```
import numpy as np
import pandas as pd
from nltk.classify import NaiveBayesClassifier
from nltk.corpus import subjectivity
from nltk.sentiment import SentimentAnalyzer
from nltk.sentiment.util import *
import matplotlib.pyplot as mlpt
```

```
    /usr/local/lib/python3.7/dist-packages/nltk/twitter/__init__.py:20: UserWarning:
      warnings.warn("The twython library has not been installed. "
```

```
import tweepy
import csv
import pandas as pd
import random
import numpy as np
import pandas as pd
```

News Data

```
# import data
url = 'https://raw.githubusercontent.com/trucntx007/NLP/data/stockerbot-export.csv'
df = pd.read_csv(url, error_bad_lines=False)

df.info()
```

```
    <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 f
```

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

	id	text	timestamp	source	symbols	com
3814	1017485603582238700	Firstenergy Corp $FE Stock Value Rose While Th...	Thu Jul 12 19:07:26 +0000 2018	reurope_stock	FE	Firs
26595	1019696729602474000	Copy lucrative traders automatically with Bitc...	Wed Jul 18 21:33:40 +0000 2018	msarybsarstee	ETN	C
25884	1019684122099216400	https://t.co/0mvtSFNitP #Nadella has done a g...	Wed Jul 18 20:43:34 +0000 2018	paranjpe12	MSFT	
23710	1019639090340229100	$TMK New Insider Filing On: LUTEK BEN Transac...	Wed Jul 18 17:44:37 +0000 2018	filing_scanner	TMK	
11429	1018955475914764300	$BA $JPM $EOG $GS $ORCL $BAC $V $EL $SPB $ACN	Mon Jul 16 20:28:11 +0000 2018	teresaarthur66	UTX	

News Data Prep

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

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:13: SettingWithCopy
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
  del sys.path[0]
0         VIDEO   I was in my office  I was minding my o...
1         The price of lumber  LB F is down 22  since hi...
2                     Who says the American Dream is dead
3         Barry Silbert is extremely optimistic on bitco...
4         How satellites avoid attacks and space junk wh...
                                ...
```

```
28259            FB   29234a9c 7f08 4d5a 985f cb1a5554ecf9
28260                                                   BTC
28261    RT  invest in hd   Nuff said    TEL  telcoin  ...
28262                                                   BTC
28263    Stellar  XLM price   0 297852 Binance registra...
Name: text, Length: 28264, dtype: object
```

```
# 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'])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopyW
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
```

```
df_cleaned.head(10)
```

	timestamp	text	symbols	datetime
11882	Mon Jul 16 23:54:50 +0000 2018	a pa ion du football tait d j connue	A	2018-07-16
11903	Tue Jul 17 00:04:26 +0000 2018	RT ShaneOliverAMP EZ shares 0 2 US shares ...	A	2018-07-17
11981	Tue Jul 17 00:45:27 +0000 2018	The geometric R matrix for affine crystals o...	A	2018-07-17
12026	Tue Jul 17 01:14:22 +0000 2018	I AM ONE EXAM BAM IT CRY OUT HUBS A...	A	2018-07-17
12068	Tue Jul 17 01:44:22 +0000 2018	ACE A FIG CRY TOO BIG I AM MS PEN T...	A	2018-07-17
12201	Tue Jul 17 02:56:35 +0000 2018	The only good moment on tonight s raw a ha	A	2018-07-17
12218	Tue Jul 17 03:11:54 +0000 2018	nicoleyyy17 m4R M1nG 4m p A 0r	A	2018-07-17

```
# new dataframe to house combined text
# text of the same company (same symbol) on a same date will be groupped together

df_cleaned1 = pd.DataFrame(columns=['datetime','text','symbols'])



# 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 = T

        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 = Tru
      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
3	NaN	myhedghog Yeah I saw AABA was selling roughl...	AABA	2018-07-11
4	NaN	BABA YAHOY AABA	AABA	2018-07-12
...	...	...	...	...
2303	NaN	Zoetis Inc ZTS Given Average Rating of Buy ...	ZTS	2018-07-15
2304	NaN	Scan results MACD Bearish Centerline Cross t...	ZTS	2018-07-16
2305	NaN	ZTS Zoetis Inc SEC Filing Form 4	ZTS	2018-07-17
2306	NaN	ZTS New Insider Filing On Fenton Andrew Tr...	ZTS	2018-07-18
2307	NaN	When you try to gauge sentiment on a ticker b...	ticker	2018-07-12

2308 rows × 4 columns

```
# drop 'datetime' column since already created 'Date'
df_cleaned1 = df_cleaned1.dropna(axis=1)
```

```
# df_cleaned1 has tweets groupby symbols and Date
df_cleaned1.sort_values('symbols')
```

	text	symbols	Date
0	a pa ion du football tait d j connue	A	2018-07-16
1	A repeat of 2002 Walmart may be looking to...	A	2018-07-17
2	ACE OUT A CGI HE PAY ALL DATA YELP ...	A	2018-07-18
10	Head To Head Analysis Unisys UIS and Altaba...	AABA	2018-07-18
8	Today s million dollar PUT options trade AM...	AABA	2018-07-16
...	...	...	...
2299	Analyst portfolio manager hunting ideas Her...	ZTS	2018-07-11
2298	Abaxis ABAX and Zoetis Merger Deal Crosses H...	ZTS	2018-07-10
2306	ZTS New Insider Filing On Fenton Andrew Tr...	ZTS	2018-07-18
2301	ZTS the bull pattern is confirmed amp a BUY...	ZTS	2018-07-13
2307	When you try to gauge sentiment on a ticker b...	ticker	2018-07-12

2308 rows × 3 columns

```
# Split timestamp
# df_cleaned['timestamp'] = pd.to_datetime(df_cleaned['timestamp'],format='%Y-%m-%d')

# df_cleaned[['dayofweek','month','day','time','timezone', 'year']] = df_cleaned.time
# df_cleaned[['hour','minute','second']] = df_cleaned.time.str.split(':',expand=True)

# df_cleaned['datetime'] = df_cleaned[['year', 'month', 'day']].agg('-'.join, axis=1)
```

Stock Data

```
stocks = pd.read_csv('https://raw.githubusercontent.com/trucntx007/NLP/main/stocks_cl
# Define the ticker list

tickers_list = []

for i in range(len(stocks)):
  tickers = stocks['ticker'][i]
  tickers_list.append(tickers)
```

```
tickers_list
```

```
!pip install yfinance
```

```
Collecting yfinance
  Downloading yfinance-0.1.67-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: multitasking>=0.0.7 in /usr/local/lib/python3.7/d
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: numpy>=1.15 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: requests>=2.20 in /usr/local/lib/python3.7/dist-p
Collecting lxml>=4.5.1
  Downloading lxml-4.6.4-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.m
     |████████████████████████████████| 6.3 MB 7.6 MB/s
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/di
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dis
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/l
Installing collected packages: lxml, yfinance
  Attempting uninstall: lxml
    Found existing installation: lxml 4.2.6
    Uninstalling lxml-4.2.6:
      Successfully uninstalled lxml-4.2.6
Successfully installed lxml-4.6.4 yfinance-0.1.67
```

```
# Fetch data from Yahoo Finance
import yfinance as yf

data = yf.download(tickers_list, start='2018-7-9',end='2018-7-19')['Adj Close']

# Drop cols with no data
data = data.dropna(axis=1)

# Print first 5 rows of the data
print(data.head())
```

```
- S: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- GLUU: No data found, symbol may be delisted
- FOXA: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- HTZ: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- ABX: No data found for this date range, symbol may be delisted
- LB: No data found, symbol may be delisted
- MON: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- JEC: No data found, symbol may be delisted
- CY: No data found, symbol may be delisted
- MYL: No data found, symbol may be delisted
- DLPH: No data found, symbol may be delisted
- AABA: No data found, symbol may be delisted
- AKS: No data found, symbol may be delisted
- FLIR: No data found, symbol may be delisted
- WFT: No data found, symbol may be delisted
- CXO: No data found, symbol may be delisted
- FNSR: No data found, symbol may be delisted
- APC: No data found, symbol may be delisted
- SYMC: No data found, symbol may be delisted
- ALXN: No data found, symbol may be delisted
- ETFC: No data found, symbol may be delisted
- MNK: No data found, symbol may be delisted
- FOX: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- TWX: No data found for this date range, symbol may be delisted
- BBT: No data found, symbol may be delisted
- ARRY: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- CBS: No data found, symbol may be delisted
- HDS: No data found, symbol may be delisted
- ESV: No data found, symbol may be delisted
- CRZO: No data found, symbol may be delisted
- HRS: No data found, symbol may be delisted
- VIAB: No data found, symbol may be delisted
- PAH: No data found for this date range, symbol may be delisted
- RHT: No data found, symbol may be delisted
- FMSA: No data found for this date range, symbol may be delisted
- DCIX: No data found, symbol may be delisted
- XL: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- BHGE: No data found, symbol may be delisted
- TIF: No data found, symbol may be delisted
- CELG: No data found, symbol may be delisted
- ETE: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- ARNC: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- WPX: No data found, symbol may be delisted
- JCP: No data found, symbol may be delisted
- UTX: No data found, symbol may be delisted
- NBL: No data found, symbol may be delisted
- STI: No data found, symbol may be delisted
- TSS: No data found, symbol may be delisted

- PX: Data doesn't exist for startDate = 1531094400, endDate = 1531958400
- VRX: No data found for this date range, symbol may be delisted
- KORS: None
                    A        AAL       AAOI  ...       ZION  ZNGA        ZTS
Date                                         ...
2018-07-09  61.546665  38.477581  45.330002  ...  48.723938  4.24  85.723831
2018-07-10  62.082947  38.291603  45.790001  ...  48.263168  4.19  84.479462
2018-07-11  61.293140  35.198521  45.759998  ...  48.001163  4.26  82.794174
2018-07-12  61.868435  35.560684  48.799999  ...  47.477158  4.40  83.744606
2018-07-13  61.800171  36.333954  47.779999  ...  46.763416  4.34  84.401085
```

```
# add column datetime
data[data.index.name] = data.index


# Transpose stock dataset
```

```
data_T = data
data_T = data_T.transpose()
data_T['symbols'] = data_T.index
```

```
# price of a stock during the period
data_Tr = data_T.melt(id_vars=['symbols'],var_name="Date", value_name='Price')
data_Tr.sort_values(['symbols', 'Date'])
```

	symbols	Date	Price
0	A	2018-07-09	61.5467
519	A	2018-07-10	62.0829
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

```
# get list of symbols
tickers_list = pd.Series(data_Tr['symbols'].unique())
tickers_list.to_csv('tickerlist.csv')
```

```
# save stock prices to StockData.csv
data.to_csv('StockData.csv')
```

Combine datasets

```
# add column 'Price' to df_cleaned1
df_cleaned1['Price'] = ''
```

```
df_cleaned1.sort_values('symbols')
```

	text	symbols	Date	Price
0	a pa ion du football tait d j connue	A	2018-07-16	
1	A repeat of 2002 Walmart may be looking to...	A	2018-07-17	
2	ACE OUT A CGI HE PAY ALL DATA YELP ...	A	2018-07-18	
10	Head To Head Analysis Unisys UIS and Altaba...	AABA	2018-07-18	
8	Today s million dollar PUT options trade AM...	AABA	2018-07-16	
...	...	...	...	...
2299	Analyst portfolio manager hunting ideas Her...	ZTS	2018-07-11	
2298	Abaxis ABAX and Zoetis Merger Deal Crosses H...	ZTS	2018-07-10	
2306	ZTS New Insider Filing On Fenton Andrew Tr...	ZTS	2018-07-18	
2301	ZTS the bull pattern is confirmed amp a BUY...	ZTS	2018-07-13	
2307	When you try to gauge sentiment on a ticker b...	ticker	2018-07-12	

2308 rows × 4 columns

```
# 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)
            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]


combined_data = df_cleaned1
combined_data.to_csv('combined_data.csv')


# 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)
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:670: SettingWithC
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stab
  iloc._setitem_with_indexer(indexer, value)
```

```
combined_data
```

	text	symbols	Date	Price	Emotion	Negative	Neutral	Positive
0	a pa ion du football tait d j connue	A	2018-07-16	61	0	0	1	0
1	A repeat of 2002 Walmart may be looking to...	A	2018-07-17	61	0	0	1	0
2	ACE OUT A CGI HE PAY ALL DATA YELP ...	A	2018-07-18	61	0.9657	0.033	0.887	0.9657
3	myhedghog Yeah I saw AABA was selling roughl...	AABA	2018-07-11	61	0.296	0	0.879	0.296
4	BABA YAHOY AABA	AABA	2018-07-12	61	0	0	1	0

NLP Text Processing

Rating of Buy ...

```
import re
import nltk
import ssl


nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import RegexpTokenizer
nltk.download('wordnet')
from nltk.stem.wordnet import WordNetLemmatizer
```

```
    [nltk_data] Downloading package stopwords to /root/nltk_data...
    [nltk_data]   Unzipping corpora/stopwords.zip.
    [nltk_data] Downloading package wordnet to /root/nltk_data...
    [nltk_data]   Unzipping corpora/wordnet.zip.
```

```
!pip install langdetect
```

```
    Collecting langdetect
      Downloading langdetect-1.0.9.tar.gz (981 kB)
         |████████████████████████████████| 981 kB 5.4 MB/s
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (fr
    Building wheels for collected packages: langdetect
      Building wheel for langdetect (setup.py) ... done
      Created wheel for langdetect: filename=langdetect-1.0.9-py3-none-any.whl size=
      Stored in directory: /root/.cache/pip/wheels/c5/96/8a/f90c59ed25d75e50a8c10a1b
    Successfully built langdetect
    Installing collected packages: langdetect
    Successfully installed langdetect-1.0.9
```

```
# 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

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']


df_cleaned
```

	timestamp	text	symbols	datetime	valid	language
11882	Mon Jul 16 23:54:50 +0000 2018	a pa ion du football tait d j connue	A	2018-07-16	True	en
11903	Tue Jul 17 00:04:26 +0000 2018	RT ShaneOliverAMP EZ shares 0 2 US shares ...	A	2018-07-17	True	en
11981	Tue Jul 17 00:45:27 +0000 2018	The geometric R matrix for affine crystals o...	A	2018-07-17	True	en
12026	Tue Jul 17 01:14:22 +0000 2018	I AM ONE EXAM BAM IT CRY OUT HUBS A...	A	2018-07-17	True	en
12068	Tue Jul 17 01:44:22 +0000 2018	ACE A FIG CRY TOO BIG I AM MS PEN T...	A	2018-07-17	True	en
...	...	...	...	...	...	...
22909	Wed Jul 18 16:33:47 +0000 2018	Analysts Set Zoetis Inc ZTS Target Price at ...	ZTS	2018-07-18	True	en
[illegible]	Wed Jul 18 [illegible]	Zoetis Inc ZTS to Issue	[illegible]	2018-07-	[illegible]	[illegible]

```
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: DeprecationWarn
19605    estate asset watch today outsized price move s...
2402     evening tweeted mid cap check gambiste top hel...
16908    see cdix offer msft amzn fb goog jnj jpm googo...
17246                             g plc long term investor
11476    keefe bruyette wood lower invesco ivz market p...
22220        patience value name xrx make back zone beyond
22006    xec open interest maturity high put high call ...
20358                               txt premarket winning
13116                                        miniapps mat
2484                analyst see eps avalonbay community avb
14362                           mtb new sec filing mtb form k
10920    illumina ilmn reach new month high analyst upg...
16382    pultegroup phm v tri pointe group tph financia...
19715                  broker issue forecast mobile u q tmus
22486    try gauge sentiment ticker come search crypto ...
15420    spy break keep eye amzn nflx fb aapl googl tsl...
12194    join u signal multiple paid group one join u g...
732      today archer daniel midland company adm report...
11286    ajaydevfan kashur bahot kiye ham dono par saja...
2602     yoyow currently worth register binance receive...
dtype: object
```

Sentiment Analysis

```
!pip install textblob
```

```
Requirement already satisfied: textblob in /usr/local/lib/python3.7/dist-package
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.7/dist-packag
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (fr
```

```
# Detect Emotions for each text Form TextBlob Library
from textblob import TextBlob

detectEmotion = []
detectPolarity = []

for txt in corpus:
    analysis = TextBlob(txt)
    Polarity = analysis.sentiment.polarity

    if Polarity < 0:
        emotion = '2'  #Negative
    elif Polarity > 0:
        emotion = '1'  #Positive
    else:
        emotion = '0'  #Neutral

    detectEmotion.append(emotion)
    detectPolarity.append(Polarity)

df_cleaned['Polarity'] = detectPolarity
df_cleaned['Emotion'] = detectEmotion


df_cleaned.sort_values(by = ['timestamp'])
#jul-9-2018
#jul-18
```

	timestamp	text	symbols	datetime	valid	language	keywords	Polar
47	Mon Jul 09 00:15:27 +0000 2018	NI high OI range is 26 00 to 26 00 for option...	NI	2018-07-09	True	en	NaN	0.160
48	Mon Jul 09 00:17:21 +0000 2018	RT TradeSatoshi AMG AMGCHAIN will be delis	AMG	2018-07-09	True	en	NaN	0.000

```
# 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']
```

RT forever t

```
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: 0.2766932005158536
    Percentage Negetive: 0.12304887268199403
    Percentage Neutral: 0.6002579268021524
```

max...

NLP Models

28259	23:46:13	7108 4d5a	FB	2018-07-	True	en	NaN	0.000

```
import nltk
from sklearn.feature_extraction.text import CountVectorizer,TfidfVectorizer
from nltk import ngrams
from sklearn.model_selection import train_test_split
import time
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

x_train, x_test, y_train, y_test = train_test_split(df_cleaned['text'], df_cleaned['Em
```

Wed Jul 18 price 0 2018-07-

```
# Ngram Models
def NgramModels(Model , txt, n):
    vect      = CountVectorizer(max_features=1000 , ngram_range=(n,n))
    train_vect= vect.fit_transform(x_train)
    test_vect = vect.transform(x_test)
```

```
        model     = Model
        t0        = time.time()
        model.fit(train_vect, y_train)
        t1        = time.time()
        predicted = model.predict(test_vect)
        t2        = time.time()
        time_train= t1-t0
        time_pred = t2-t1

        accuracy  = model.score(train_vect, y_train)
        predicted = model.predict(test_vect)

        report = classification_report(y_test, predicted, output_dict=True)
        print("Models with " , n , "-grams :\n")
        print('********************** \n')
        print(txt)
        print("Training time: %fs; Prediction time: %fs \n" % (time_train, time_pred))
        print('Accuracy score train set :', accuracy)
        print('Accuracy score test set  :', accuracy_score(y_test, predicted),'\n')
        print('Positive: ', report['1'])
        print('Neutral : ', report['0'])
        print('Negative: ', report['2'])
        print('\n -------------------------------------------------------------------------


    def KNN_Ngram(n):
        vect      = CountVectorizer(max_features=1000 , ngram_range=(n,n))
        train_vect= vect.fit_transform(x_train)
        test_vect = vect.transform(x_test)

        for k in [1,3,5,7,10]:
            model = KNeighborsClassifier(n_neighbors=k,algorithm='brute')
            t0        = time.time()
            model.fit(train_vect, y_train)
            t1        = time.time()
            predicted = model.predict(test_vect)
            t2        = time.time()
            time_train= t1-t0
            time_pred = t2-t1

            accuracy  = model.score(train_vect, y_train)
            predicted = model.predict(test_vect)

            report = classification_report(y_test, predicted, output_dict=True)

            print("Models with " , n , "-grams :\n")
            print('********************** \n')
            print("Classification Report for k = {} is:\n".format(k))
            print("Training time: %fs ; Prediction time: %fs \n" % (time_train, time_pred
            print('Accuracy score train set :', accuracy)
            print('Accuracy score test set  :', accuracy_score(y_test, predicted),'\n')
            print('Positive: ', report['1'])
            print('Neutral : ', report['0'])
            print('Negative: ', report['2'])
            print('\n ------------------------------------------------------------------


    def TFIDFModels(Model,txt):
        vect      = TfidfVectorizer(min_df = 5, max_df =0.8, sublinear_tf = True, use_idf
        train_vect= vect.fit_transform(x_train)
        test_vect = vect.transform(x_test)

        model     = Model
        t0        = time.time()
        model.fit(train_vect, y_train)
        t1        = time.time()
        predicted = model.predict(test_vect)
        t2        = time.time()
        time_train= t1-t0
        time_pred = t2-t1

        accuracy  = model.score(train_vect, y_train)
        predicted = model.predict(test_vect)

        report = classification_report(y_test, predicted, output_dict=True)

        print(txt)
        print("Training time: %fs; Prediction time: %fs \n" % (time_train, time_pred))
        print('Accuracy score train set :', accuracy)
        print('Accuracy score test set  :', accuracy_score(y_test, predicted),'\n')
        print('Positive: ', report['1'])
        print('Neutral : ', report['0'])
        print('Negative: ', report['2'])
        print('\n ----------------------------------------------------------------------


    def KNN_TFIDF():
        vect      = TfidfVectorizer(min_df = 5, max_df =0.8, sublinear_tf = True, use_idf
        train_vect= vect.fit_transform(x_train)
        test_vect = vect.transform(x_test)

        for k in [1,3,5,7,10]:

            model = KNeighborsClassifier(n_neighbors=k,algorithm='brute')
            t0        = time.time()
            model.fit(train_vect, y_train)
            t1        = time.time()
            predicted = model.predict(test_vect)
            t2        = time.time()
            time_train= t1-t0
            time_pred = t2-t1

            accuracy  = model.score(train_vect, y_train)
            predicted = model.predict(test_vect)

            report = classification_report(y_test, predicted, output_dict=True)

            print("Classification Report for k = {} is:\n".format(k))
            print("Training time: %fs ; Prediction time: %fs \n" % (time_train, time_pred
            print('Accuracy score train set :', accuracy)
            print('Accuracy score test set  :', accuracy_score(y_test, predicted),'\n')
            print('Positive: ', report['1'])
            print('Neutral : ', report['0'])
            print('Negative: ', report['2'])
            print('\n ------------------------------------------------------------------
```

Train Models and Make Predictions

```
SupportVectorClassifier=svm.SVC(kernel='linear')

LogReg2 = NgramModels(Model = LogisticRegression(),txt = '\nLogistic Regression Model
LogReg3 = NgramModels(Model = LogisticRegression(),txt = 'Logistic Regression Model :

svm2 = NgramModels(Model = SupportVectorClassifier ,txt = 'Support Vectoer Classifier
svm3 = NgramModels(Model = SupportVectorClassifier ,txt = 'Support Vectoer Classifier

DecTree2 = NgramModels(Model = tree.DecisionTreeClassifier(),txt = 'Decision Tree Clas
DecTree3 = NgramModels(Model = tree.DecisionTreeClassifier(),txt = 'Decision Tree Clas

KNN2=KNN_Ngram(2)
KNN3=KNN_Ngram(3)

print('Models with Tfidf Feature extraction Techniques : \n')
print('************************************************* \n')

LogReg = TFIDFModels(Model = LogisticRegression(),txt = 'Logistic Regression Model : 
svm = TFIDFModels(Model = SupportVectorClassifier,txt = 'Support Vector Classifier Mod
DecTree = TFIDFModels(Model = tree.DecisionTreeClassifier(),txt = 'Decision Tree Class
knn_tfidf = KNN_TFIDF()
```

```
Support Vector Classifier Model :

Training time: 24.267497s; Prediction time: 3.619398s

Accuracy score train set : 0.9783756740230141
Accuracy score test set  : 0.9573143619386394

Positive:  {'precision': 0.9718061674008811, 'recall': 0.926112510495382, 'f1-sc
Neutral :  {'precision': 0.9488695652173913, 'recall': 0.9952572053994893, 'f1-s
Negative:  {'precision': 0.9733606557377049, 'recall': 0.8392226148409894, 'f1-s

 ---------------------------------------------------------------------------------
```

```
Decision Tree Classifier Model :

Training time: 3.949574s; Prediction time: 0.003888s

Accuracy score train set : 0.9999444104730669
Accuracy score test set  : 0.9466429524232992

Positive:  {'precision': 0.9348561759729273, 'recall': 0.927791771620487, 'f1-sc
Neutral :  {'precision': 0.9638467100506146, 'recall': 0.9726377234585918, 'f1-s
Negative:  {'precision': 0.8854545454545455, 'recall': 0.8604240282685512, 'f1-s

 ----------------------------------------------------------------------------

Classification Report for k = 1 is:

Training time: 0.014141s ; Prediction time: 1.284858s

Accuracy score train set : 0.9999444104730669
Accuracy score test set  : 0.8268119164072921

Positive:  {'precision': 0.9417360285374554, 'recall': 0.6649874055415617, 'f1-s
Neutral :  {'precision': 0.8140560983296565, 'recall': 0.9423568040861, 'f1-scor
Negative:  {'precision': 0.7107438016528925, 'recall': 0.607773851590106, 'f1-sc

 ----------------------------------------------------------------------------


Classification Report for k = 3 is:

Training time: 0.021028s ; Prediction time: 1.512460s

Accuracy score train set : 0.8602479292901217
Accuracy score test set  : 0.7810137839039573

Positive:  {'precision': 0.9493293591654247, 'recall': 0.5348446683459278, 'f1-s
Neutral :  {'precision': 0.7564362163725774, 'recall': 0.9540313754104341, 'f1-s
Negative:  {'precision': 0.7054054054054054, 'recall': 0.46113074204946997, 'f1-

 ----------------------------------------------------------------------------

Classification Report for k = 5 is:

Training time: 0.022029s ; Prediction time: 1.862504s

Accuracy score train set : 0.8035466118183334
Accuracy score test set  : 0.7581147176522899
```

Results

```
idx = pd.MultiIndex.from_product([['2-grams', '3-grams', 'TFIDF'],['Accuracy Training
col = ['LogisticRegression', 'SupportVectorClassifier', 'DecisionTree', 'KNeighborsCla
```

```
Result = pd.DataFrame('*', idx, col)

Result.LogisticRegression=['79.17','76.92','73.60','71.99','95.73','93.35']
Result.SupportVectorClassifier=['78.91','76.94','73.65','72.15','97.83','95.99']
Result.DecisionTree=['82.66','77.32','74.12','72.15','1.0','95.96']
Result.KNeighborsClassifier=['80.77','74.24','73.08','70.69','1.0','82.46']


Result
```

FeatureExtraction	Metric	LogisticRegression	SupportVectorClassifier	Decisio
2-grams	Accuracy Training %	79.17	78.91	
	Accuracy Testing %	76.92	76.94	
3-grams	Accuracy Training %	73.60	73.65	
	Accuracy Testing %	71.99	72.15	
TFIDF	Accuracy Training %	95.73	97.83	
	Accuracy Testing %	93.35	95.99	

Stock Price Predictions using sentiment analysis

```
# columns to be used for prediction
df_stock_val = combined_data[['Date','Price','Emotion','Negative','Neutral','Positive'
df_stock_val
```

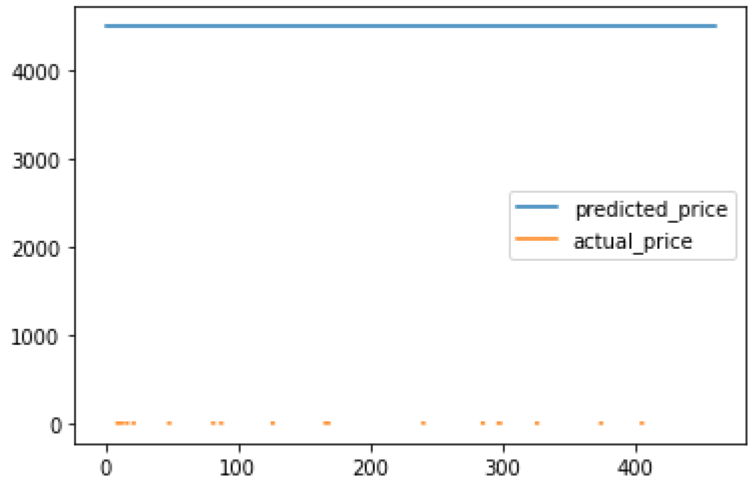
	Date	Price	Emotion	Negative	Neutral	Positive
0	2018-07-16	61	0	0	1	0
1	2018-07-17	61	0	0	1	0
2	2018-07-18	61	0.9657	0.033	0.887	0.9657
3	2018-07-11	61	0.296	0	0.879	0.296
4	2018-07-12	61	0	0	1	0
...	...	...	...	...	...	...
2303	2018-07-15	84	0	0	1	0
2304	2018-07-16	82	0	0	1	0
2307	2018-07-12	84	0	0	1	0

```
# Split dataset for training and testing
x_train, x_test, y_train, y_test = train_test_split(df_stock_val['Price'], df_stock_va
```

```
ls_sentiments_score = []
for date, row in x_train.iteritems():
    sentiment_score = np.asarray([combined_data.loc[date, 'Emotion']])
    ls_sentiments_score.append(sentiment_score)
numpy_dataframe_train = np.asarray(ls_sentiments_score)



ls_sentiments_score = []
for date, row in x_test.iteritems():
    sentiment_score = np.asarray([combined_data.loc[date, 'Emotion']])
    ls_sentiments_score.append(sentiment_score)
numpy_dataframe_test = np.asarray(ls_sentiments_score)



from sklearn.metrics import precision_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import accuracy_score


# from treeinterpreter import treeinterpreter as ti
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import classification_report,confusion_matrix

rf = RandomForestRegressor()
rf.fit(numpy_dataframe_train, y_train)
prediction=rf.predict(numpy_dataframe_test)
import matplotlib.pyplot as plt
%matplotlib inline
#idx = pd.date_range(test_data_start, test_data_end)
```

```
predictions_df = pd.DataFrame(data=prediction[0:], columns=['Price'])
predictions_df['Price'] = predictions_df['Price'].apply(np.int64)
predictions_df['Price'] = predictions_df['Price'] + 4500
predictions_df['actual_value'] = y_test
predictions_df.columns = ['predicted_price', 'actual_price']
predictions_df.plot()
predictions_df['predicted_price'] = predictions_df['predicted_price'].apply(np.int64)
y_test = y_test.apply(np.int64)
#print(accuracy_score(test['adj_close_price'],predictions_df['predicted_price']))
print(rf.score(numpy_dataframe_train, y_train))
```

```
0.9999984586900995
```



✓ 0s completed at 9:27 PM