the data into train and test. cleaned "Text" in dataframe is used as ['Feature'] and "Sentiments" are used as ['Target'].
  - It will select the training and testing sets based on random sampling.
  - Train set for Text and Sentiment will be used for training the models
  - Test set will be used to evaluate the performance of models
  - The purpose of this division is to judge that how the model will act on the unseen data

Resouse Reference:

1. *https://towardsdatascience.com/understanding-train-test-split-scikit-learn-python-ea676d5e3d1*

In [81]:
```python
from sklearn.model_selection import train_test_split
# This will result a list with the split between 75% ['Training Dataset'] and 25% ['Testing Dataset']
# Random state is kept at 0 to see how well models will perform to input because
# Same Data will be used for Classification approach.
X_train, X_test, y_train, y_test = train_test_split(combined_dataset.Text, combined_dataset.Sentiment, train_si
```

In [82]:
```python
X_train
```

Out[82]:
```
78388                         mood work today gon na way busi
861734     want job quot extrem angler quot ... hooray sc...
1363203                   lol hey know thing lawn mower right
1140986                                lol .... know fool
397927                                          dunno one
                                ...
152315              ga price keep go jump like cent last week
963395     good morn twittervers work parent hous morn be...
117952                     lone ....... bore ...... tire yet
1484405    brace may tomorrow ... plea plea plea let come...
305711            hey splogin system get jumpi spap.q live gt
Name: Text, Length: 1061571, dtype: object
```

In [83]:
```python
y_train
```

Out[83]:
```
78388      negative
861734     positive
1363203    positive
1140986    positive
397927     negative
             ...
152315     negative
963395     positive
117952     negative
1484405    positive
305711     negative
Name: Sentiment, Length: 1061571, dtype: object
```

In [84]:
```python
X_test
```

Out[84]:
```
1199744                  good night hope cbbc fingi goe well
385472                             enjoy lie listent rain
846346                                              co-op
701359                                       aw crap ice
568315                           ye turn smiley face upsid
                                ...
379892                 alway find way spend much money starv
1229615                 final interview tomorrow finger cross
1074356                                   back kingdom prais
370054     hmm amazon link wo n't work pay itun one thank...
911467     astro run behind huh big road trip rocki start...
Name: Text, Length: 454959, dtype: object
```

In [85]:
```python
y_test
```

Out[85]:
```
1199744    positive
385472     negative
846346     positive
701359     negative
568315     negative
             ...
379892     negative
1229615    positive
1074356    positive
370054     negative
911467     positive
Name: Sentiment, Length: 454959, dtype: object
```

You can notice that same length of documents divided into feature and label by the SKlearn class

	Train	Test
x	1,061,571	454,959
y	1,061,571	454,959

2.4 | Feature Extraction

- Step 1: Tokenization
  - Tokenization is the process of dividing text into "sentences, words or aplhabets".
  - There are multiple method for tokenization and almost all libraries such as NLTK, keras, spaCy has capabilities of performing tokeization.
  - We have used NLTK tokenize so far during preprocessing and cleaning of the Datasets. Therefore, we will continue to use the same during this stage.
  - There is separate function created for tokenization using the "word_tokenize" method from NLTK.
  - Example:
    - "Natural language processing is fun to learn" **Tokenize to**
    - "Natura" "language" "processing" "is" "fun" "to" "learn"

In [86]:
```python
# Function to tokenize the sentences of datasets
def tokenize(s): return word_tokenize(s)
```

- Step 2: Feature Extraction using term frequency-inverse document frequency (td-idf) technize and Bag of Words
  - Next step is get the numerical representation of documents ['Twitter Datatset'] in a vector.
  - We have already used countvector in the dataset analysis and feature presentation section.
  - Vectorization on pandas dataset will result in sparse matrix having characteristic:
    - Column: Number of explicit token in the document.
    - Rows: Number of documents in the set of documents
    - Cell: Count of frequency of words.
  - I have selected TF-IDF means Term Frequency — Inverse Document Frequency over count vector for vectorization due to following reasons:
    - Count vectors will only check the frequecny of text in a text or doucments.
    - It will not categorize high or least important words.
    - Words rarely used in the corpus will be given full weight.
    - It lacks the power of identifing lingustic relationship between the words.
    - TF-IDF is another layer on top of count vectorizers. It not just cosnider the frquency but also it relevance of importnace.
    - It is possible to reduce the input dimension by removing less important words

TFIDF

For a term i in document j:

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Resouse Reference:

1. *https://medium.com/artificial-coder count-vectorizers-vs-tfidf-vectorizers-natural-language-processing-b5371f51a40c]*
2. *https://www.analyticsvidhya.com/blog/2021/07/bag-of-words-vs-tfidf-vectorization-a-hands-on-tutorial/*

In [87]:
```python
from sklearn.feature_extraction.text import TfidfVectorizer
tfid = TfidfVectorizer(strip_accents='unicode', tokenizer=tokenize, ngram_range=(1, 2), max_df=0.9, min_df=3, s

# Function to apply tranform on test and train subsets
# It will return the sparse matrix
def bagwords(data , method:str):
    if method == 'ft':
        return tfid.fit_transform(data)  #Creating a document-term matrix with fit_transform method
    elif method == 't':
        return tfid.transform(data) #Test sparse document-term matrix with transform method
    else:
        print('Please enter correct data or input method')



X_tf_train = bagwords(X_train, 'ft') #Term frequency of train dataset
X_tf_test = bagwords(X_test, 't') #Term frequency of test dataset

#Explanation on Scikit-lear Transofrmer and fit, tranform and fit_transform
#https://www.analyticsvidhya.com/blog/2021/04/difference-between-fit-transform-fit_transform-methods-in-scikit-
```

In [88]:
```python
X_tf_train
```

Out[88]:
```
<1061571x418862 sparse matrix of type '<class 'numpy.float64'>'
        with 12812048 stored elements in Compressed Sparse Row format>
```

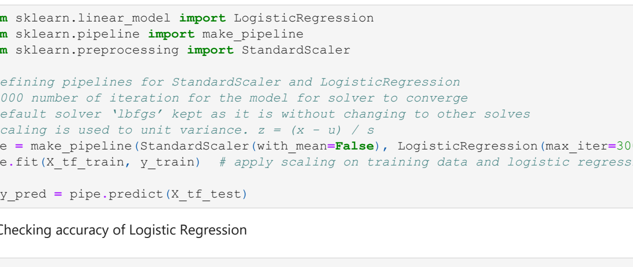
In [89]:
```python
X_tf_test
```

Out[89]:
```
<454959x418862 sparse matrix of type '<class 'numpy.float64'>'
        with 5327632 stored elements in Compressed Sparse Row format>
```

454,959 are test documents and 1,061,571 are training documents. Cell containing zeros are 418,862 represents small portion of tokens

2.5 | Baseline Performance

- I have selected **logistic regression** as baseline performance for supervised learning
  - Logistic regression in NLP used for classification problems.
  - It takes the data point of n features and classify as 1 or 0.
  - Scaler out is fet to signmoid function to get the scaler output classes in logistic regression.
  - If probability is greated than 0.5 then class 1 is defined by the algorithm.
  - If probability is less than 0.5 then class 0 is defined by the algorithm.



Resource Reference:

1. *https://medium.com/@rithwikkukunuri30/logistic-regression-and-its-applications-in-natural-language-processing-5f835c901fa7*

In [90]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

# Defining pipelines for StandardScaler and LogisticRegression
# 3000 number of iteration for the model for solver to converge
# Default solver 'lbfgs' kept as it is without changing to other solves
# Scaling is used to unit variance. z = (x - u) / s
pipe = make_pipeline(StandardScaler(with_mean=False), LogisticRegression(max_iter=3000))
pipe.fit(X_tf_train, y_train)  # apply scaling on training data and logistic regression classifier

lg_y_pred = pipe.predict(X_tf_test)
```

- Checking accuracy of Logistic Regression

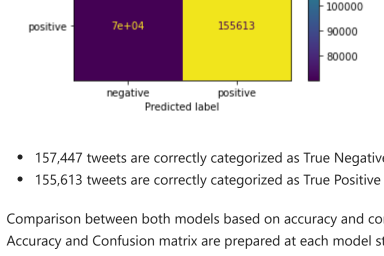
In [91]:
```python
from sklearn import metrics
print("Logistic regression accuracy(%):", metrics.accuracy_score(y_test, lg_y_pred)*100)
```

```
Logistic regression accuracy(%): 68.81059611965034
```

- Confusion Matrix of Logistic Regression

In [92]:
```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
cm = confusion_matrix(y_test, lg_y_pred, labels=pipe.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=pipe.classes_)
plt.gcf().set_size_inches(15, 15)
disp.plot()
plt.show()
```

```
<Figure size 1080x1080 with 0 Axes>
```



- 157,447 tweets are correctly categorized as True Negative (TN)
- 155,613 tweets are correctly categorized as True Positive (TP)

Comparison between both models based on accuracy and confusion matrix is conducted in the evaluation section below. However, Accuracy and Confusion matrix are prepared at each model stage

2.6 | Classification approach

- I have selected **Naive Bayes** as classification approach for supervised learning but Multinominal bayes classifier is used for model training. It is expected to give better performance over regression logistic based due to following reasons:
  - Bayes therom finds the probability of evetn based on event already occured
  - It chec the probability of even after the evidence is seen.
  - Naive assumption are taken in the naive bayes classifier to split the evidence into distint parts
  - Multinomial Naive Bayes: Feature vectors represent the frequencies with which certain events have been generated by a multinomial distribution.

Resource Reference:

1. *https://www.geeksforgeeks.org/naive-bayes-classifiers/?ref=gcse*

In [93]:
```
# Importing Multinomial naive bayes from sklearn
from sklearn.naive_bayes import MultinomialNB
# Declaring a classifier
nb = MultinomialNB()
# Training a model
nb.fit(X_tf_train, y_train) # Term frequency is provided for training set
```
Out[93]:
```
MultinomialNB()
```

In [94]:
```
# Testing model by providing testing dataset
y_pred = nb.predict(X_tf_test)
```

In [95]:
```
import collections
# Checking count of Sentiments in model predictions
collections.Counter(y_pred)
```
Out[95]:
```
Counter({'positive': 224652, 'negative': 230307})
```

In [96]:
```
collections.Counter(y_test)
```
Out[96]:
```
Counter({'positive': 225772, 'negative': 229187})
```

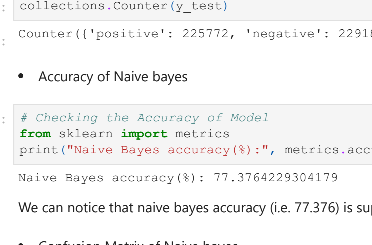
- Accuracy of Naive bayes

In [98]:
```
# Checking the Accuracy of Model
from sklearn import metrics
print("Naive Bayes accuracy(%):", metrics.accuracy_score(y_test, y_pred)*100)
```
```
Naive Bayes accuracy(%): 77.3764229304179
```

We can notice that naive bayes accuracy (i.e. 77.376) is superior in comparison to Logistic regression (68.18).

- Confusion Matrix of Naive bayes

In [99]:
```
cm = confusion_matrix(y_test, y_pred, labels=nb.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=nb.classes_)
disp.plot()
plt.show()
```



- 178,283 tweets are correctly categorized as True Negative (TN)
- 173,749 tweets are correctly categorized as True Positive (TP)

Accuracy of both models are further analysed in the section below using "Precision, Recall, F1-Measure and ROC" matrics.

Section 3. Conclusion

3.1 | Precision, Recall & F1-Measure

- **Accuracy** checks the correct prediction made by model over the total observation
- **Precision** checks the count of correct prediction made by the model
- **Recall** is opposit to the precision. It check how often correct prediction made by the model when result is actually correct
- **F-measre** is harmoic mean of both precision and recall

Formulas

- **Accuracy:** Number of correct predictions / total number of predictions ==> (TP + TN) / (TP + TN + FP + FN)
- **Precision:** (TP) / (TP + FP)
- **Recall:** (TP) / (TP + FN)
- **F1 Measure:** 2 * (precision * recall) / (precision + recall)

Resource Reference:

1. *https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62*
2. *https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/*

Since, we have already caluclated accuracy of both classifier above, therefore, these are not reproduced or recalculated in this section.

- **Baseline performance -** Precision, Recall and F1 Meaure of linear regression

```
Precession Score
```

In [101…
```
#Checking the precession score of logistic regression
print('Logistic Regression Precision score: %.3f' % metrics.precision_score(y_test, lg_y_pred, pos_label='posit
```
```
Logistic Regression Precision score: 0.684
```

```
Recall  Score
```

In [102…
```
#Checking the recall score of logistic regression
print('Logistic Regression Recall score: %.3f' % metrics.recall_score(y_test,lg_y_pred, pos_label='positive'))
```
```
Logistic Regression Recall score: 0.689
```

```
F1 Measure
```

In [103…
```
#Checking F1 measure of Logistic regression
print('Logistic Regression  F1 measure : %.3f' % metrics.f1_score(y_test, lg_y_pred, pos_label='positive'))
```
```
Logistic Regression  F1 measure : 0.687
```

- **Classification approach -** Precision, Recall and F1 Meaure of Naive Bayes

```
Precession Score
```

In [106…
```
#Checking the precession score of Naive Bayes
print('Naive Bayes Precision score: %.3f' % metrics.precision_score(y_test, y_pred, pos_label='positive'))
```
```
Naive Bayes Precision score: 0.773
```

```
Recall  Score
```

In [107…
```
#Checking the recall score of Naive Bayes
print('Naive Bayes Recall score: %.3f' % metrics.recall_score(y_test,y_pred, pos_label='positive'))
```
```
Naive Bayes Recall score: 0.770
```

```
F1 Measure
```

In [108…
```
#Checking F1 measure of Naive Bayes
print('Naive Bayes F1 measure : %.3f' % metrics.f1_score(y_test, y_pred, pos_label='positive'))
```
```
Naive Bayes F1 measure : 0.771
```

Comparison between Logical Regression and Multinomial Naive Bayes model

	Logical Regression	Multinomial Naive Bayes
Accuracy	0.688	0.774
Precision	0.684	0.773
Recall	0.689	0.770
F-1 Measure	0.687	0.771

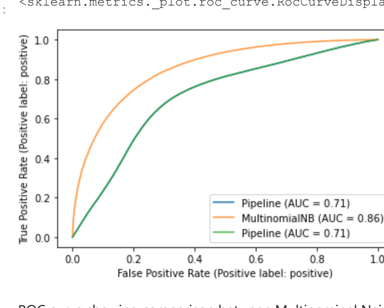
3.2 | ROC Curve

- **ROC curve (receiver operating characteristic curve)** It is a graph to show the performance of Classsifier at classification thresholdes. It is based on two parameters:
  - **True Positive Rate:** synonym for recall ==> TPR = (TP) / (TP + FN)
  - **False Positive Rate:** FPR = (FP) / (FP + TN)

  Resouse Reference:

  1. *https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc*

In [109…
```
from sklearn.metrics import RocCurveDisplay

# To display ROC curve for both classifiers
lr_disp = RocCurveDisplay.from_estimator(pipe, X_tf_test, y_test)

ax = plt.gca()
nb_disp = RocCurveDisplay.from_estimator(nb, X_tf_test, y_test, ax=ax, alpha=0.8)
lr_disp.plot(ax=ax, alpha=0.8)
```
Out[109]:
```
<sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x7faab390c670>
```



ROC curve showing comparison between Multinominal Naive Bayes and Regression logistic classifier. We can notice that AUC (Area under the ROC Curve) shows that Naive Bayes was efficient since begining and it has more probability of ranking random positive than a random negative. AUC rate of Naive Bayes is 0.86 in comparison to Logistic Regression (which is referred as "Pipeline" in chart above)

3.3 | Summary and Conclusions

Conclusion

The purpose of this project is to prepare a text classifier, which could be beneficial for the investor or financial institution in assessing the public rational behaviour in order to align their investment decisions accordingly. Here is a quick recap of the approach successfully followed:

- Arrange a twitter dataset;
- Conduct pre-processing of the dataset;
- Start with quick and simple model;
- Get the classification prediction from base model;
- Prepare another classifier using a different approach to match the performance with the base model.

In terms of comparison between baseline and classification approach, Multinomial Naive Bayes has clearly outperformed Logical regression in categorizing the tweets as True Positive or True Negative. The Accuracy of Naive Bayes is 12.5% more in comparison to Logical Regression. Moreover, Naive Bayes has also posted better scores also for precision, recall and F-1 Measure.

Furthermore, ROC-AUC also shows Naive Bayes with an efficiency ratio of 0.86, and it has better categorization since start. Naive Bayes performed well in comparison, but the accuracy ratio of 77.4% is less than other models available online. It could be because of following reasons:

- The datasets used to train other models are quite small, and text classifiers such as Naive Bayes perform quite well on small datasets.
- As we have seen in the features extraction and presentation part, the same terms are used very often among positive and negative sentiments.

There is no standard benchmark available to judge any classifier, but there is always room for improvement on any implementation.

Further Improvement

- The performance and accuracy could be measured by using other classifiers of the Naive Bayes family, such as Gaussian Naive Bayes and Bernoulli Naive Bayes. These may model them more accurately and perform better as a result.
- Secondly, There is also the possibility of getting better results by changing re-processing methodology that converts text to numbers.

~ 313 Word Count

General Resource Reference:

Plotting Guideline:

- https://towardsdatascience.com/different-bar-charts-in-python-6d984b9c6b17
- https://blog.insightdatascience.com/how-to-solve-90-of-nlp-problems-a-step-by-step-guide-fda605278e4e

Markdown Guideline:

- https://www.markdownguide.org/basic-syntax/
- https://www.markdownguide.org/hacks/#color

Literature:

- Speculator and Influencer Evaluation in Stock Market by Using Social Media [2020 IEEE publication - International Conference on Big Data (Big Data)]
- Sentiment Analysis in Financial Markets [2014 IEEE publication - University of Siegen Institute of Knowledge Based Systems Germany]