# **STOCK PREDICTION USING TWITTER SENTIMENT ANALYSIS**

#### **importing machine learning libraries**

In [41]:

**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

#### **importing library to fetch data from twitter**

In [42]:

**import** tweepy

**import** csv

**import** pandas **as** pd

**import** random

**import** numpy **as** np

**import** pandas **as** pd

#### **setting up consumer key and access token**

In [43]:

consumer\_key **=** '3jmA1BqasLHfItBXj3KnAIGFB'

consumer\_secret **=** 'imyEeVTctFZuK62QHmL1I0AUAMudg5HKJDfkx0oR7oFbFinbvA'

access\_token **=** '265857263-pF1DRxgIcxUbxEEFtLwLODPzD3aMl6d4zOKlMnme'

access\_token\_secret **=** 'uUFoOOGeNJfOYD3atlcmPtaxxniXxQzAU4ESJLopA1lbC'

auth **=** tweepy**.**OAuthHandler(consumer\_key, consumer\_secret)

auth**.**set\_access\_token(access\_token, access\_token\_secret)

api **=** tweepy**.**API(auth,wait\_on\_rate\_limit**=True**)

#### **Fetching tweets for United Airlines in extended mode (means entire tweet will come and not just few words + link)**

In [44]:

fetch\_tweets**=**tweepy**.**Cursor(api**.**search, q**=**"#unitedAIRLINES",count**=**100, lang **=**"en",since**=**"2018-9-13", tweet\_mode**=**"extended")**.**items()

data**=**pd**.**DataFrame(data**=**[[tweet\_info**.**created\_at**.**date(),tweet\_info**.**full\_text]**for** tweet\_info **in** fetch\_tweets],columns**=**['Date','Tweets'])

In [ ]:

data

#### **Removing special character from each tweets**

In [45]:

data**.**to\_csv("Tweets.csv")

cdata**=**pd**.**DataFrame(columns**=**['Date','Tweets'])

total**=**100

index**=**0

**for** index,row **in** data**.**iterrows():

stre**=**row["Tweets"]

my\_new\_string **=** re**.**sub('[^ a-zA-Z0-9]', '', stre)

temp\_df **=** pd**.**DataFrame([[data["Date"]**.**iloc[index],

my\_new\_string]], columns **=** ['Date','Tweets'])

cdata **=** pd**.**concat([cdata, temp\_df], axis **=** 0)**.**reset\_index(drop **=** **True**)

*# index=index+1*

*#print(cdata.dtypes)*

#### **Displaying the data with date and tweets, you can notice there are multiple tweets for each day. So we will club them together later.**

In [46]:

cdata

Out[46]:

|  | **Date** | **Tweets** |
| --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... |
| **1** | 2021-09-22 | RT diecastryan A nice full lineup at IAD last ... |
| **2** | 2021-09-22 | United Airlines resuming Airline Tickets Reser... |
| **3** | 2021-09-22 | RT diecastryan A nice full lineup at IAD last ... |
| **4** | 2021-09-22 | lol FAANews united does not give a single damn... |
| **...** | ... | ... |
| **367** | 2021-09-13 | Thank You unitedAIRLINES httpstcoRU897P5rqI |
| **368** | 2021-09-13 | Where does the journey take you luggage tra... |
| **369** | 2021-09-13 | RT n194at United Air LinesDouglas DC852 N8062U... |
| **370** | 2021-09-13 | It is so ignorant to have 1299 in flight wifi ... |
| **371** | 2021-09-13 | Exactly But we have pretty options than United... |

372 rows × 2 columns

#### **Creating a dataframe where we will combine the tweets date wise and store into**

In [47]:

ccdata**=**pd**.**DataFrame(columns**=**['Date','Tweets'])

In [48]:

indx**=**0

get\_tweet**=**""

**for** i **in** range(0,len(cdata)**-**1):

get\_date**=**cdata**.**Date**.**iloc[i]

next\_date**=**cdata**.**Date**.**iloc[i**+**1]

**if**(str(get\_date)**==**str(next\_date)):

get\_tweet**=**get\_tweet**+**cdata**.**Tweets**.**iloc[i]**+**" "

**if**(str(get\_date)**!=**str(next\_date)):

temp\_df **=** pd**.**DataFrame([[get\_date,

get\_tweet]], columns **=** ['Date','Tweets'])

ccdata **=** pd**.**concat([ccdata, temp\_df], axis **=** 0)**.**reset\_index(drop **=** **True**)

get\_tweet**=**" "

#### **All the tweets has been clubbed as per their date.**

In [49]:

ccdata

Out[49]:

|  | **Date** | **Tweets** |
| --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... |

#### **Now to know the "closing price" of each day we will import STOCK PRICE DATA for UNITED AIRLINES from "yahoo.finance". We will consider "Close" price only.**

In [50]:

read\_stock\_p**=**pd**.**read\_csv('UAL.csv')

*# DOWNLOAD UPDATED CLOSE PRICE FROM https://finance.yahoo.com/quote/UAL/history?period1=1598918400&period2=1632268800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true*

read\_stock\_p

Out[50]:

|  | **Date** | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2020-09-01 | 35.250000 | 37.240002 | 34.950001 | 36.009998 | 36.009998 | 29722200 |
| **1** | 2020-09-02 | 36.099998 | 37.099998 | 35.209999 | 36.889999 | 36.889999 | 26622800 |
| **2** | 2020-09-03 | 37.130001 | 39.770000 | 36.139999 | 37.400002 | 37.400002 | 53966400 |
| **3** | 2020-09-04 | 38.150002 | 38.740002 | 36.459999 | 38.209999 | 38.209999 | 33121600 |
| **4** | 2020-09-08 | 37.299999 | 38.480000 | 36.480000 | 37.279999 | 37.279999 | 33207100 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **261** | 2021-09-15 | 43.650002 | 43.910000 | 43.020000 | 43.860001 | 43.860001 | 10321600 |
| **262** | 2021-09-16 | 43.860001 | 45.410000 | 43.849998 | 44.470001 | 44.470001 | 12204300 |
| **263** | 2021-09-17 | 44.779999 | 45.500000 | 44.110001 | 44.540001 | 44.540001 | 11733300 |
| **264** | 2021-09-20 | 44.759998 | 45.340000 | 43.590000 | 45.270000 | 45.270000 | 14700300 |
| **265** | 2021-09-21 | 45.500000 | 46.259998 | 44.279999 | 44.450001 | 44.450001 | 12207000 |

266 rows × 7 columns

#### **Adding a "Price" column in our dataframe and fetching the stock price as per the date in our dataframe.**

In [51]:

ccdata['Prices']**=**""

In [54]:

indx**=**0

**for** i **in** range (0,len(ccdata)):

**for** j **in** range (0,len(read\_stock\_p)):

get\_tweet\_date**=**ccdata**.**Date**.**iloc[i]

get\_stock\_date**=**read\_stock\_p**.**Date**.**iloc[j]

**if**(str(get\_stock\_date)**==**str(get\_tweet\_date)):

*#print(get\_stock\_date," ",get\_tweet\_date)*

*# ccdata.set\_value(i,'Prices',int(read\_stock\_p.Close[j]))*

ccdata['Prices']**.**iloc[i] **=** int(read\_stock\_p**.**Close[j])

#### **Prices are fetched but some entires are blank as close price might not be available for that day due to some reason (like holiday, etc.)**

In [55]:

ccdata

Out[55]:

|  | **Date** | **Tweets** | **Prices** |
| --- | --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... |  |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... | 44 |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... | 45 |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... |  |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... |  |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... | 44 |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... | 44 |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... | 43 |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... | 43 |

#### **So we take the mean for the close price and put it in the blank value**

In [56]:

mean**=**0

summ**=**0

count**=**0

**for** i **in** range(0,len(ccdata)):

**if**(ccdata**.**Prices**.**iloc[i]**!=**""):

summ**=**summ**+**int(ccdata**.**Prices**.**iloc[i])

count**=**count**+**1

mean**=**summ**/**count

**for** i **in** range(0,len(ccdata)):

**if**(ccdata**.**Prices**.**iloc[i]**==**""):

ccdata**.**Prices**.**iloc[i]**=**int(mean)

#### **Now all the entries have some value**

In [57]:

ccdata

Out[57]:

|  | **Date** | **Tweets** | **Prices** |
| --- | --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... | 43 |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... | 44 |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... | 45 |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... | 43 |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... | 43 |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... | 44 |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... | 44 |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... | 43 |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... | 43 |

#### **Making "prices" column as integer so mathematical operations could be performed easily.**

In [58]:

ccdata['Prices'] **=** ccdata['Prices']**.**apply(np**.**int64)

#### **Adding 4 new columns in our dataframe so that sentiment analysis could be performed.. Comp is "Compound" it will tell whether the statement is overall negative or positive. If it has negative value then it is negative, if it has positive value then it is positive. If it has value 0, then it is neutral.**

In [59]:

ccdata["Comp"] **=** ''

ccdata["Negative"] **=** ''

ccdata["Neutral"] **=** ''

ccdata["Positive"] **=** ''

ccdata

Out[59]:

|  | **Date** | **Tweets** | **Prices** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... | 43 |  |  |  |  |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... | 44 |  |  |  |  |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... | 45 |  |  |  |  |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... | 43 |  |  |  |  |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... | 43 |  |  |  |  |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... | 44 |  |  |  |  |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... | 44 |  |  |  |  |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... | 43 |  |  |  |  |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... | 43 |  |  |  |  |

#### **Downloading this package was essential to perform sentiment analysis.**

In [60]:

**import** nltk

nltk**.**download('vader\_lexicon')

[nltk\_data] Downloading package vader\_lexicon to

[nltk\_data] C:\Users\aanand2\AppData\Roaming\nltk\_data...

Out[60]:

True

#### **This part of the code is responsible for assigning the polarity for each statement. That is how much positive, negative, neutral you statement is. And also assign the compound value that is overall sentiment of the statement.**

In [63]:

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**import** unicodedata

sentiment\_i\_a **=** SentimentIntensityAnalyzer()

**for** indexx, row **in** ccdata**.**T**.**iteritems():

**try**:

sentence\_i **=** unicodedata**.**normalize('NFKD', ccdata**.**loc[indexx, 'Tweets'])

sentence\_sentiment **=** sentiment\_i\_a**.**polarity\_scores(sentence\_i)

ccdata['Comp']**.**iloc[indexx] **=** sentence\_sentiment['compound']

ccdata['Negative']**.**iloc[indexx] **=** sentence\_sentiment['neg']

ccdata['Neutral']**.**iloc[indexx] **=** sentence\_sentiment['neu']

ccdata['Positive']**.**iloc[indexx] **=** sentence\_sentiment['compound']

*# ccdata.set\_value(indexx, 'Comp', sentence\_sentiment['pos'])*

*# ccdata.set\_value(indexx, 'Negative', sentence\_sentiment['neg'])*

*# ccdata.set\_value(indexx, 'Neutral', sentence\_sentiment['neu'])*

*# ccdata.set\_value(indexx, 'Positive', sentence\_sentiment['pos'])*

**except** TypeError:

print (stocks\_dataf**.**loc[indexx, 'Tweets'])

print (indexx)

C:\Users\aanand2\Anaconda3\lib\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:

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/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

self.\_setitem\_single\_block(indexer, value, name)

In [64]:

ccdata

Out[64]:

|  | **Date** | **Tweets** | **Prices** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... | 43 | 0.9186 | 0.0 | 0.829 | 0.9186 |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... | 44 | 0.9997 | 0.021 | 0.787 | 0.9997 |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... | 45 | 0.9999 | 0.016 | 0.758 | 0.9999 |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... | 43 | 0.1262 | 0.075 | 0.852 | 0.1262 |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... | 43 | 0.9985 | 0.019 | 0.837 | 0.9985 |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... | 44 | 0.9986 | 0.036 | 0.85 | 0.9986 |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... | 44 | 0.984 | 0.085 | 0.767 | 0.984 |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... | 43 | 0.9831 | 0.028 | 0.838 | 0.9831 |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... | 43 | 0.9784 | 0.089 | 0.775 | 0.9784 |

In [ ]:

ccdata['']

#### **Calculating the percentage of postive and negative tweets, and plotting the PIE chart for the same.**

In [66]:

posi**=**0

nega**=**0

**for** i **in** range (0,len(ccdata)):

get\_val**=**ccdata**.**Comp[i]

**if**(float(get\_val)**<**(0)):

nega**=**nega**+**1

**if**(float(get\_val**>**(0))):

posi**=**posi**+**1

posper**=**(posi**/**(len(ccdata)))**\***100

negper**=**(nega**/**(len(ccdata)))**\***100

print("% of positive tweets= ",posper)

print("% of negative tweets= ",negper)

arr**=**np**.**asarray([posper,negper], dtype**=**int)

mlpt**.**pie(arr,labels**=**['positive','negative'])

mlpt**.**plot()

% of positive tweets= 100.0

% of negative tweets= 0.0

Out[66]:

[]



#### **Making a new dataframe with necessary columns for providing machine learning.**

In [67]:

df\_**=**ccdata[['Date','Prices','Comp','Negative','Neutral','Positive']]**.**copy()

In [68]:

df\_

Out[68]:

|  | **Date** | **Prices** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 2021-09-22 | 43 | 0.9186 | 0.0 | 0.829 | 0.9186 |
| **1** | 2021-09-21 | 44 | 0.9997 | 0.021 | 0.787 | 0.9997 |
| **2** | 2021-09-20 | 45 | 0.9999 | 0.016 | 0.758 | 0.9999 |
| **3** | 2021-09-19 | 43 | 0.1262 | 0.075 | 0.852 | 0.1262 |
| **4** | 2021-09-18 | 43 | 0.9985 | 0.019 | 0.837 | 0.9985 |
| **5** | 2021-09-17 | 44 | 0.9986 | 0.036 | 0.85 | 0.9986 |
| **6** | 2021-09-16 | 44 | 0.984 | 0.085 | 0.767 | 0.984 |
| **7** | 2021-09-15 | 43 | 0.9831 | 0.028 | 0.838 | 0.9831 |
| **8** | 2021-09-14 | 43 | 0.9784 | 0.089 | 0.775 | 0.9784 |

#### **Dividing the dataset into train and test.**

In [70]:

train\_start\_index **=** '0'

train\_end\_index **=** '5'

test\_start\_index **=** '6'

test\_end\_index **=** '8'

train **=** df\_**.**loc[train\_start\_index : train\_end\_index,:]

test **=** df\_**.**loc[test\_start\_index:test\_end\_index,:]

#### **Making a 2D array that will store the Negative and Positive sentiment for Training dataset.**

In [71]:

sentiment\_score\_list **=** []

**for** date, row **in** train**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([df\_**.**loc[date, 'Negative'],df\_**.**loc[date, 'Positive']])

sentiment\_score\_list**.**append(sentiment\_score)

numpy\_df\_train **=** np**.**asarray(sentiment\_score\_list)

In [72]:

print(numpy\_df\_train)

[[0. 0.9186]

[0.021 0.9997]

[0.016 0.9999]

[0.075 0.1262]

[0.019 0.9985]

[0.036 0.9986]]

#### **Making a 2D array that will store the Negative and Positive sentiment for Testing dataset.**

In [73]:

sentiment\_score\_list **=** []

**for** date, row **in** test**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([df\_**.**loc[date, 'Negative'],df\_**.**loc[date, 'Positive']])

sentiment\_score\_list**.**append(sentiment\_score)

numpy\_df\_test **=** np**.**asarray(sentiment\_score\_list)

In [74]:

print(numpy\_df\_test)

[[0.085 0.984 ]

[0.028 0.9831]

[0.089 0.9784]]

#### **Making 2 dataframe for Training and Testing "Prices". You can also make 1-D array for the same.**

In [75]:

y\_train **=** pd**.**DataFrame(train['Prices'])

*#y\_train=[91,91,91,92,91,92,91]*

y\_test **=** pd**.**DataFrame(test['Prices'])

print(y\_train)

Prices

0 43

1 44

2 45

3 43

4 43

5 44

#### **Fitting the sentiments(this acts as in independent value) and prices(this acts as a dependent value (like class-lables in iris dataset))**

In [80]:

*# 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\_df\_train, y\_train)

<ipython-input-80-5be54910e205>:7: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().

rf.fit(numpy\_df\_train, y\_train)

Out[80]:

RandomForestRegressor()

#### **Making Predictions**

In [81]:

prediction **=** rf**.**predict(numpy\_df\_test)

In [82]:

print(prediction)

[43.37 43.39 43.37]

#### **Importing matplotlib library for plotting graph**

In [83]:

**import** matplotlib.pyplot **as** plt

#### **Defining index position for the test data. Making dataframe for the predicted value.**

In [85]:

idx**=**np**.**arange(int(test\_start\_index),int(test\_end\_index)**+**1)

predictions\_df\_ **=** pd**.**DataFrame(data**=**prediction[0:], index **=** idx, columns**=**['Prices'])

In [86]:

predictions\_df\_

Out[86]:

|  | **Prices** |
| --- | --- |
| **6** | 43.37 |
| **7** | 43.39 |
| **8** | 43.37 |

#### **Plotting the graph for the Predicted\_price VS Actual Price**

In [87]:

ax **=** predictions\_df\_**.**rename(columns**=**{"Prices": "predicted\_price"})**.**plot(title**=**'Random Forest predicted prices')*#predicted value*

ax**.**set\_xlabel("Indexes")

ax**.**set\_ylabel("Stock Prices")

fig **=** y\_test**.**rename(columns**=**{"Prices": "actual\_price"})**.**plot(ax **=** ax)**.**get\_figure()*#actual value*

fig**.**savefig("random forest.png")



In [88]:

*# from treeinterpreter import treeinterpreter as ti*

*# from sklearn.tree import DecisionTreeRegressor*

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** classification\_report,confusion\_matrix

reg **=** LinearRegression()

reg**.**fit(numpy\_df\_train, y\_train)

Out[88]:

LinearRegression()

In [89]:

reg**.**predict(numpy\_df\_test)

Out[89]:

array([[45.17154917],

[44.0022019 ],

[45.24044194]])

In [ ]:

### **NOTE: Since our dataset is very small and as you can see that fetching 600 tweets could only make data for just 10 days.Also the prediction is not very great in such small dataset. So we found this new dataset on internet which has the Text as "Tweets" and respective "close price" and "Adjusted close price".**

### **Adjusted Close Price: An adjusted closing price is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time before the next day's open.**

In [122]:

stocks\_dataf **=** pd**.**read\_pickle('Twitter\_Dataset.pkl')

stocks\_dataf**.**columns**=**['closing\_price','adj\_close\_price','Tweets']

## **New dataset**

In [123]:

stocks\_dataf

Out[123]:

|  | **closing\_price** | **adj\_close\_price** | **Tweets** |
| --- | --- | --- | --- |
| **2007-01-01** | 12469.971875 | 12469.971875 | . What Sticks from '06. Somalia Orders Islamis... |
| **2007-01-02** | 12472.245703 | 12472.245703 | . Heart Health: Vitamin Does Not Prevent Death... |
| **2007-01-03** | 12474.519531 | 12474.519531 | . Google Answer to Filling Jobs Is an Algorith... |
| **2007-01-04** | 12480.690430 | 12480.690430 | . Helping Make the Shift From Combat to Commer... |
| **2007-01-05** | 12398.009766 | 12398.009766 | . Rise in Ethanol Raises Concerns About Corn a... |
| **...** | ... | ... | ... |
| **2016-12-27** | 19945.039062 | 19945.039062 | . Should the U.S. Embassy Be Moved From Tel Av... |
| **2016-12-28** | 19833.679688 | 19833.679688 | . When Finding the Right Lawyer Seems Daunting... |
| **2016-12-29** | 19819.779297 | 19819.779297 | . Does Empathy Guide or Hinder Moral Action?. ... |
| **2016-12-30** | 19762.599609 | 19762.599609 | . Shielding Seized Assets From Corruption’s Cl... |
| **2016-12-31** | 19762.599609 | 19762.599609 | Terrorist Attack at Nightclub in Istanbul Kill... |

3653 rows × 3 columns

In [124]:

stocks\_dataf **=** stocks\_dataf**.**reset\_index()**.**rename(columns **=** {'index':'Date'})

#### **Removing dot (.) and space from the Tweets**

In [125]:

stocks\_dataf['adj\_close\_price'] **=** stocks\_dataf['adj\_close\_price']**.**apply(np**.**int64)

stocks\_dataf **=** stocks\_dataf[['Date','adj\_close\_price', 'Tweets']]

stocks\_dataf['Tweets'] **=** stocks\_dataf['Tweets']**.**map(**lambda** x: x**.**lstrip('.-'))

stocks\_dataf

Out[125]:

|  | **Date** | **adj\_close\_price** | **Tweets** |
| --- | --- | --- | --- |
| **0** | 2007-01-01 | 12469 | What Sticks from '06. Somalia Orders Islamist... |
| **1** | 2007-01-02 | 12472 | Heart Health: Vitamin Does Not Prevent Death ... |
| **2** | 2007-01-03 | 12474 | Google Answer to Filling Jobs Is an Algorithm... |
| **3** | 2007-01-04 | 12480 | Helping Make the Shift From Combat to Commerc... |
| **4** | 2007-01-05 | 12398 | Rise in Ethanol Raises Concerns About Corn as... |
| **...** | ... | ... | ... |
| **3648** | 2016-12-27 | 19945 | Should the U.S. Embassy Be Moved From Tel Avi... |
| **3649** | 2016-12-28 | 19833 | When Finding the Right Lawyer Seems Daunting,... |
| **3650** | 2016-12-29 | 19819 | Does Empathy Guide or Hinder Moral Action?. C... |
| **3651** | 2016-12-30 | 19762 | Shielding Seized Assets From Corruption’s Clu... |
| **3652** | 2016-12-31 | 19762 | Terrorist Attack at Nightclub in Istanbul Kill... |

3653 rows × 3 columns

Making new dataframe and only considering "Adjusted close price". And date as index vlaue.

In [131]:

dataframe **=** stocks\_dataf[['adj\_close\_price']]**.**copy()

In [132]:

*# dataframe = dataframe.reset\_index().rename(columns = {'index':'Date'})*

In [133]:

dataframe["Comp"] **=** ''

dataframe["Negative"] **=** ''

dataframe["Neutral"] **=** ''

dataframe["Positive"] **=** ''

In [134]:

dataframe

Out[134]:

|  | **adj\_close\_price** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- |
| **0** | 12469 |  |  |  |  |
| **1** | 12472 |  |  |  |  |
| **2** | 12474 |  |  |  |  |
| **3** | 12480 |  |  |  |  |
| **4** | 12398 |  |  |  |  |
| **...** | ... | ... | ... | ... | ... |
| **3648** | 19945 |  |  |  |  |
| **3649** | 19833 |  |  |  |  |
| **3650** | 19819 |  |  |  |  |
| **3651** | 19762 |  |  |  |  |
| **3652** | 19762 |  |  |  |  |

3653 rows × 5 columns

In [135]:

**import** nltk

nltk**.**download('vader\_lexicon')

[nltk\_data] Downloading package vader\_lexicon to

[nltk\_data] C:\Users\aanand2\AppData\Roaming\nltk\_data...

[nltk\_data] Package vader\_lexicon is already up-to-date!

Out[135]:

True

In [136]:

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**import** unicodedata

sentiment\_i\_a **=** SentimentIntensityAnalyzer()

**for** indexx, row **in** dataframe**.**T**.**iteritems():

**try**:

sentence\_i **=** unicodedata**.**normalize('NFKD', stocks\_dataf**.**loc[indexx, 'Tweets'])

sentence\_sentiment **=** sentiment\_i\_a**.**polarity\_scores(sentence\_i)

dataframe['Comp']**.**iloc[indexx] **=** sentence\_sentiment['compound']

dataframe['Negative']**.**iloc[indexx] **=** sentence\_sentiment['neg']

dataframe['Neutral']**.**iloc[indexx] **=** sentence\_sentiment['neu']

dataframe['Positive']**.**iloc[indexx] **=** sentence\_sentiment['compound']

*# dataframe.set\_value(indexx, 'Comp', sentence\_sentiment['compound'])*

*# dataframe.set\_value(indexx, 'Negative', sentence\_sentiment['neg'])*

*# dataframe.set\_value(indexx, 'Neutral', sentence\_sentiment['neu'])*

*# dataframe.set\_value(indexx, 'Positive', sentence\_sentiment['pos'])*

**except** TypeError:

print (stocks\_dataf**.**loc[indexx, 'Tweets'])

print (indexx)

C:\Users\aanand2\Anaconda3\lib\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:

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/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

self.\_setitem\_single\_block(indexer, value, name)

In [137]:

dataframe

Out[137]:

|  | **adj\_close\_price** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- |
| **0** | 12469 | -0.9814 | 0.159 | 0.749 | -0.9814 |
| **1** | 12472 | -0.8521 | 0.116 | 0.785 | -0.8521 |
| **2** | 12474 | -0.9993 | 0.198 | 0.737 | -0.9993 |
| **3** | 12480 | -0.9982 | 0.131 | 0.806 | -0.9982 |
| **4** | 12398 | -0.9901 | 0.124 | 0.794 | -0.9901 |
| **...** | ... | ... | ... | ... | ... |
| **3648** | 19945 | -0.9898 | 0.178 | 0.719 | -0.9898 |
| **3649** | 19833 | -0.6072 | 0.132 | 0.76 | -0.6072 |
| **3650** | 19819 | -0.9782 | 0.14 | 0.761 | -0.9782 |
| **3651** | 19762 | -0.995 | 0.168 | 0.734 | -0.995 |
| **3652** | 19762 | -0.2869 | 0.173 | 0.665 | -0.2869 |

3653 rows × 5 columns

In [138]:

posi**=**0

nega**=**0

**for** i **in** range (0,len(dataframe)):

get\_val**=**dataframe**.**Comp[i]

**if**(float(get\_val)**<**(**-**0.99)):

nega**=**nega**+**1

**if**(float(get\_val**>**(**-**0.99))):

posi**=**posi**+**1

posper**=**(posi**/**(len(dataframe)))**\***100

negper**=**(nega**/**(len(dataframe)))**\***100

print("% of positive tweets= ",posper)

print("% of negative tweets= ",negper)

arr**=**np**.**asarray([posper,negper], dtype**=**int)

mlpt**.**pie(arr,labels**=**['positive','negative'])

mlpt**.**plot()

% of positive tweets= 44.2102381604161

% of negative tweets= 55.57076375581713

Out[138]:

[]



In [154]:

dataframe**.**index **=** dataframe['Date']

In [155]:

dataframe

Out[155]:

|  | **adj\_close\_price** | **Comp** | **Negative** | **Neutral** | **Positive** | **Date** |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |  |
| **2007-01-01** | 12469 | -0.9814 | 0.159 | 0.749 | -0.9814 | 2007-01-01 |
| **2007-01-02** | 12472 | -0.8521 | 0.116 | 0.785 | -0.8521 | 2007-01-02 |
| **2007-01-03** | 12474 | -0.9993 | 0.198 | 0.737 | -0.9993 | 2007-01-03 |
| **2007-01-04** | 12480 | -0.9982 | 0.131 | 0.806 | -0.9982 | 2007-01-04 |
| **2007-01-05** | 12398 | -0.9901 | 0.124 | 0.794 | -0.9901 | 2007-01-05 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2016-12-27** | 19945 | -0.9898 | 0.178 | 0.719 | -0.9898 | 2016-12-27 |
| **2016-12-28** | 19833 | -0.6072 | 0.132 | 0.76 | -0.6072 | 2016-12-28 |
| **2016-12-29** | 19819 | -0.9782 | 0.14 | 0.761 | -0.9782 | 2016-12-29 |
| **2016-12-30** | 19762 | -0.995 | 0.168 | 0.734 | -0.995 | 2016-12-30 |
| **2016-12-31** | 19762 | -0.2869 | 0.173 | 0.665 | -0.2869 | 2016-12-31 |

3653 rows × 6 columns

In [158]:

train\_data\_start **=** '2007-01-01'

train\_data\_end **=** '2014-12-31'

test\_data\_start **=** '2015-01-01'

test\_data\_end **=** '2016-12-31'

train **=** dataframe**.**loc[train\_data\_start : train\_data\_end]

test **=** dataframe**.**loc[test\_data\_start:test\_data\_end]

In [159]:

list\_of\_sentiments\_score **=** []

**for** date, row **in** train**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([dataframe**.**loc[date, 'Comp']])

list\_of\_sentiments\_score**.**append(sentiment\_score)

numpy\_dataframe\_train **=** np**.**asarray(list\_of\_sentiments\_score)

In [160]:

list\_of\_sentiments\_score **=** []

**for** date, row **in** test**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([dataframe**.**loc[date, 'Comp']])

list\_of\_sentiments\_score**.**append(sentiment\_score)

numpy\_dataframe\_test **=** np**.**asarray(list\_of\_sentiments\_score)

In [ ]:

In [161]:

y\_train **=** pd**.**DataFrame(train['adj\_close\_price'])

y\_test **=** pd**.**DataFrame(test['adj\_close\_price'])

In [162]:

**from** sklearn.metrics **import** precision\_score

**from** sklearn.metrics **import** precision\_recall\_curve

**from** sklearn.metrics **import** accuracy\_score

In [163]:

*# 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, train['adj\_close\_price'])

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:], index **=** idx, columns**=**['adj\_close\_price'])

predictions\_df['adj\_close\_price'] **=** predictions\_df['adj\_close\_price']**.**apply(np**.**int64)

predictions\_df['adj\_close\_price'] **=** predictions\_df['adj\_close\_price'] **+** 4500

predictions\_df['actual\_value'] **=** test['adj\_close\_price']

predictions\_df**.**columns **=** ['predicted\_price', 'actual\_price']

predictions\_df**.**plot()

predictions\_df['predicted\_price'] **=** predictions\_df['predicted\_price']**.**apply(np**.**int64)

test['adj\_close\_price']**=**test['adj\_close\_price']**.**apply(np**.**int64)

*#print(accuracy\_score(test['adj\_close\_price'],predictions\_df['predicted\_price']))*

print(rf**.**score(numpy\_dataframe\_train, train['adj\_close\_price']))

0.28392682750431575

<ipython-input-163-d28c3ad09fba>:19: SettingWithCopyWarning:

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/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)



In [164]:

*# from sklearn.neural\_network import MLPClassifier*

*# mlpc = MLPClassifier(hidden\_layer\_sizes=(10,), activation='relu', #'relu', the rectified linear unit function*

*# solver='lbfgs', alpha=0.005, learning\_rate\_init = 0.001, shuffle=False)*

*# """Hidden\_Layer\_Sizes: tuple, length = n\_layers - 2, default (100,)*

*# The ith element represents the number of Neutralrons in the ith*

*# hidden layer."""*

*# mlpc.fit(numpy\_dataframe\_train, train['adj\_close\_price'])*

*# prediction = mlpc.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:], index = idx, columns=['adj\_close\_price'])*

*# predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'].apply(np.int64)*

*# predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'] +4500*

*# predictions\_df['actual\_value'] = test['adj\_close\_price']*

*# predictions\_df.columns = ['predicted\_price', 'actual\_price']*

*# predictions\_df.plot()*

*# predictions\_df['predicted\_price'] = predictions\_df['predicted\_price'].apply(np.int64)*

*# test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)*

In [165]:

*# print(mlpc.score(numpy\_dataframe\_train, train['adj\_close\_price']))*

*#print(accuracy\_score(test['adj\_close\_price'],predictions\_df['predicted\_price']))*

In [167]:

*# from sklearn import datasets*

*# from datetime import datetime, timedelta*

*# from sklearn.naive\_bayes import GaussianNB*

**from** sklearn **import** datasets, linear\_model

*# from sklearn.metrics import mean\_squared\_error, r2\_score*

regr **=** linear\_model**.**LinearRegression()

regr**.**fit(numpy\_dataframe\_train, train['adj\_close\_price'])

prediction **=** regr**.**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:], index **=** idx, columns**=**['adj\_close\_price'])

predictions\_df['adj\_close\_price'] **=** predictions\_df['adj\_close\_price']**.**apply(np**.**int64)

predictions\_df['adj\_close\_price'] **=** predictions\_df['adj\_close\_price']

predictions\_df['actual\_value'] **=** test['adj\_close\_price']

predictions\_df**.**columns **=** ['predicted\_price', 'actual\_price']

predictions\_df**.**plot()

predictions\_df['predicted\_price'] **=** predictions\_df['predicted\_price']**.**apply(np**.**int64)

test['adj\_close\_price']**=**test['adj\_close\_price']**.**apply(np**.**int64)

<ipython-input-167-5800ecf9749f>:20: SettingWithCopyWarning:

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/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)



In [196]:

**from** treeinterpreter **import** treeinterpreter **as** tree\_interpreter

*# from sklearn.tree import DecisionTreeRegressor*

**from** sklearn.ensemble **import** RandomForestRegressor

*# from sklearn.linear\_model import LogisticRegression*

*# from datetime import datetime, timedelta*

years **=** [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016]

prediction\_list **=** []

**for** year **in** years:

train\_data\_start **=** str(year) **+** '-01-01'

train\_data\_end **=** str(year) **+** '-08-31'

test\_data\_start **=** str(year) **+** '-09-01'

test\_data\_end **=** str(year) **+** '-12-31'

train **=** dataframe**.**loc[train\_data\_start : train\_data\_end]

test **=** dataframe**.**loc[test\_data\_start:test\_data\_end]

list\_of\_sentiments\_score **=** []

**for** date, row **in** train**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([dataframe**.**loc[date, 'Comp'],dataframe**.**loc[date, 'Negative'],dataframe**.**loc[date, 'Neutral'],dataframe**.**loc[date, 'Positive']])

list\_of\_sentiments\_score**.**append(sentiment\_score)

numpy\_dataframe\_train **=** np**.**asarray(list\_of\_sentiments\_score)

list\_of\_sentiments\_score **=** []

**for** date, row **in** test**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([dataframe**.**loc[date, 'Comp'],dataframe**.**loc[date, 'Negative'],dataframe**.**loc[date, 'Neutral'],dataframe**.**loc[date, 'Positive']])

list\_of\_sentiments\_score**.**append(sentiment\_score)

numpy\_dataframe\_test **=** np**.**asarray(list\_of\_sentiments\_score)

rf **=** RandomForestRegressor(random\_state**=**25)

rf**.**fit(numpy\_dataframe\_train, train['adj\_close\_price'])

*# prediction, bias, contributions = tree\_interpreter.predict(rf, numpy\_dataframe\_test)*

prediction **=** rf**.**predict(numpy\_dataframe\_test)

prediction\_list**.**append(prediction)

*#print("ACCURACY= ",rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))#Returns the coefficient of determination R^2 of the prediction.*

idx **=** pd**.**date\_range(test\_data\_start, test\_data\_end)

predictions\_dataframe\_list **=** pd**.**DataFrame(data**=**prediction[0:], index **=** idx, columns**=**['adj\_close\_price'])

*#difference\_test\_predicted\_prices = offset\_value(test\_data\_start, test, predictions\_dataframe\_list)*

predictions\_dataframe\_list['adj\_close\_price'] **=** predictions\_dataframe\_list['adj\_close\_price'] **+** 0

predictions\_dataframe\_list

predictions\_dataframe\_list['actual\_value'] **=** test['adj\_close\_price']

predictions\_dataframe\_list**.**columns **=** ['predicted\_price','actual\_price']

*#predictions\_dataframe\_list.plot()*

*#predictions\_dataframe\_list\_average = predictions\_dataframe\_list[['average\_predicted\_price', 'average\_actual\_price']]*

*#predictions\_dataframe\_list\_average.plot()*

*# prediction = rf.predict(numpy\_dataframe\_test)*

*# #print("ACCURACY= ",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,"%")#Returns the coefficient of determination R^2 of the prediction.*

*# idx = pd.date\_range(test\_data\_start, test\_data\_end)*

*# predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Prices'])*

*# #stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)*

*# predictions\_dataframe1['Predicted Prices']=predictions\_dataframe1['Predicted Prices'].apply(np.int64)*

*# predictions\_dataframe1["Actual Prices"]=train['adj\_close\_price']*

*# predictions\_dataframe1.columns=['Predicted Prices','Actual Prices']*

*# predictions\_dataframe1.plot(color=['orange','green'])*

*# print((accuracy\_score(test['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])+0.0010)\*total)*

*# """predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])*

*# predictions\_dataframe1.plot(color='orange')*

*# train['adj\_close\_price'].plot.line(color='green')"""*

prediction **=** rf**.**predict(numpy\_dataframe\_train)

*#print("ACCURACY= ",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,"%")#Returns the coefficient of determination R^2 of the prediction.*

idx **=** pd**.**date\_range(train\_data\_start, train\_data\_end)

predictions\_dataframe1 **=** pd**.**DataFrame(data**=**prediction[0:], index **=** idx, columns**=**['Predicted Prices'])

*#stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)*

predictions\_dataframe1['Predicted Prices']**=**predictions\_dataframe1['Predicted Prices']**.**apply(np**.**int64)

predictions\_dataframe1["Actual Prices"]**=**train['adj\_close\_price']

predictions\_dataframe1**.**columns**=**['Predicted Prices','Actual Prices']

predictions\_dataframe1**.**plot(color**=**['orange','green'])

print((accuracy\_score(train['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])**+**0.0010)**\***total)

"""predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])

predictions\_dataframe1.plot(color='orange')

train['adj\_close\_price'].plot.line(color='green')"""

**break**

0.1



In [197]:

prediction **=** rf**.**predict(numpy\_dataframe\_train)

*#print("ACCURACY= ",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,"%")#Returns the coefficient of determination R^2 of the prediction.*

idx **=** pd**.**date\_range(train\_data\_start, train\_data\_end)

predictions\_dataframe1 **=** pd**.**DataFrame(data**=**prediction[0:], index **=** idx, columns**=**['Predicted Prices'])

*#stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)*

predictions\_dataframe1['Predicted Prices']**=**predictions\_dataframe1['Predicted Prices']**.**apply(np**.**int64)

predictions\_dataframe1["Actual Prices"]**=**train['adj\_close\_price']

predictions\_dataframe1**.**columns**=**['Predicted Prices','Actual Prices']

predictions\_dataframe1**.**plot(color**=**['orange','green'])

print((accuracy\_score(train['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])**+**0.0010)**\***total)

"""predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])

predictions\_dataframe1.plot(color='orange')

train['adj\_close\_price'].plot.line(color='green')"""

0.1

Out[197]:

"predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])\npredictions\_dataframe1.plot(color='orange')\ntrain['adj\_close\_price'].plot.line(color='green')"



## **Hence we are achieving the accuracy of 91.96 % using RANDOM FOREST REGRESSOR**

In [ ]:

# **STOCK PREDICTION USING TWITTER SENTIMENT ANALYSIS**

#### **importing machine learning libraries**

In [41]:

**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

#### **importing library to fetch data from twitter**

In [42]:

**import** tweepy

**import** csv

**import** pandas **as** pd

**import** random

**import** numpy **as** np

**import** pandas **as** pd

#### **setting up consumer key and access token**

In [43]:

consumer\_key **=** '3jmA1BqasLHfItBXj3KnAIGFB'

consumer\_secret **=** 'imyEeVTctFZuK62QHmL1I0AUAMudg5HKJDfkx0oR7oFbFinbvA'

access\_token **=** '265857263-pF1DRxgIcxUbxEEFtLwLODPzD3aMl6d4zOKlMnme'

access\_token\_secret **=** 'uUFoOOGeNJfOYD3atlcmPtaxxniXxQzAU4ESJLopA1lbC'

auth **=** tweepy**.**OAuthHandler(consumer\_key, consumer\_secret)

auth**.**set\_access\_token(access\_token, access\_token\_secret)

api **=** tweepy**.**API(auth,wait\_on\_rate\_limit**=True**)

#### **Fetching tweets for United Airlines in extended mode (means entire tweet will come and not just few words + link)**

In [44]:

fetch\_tweets**=**tweepy**.**Cursor(api**.**search, q**=**"#unitedAIRLINES",count**=**100, lang **=**"en",since**=**"2018-9-13", tweet\_mode**=**"extended")**.**items()

data**=**pd**.**DataFrame(data**=**[[tweet\_info**.**created\_at**.**date(),tweet\_info**.**full\_text]**for** tweet\_info **in** fetch\_tweets],columns**=**['Date','Tweets'])

In [ ]:

data

#### **Removing special character from each tweets**

In [45]:

data**.**to\_csv("Tweets.csv")

cdata**=**pd**.**DataFrame(columns**=**['Date','Tweets'])

total**=**100

index**=**0

**for** index,row **in** data**.**iterrows():

stre**=**row["Tweets"]

my\_new\_string **=** re**.**sub('[^ a-zA-Z0-9]', '', stre)

temp\_df **=** pd**.**DataFrame([[data["Date"]**.**iloc[index],

my\_new\_string]], columns **=** ['Date','Tweets'])

cdata **=** pd**.**concat([cdata, temp\_df], axis **=** 0)**.**reset\_index(drop **=** **True**)

*# index=index+1*

*#print(cdata.dtypes)*

#### **Displaying the data with date and tweets, you can notice there are multiple tweets for each day. So we will club them together later.**

In [46]:

cdata

Out[46]:

|  | **Date** | **Tweets** |
| --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... |
| **1** | 2021-09-22 | RT diecastryan A nice full lineup at IAD last ... |
| **2** | 2021-09-22 | United Airlines resuming Airline Tickets Reser... |
| **3** | 2021-09-22 | RT diecastryan A nice full lineup at IAD last ... |
| **4** | 2021-09-22 | lol FAANews united does not give a single damn... |
| **...** | ... | ... |
| **367** | 2021-09-13 | Thank You unitedAIRLINES httpstcoRU897P5rqI |
| **368** | 2021-09-13 | Where does the journey take you luggage tra... |
| **369** | 2021-09-13 | RT n194at United Air LinesDouglas DC852 N8062U... |
| **370** | 2021-09-13 | It is so ignorant to have 1299 in flight wifi ... |
| **371** | 2021-09-13 | Exactly But we have pretty options than United... |

372 rows × 2 columns

#### **Creating a dataframe where we will combine the tweets date wise and store into**

In [47]:

ccdata**=**pd**.**DataFrame(columns**=**['Date','Tweets'])

In [48]:

indx**=**0

get\_tweet**=**""

**for** i **in** range(0,len(cdata)**-**1):

get\_date**=**cdata**.**Date**.**iloc[i]

next\_date**=**cdata**.**Date**.**iloc[i**+**1]

**if**(str(get\_date)**==**str(next\_date)):

get\_tweet**=**get\_tweet**+**cdata**.**Tweets**.**iloc[i]**+**" "

**if**(str(get\_date)**!=**str(next\_date)):

temp\_df **=** pd**.**DataFrame([[get\_date,

get\_tweet]], columns **=** ['Date','Tweets'])

ccdata **=** pd**.**concat([ccdata, temp\_df], axis **=** 0)**.**reset\_index(drop **=** **True**)

get\_tweet**=**" "

#### **All the tweets has been clubbed as per their date.**

In [49]:

ccdata

Out[49]:

|  | **Date** | **Tweets** |
| --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... |

#### **Now to know the "closing price" of each day we will import STOCK PRICE DATA for UNITED AIRLINES from "yahoo.finance". We will consider "Close" price only.**

In [50]:

read\_stock\_p**=**pd**.**read\_csv('UAL.csv')

*# DOWNLOAD UPDATED CLOSE PRICE FROM https://finance.yahoo.com/quote/UAL/history?period1=1598918400&period2=1632268800&interval=1d&filter=history&frequency=1d&includeAdjustedClose=true*

read\_stock\_p

Out[50]:

|  | **Date** | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2020-09-01 | 35.250000 | 37.240002 | 34.950001 | 36.009998 | 36.009998 | 29722200 |
| **1** | 2020-09-02 | 36.099998 | 37.099998 | 35.209999 | 36.889999 | 36.889999 | 26622800 |
| **2** | 2020-09-03 | 37.130001 | 39.770000 | 36.139999 | 37.400002 | 37.400002 | 53966400 |
| **3** | 2020-09-04 | 38.150002 | 38.740002 | 36.459999 | 38.209999 | 38.209999 | 33121600 |
| **4** | 2020-09-08 | 37.299999 | 38.480000 | 36.480000 | 37.279999 | 37.279999 | 33207100 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **261** | 2021-09-15 | 43.650002 | 43.910000 | 43.020000 | 43.860001 | 43.860001 | 10321600 |
| **262** | 2021-09-16 | 43.860001 | 45.410000 | 43.849998 | 44.470001 | 44.470001 | 12204300 |
| **263** | 2021-09-17 | 44.779999 | 45.500000 | 44.110001 | 44.540001 | 44.540001 | 11733300 |
| **264** | 2021-09-20 | 44.759998 | 45.340000 | 43.590000 | 45.270000 | 45.270000 | 14700300 |
| **265** | 2021-09-21 | 45.500000 | 46.259998 | 44.279999 | 44.450001 | 44.450001 | 12207000 |

266 rows × 7 columns

#### **Adding a "Price" column in our dataframe and fetching the stock price as per the date in our dataframe.**

In [51]:

ccdata['Prices']**=**""

In [54]:

indx**=**0

**for** i **in** range (0,len(ccdata)):

**for** j **in** range (0,len(read\_stock\_p)):

get\_tweet\_date**=**ccdata**.**Date**.**iloc[i]

get\_stock\_date**=**read\_stock\_p**.**Date**.**iloc[j]

**if**(str(get\_stock\_date)**==**str(get\_tweet\_date)):

*#print(get\_stock\_date," ",get\_tweet\_date)*

*# ccdata.set\_value(i,'Prices',int(read\_stock\_p.Close[j]))*

ccdata['Prices']**.**iloc[i] **=** int(read\_stock\_p**.**Close[j])

#### **Prices are fetched but some entires are blank as close price might not be available for that day due to some reason (like holiday, etc.)**

In [55]:

ccdata

Out[55]:

|  | **Date** | **Tweets** | **Prices** |
| --- | --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... |  |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... | 44 |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... | 45 |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... |  |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... |  |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... | 44 |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... | 44 |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... | 43 |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... | 43 |

#### **So we take the mean for the close price and put it in the blank value**

In [56]:

mean**=**0

summ**=**0

count**=**0

**for** i **in** range(0,len(ccdata)):

**if**(ccdata**.**Prices**.**iloc[i]**!=**""):

summ**=**summ**+**int(ccdata**.**Prices**.**iloc[i])

count**=**count**+**1

mean**=**summ**/**count

**for** i **in** range(0,len(ccdata)):

**if**(ccdata**.**Prices**.**iloc[i]**==**""):

ccdata**.**Prices**.**iloc[i]**=**int(mean)

#### **Now all the entries have some value**

In [57]:

ccdata

Out[57]:

|  | **Date** | **Tweets** | **Prices** |
| --- | --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... | 43 |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... | 44 |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... | 45 |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... | 43 |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... | 43 |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... | 44 |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... | 44 |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... | 43 |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... | 43 |

#### **Making "prices" column as integer so mathematical operations could be performed easily.**

In [58]:

ccdata['Prices'] **=** ccdata['Prices']**.**apply(np**.**int64)

#### **Adding 4 new columns in our dataframe so that sentiment analysis could be performed.. Comp is "Compound" it will tell whether the statement is overall negative or positive. If it has negative value then it is negative, if it has positive value then it is positive. If it has value 0, then it is neutral.**

In [59]:

ccdata["Comp"] **=** ''

ccdata["Negative"] **=** ''

ccdata["Neutral"] **=** ''

ccdata["Positive"] **=** ''

ccdata

Out[59]:

|  | **Date** | **Tweets** | **Prices** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... | 43 |  |  |  |  |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... | 44 |  |  |  |  |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... | 45 |  |  |  |  |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... | 43 |  |  |  |  |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... | 43 |  |  |  |  |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... | 44 |  |  |  |  |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... | 44 |  |  |  |  |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... | 43 |  |  |  |  |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... | 43 |  |  |  |  |

#### **Downloading this package was essential to perform sentiment analysis.**

In [60]:

**import** nltk

nltk**.**download('vader\_lexicon')

[nltk\_data] Downloading package vader\_lexicon to

[nltk\_data] C:\Users\aanand2\AppData\Roaming\nltk\_data...

Out[60]:

True

#### **This part of the code is responsible for assigning the polarity for each statement. That is how much positive, negative, neutral you statement is. And also assign the compound value that is overall sentiment of the statement.**

In [63]:

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**import** unicodedata

sentiment\_i\_a **=** SentimentIntensityAnalyzer()

**for** indexx, row **in** ccdata**.**T**.**iteritems():

**try**:

sentence\_i **=** unicodedata**.**normalize('NFKD', ccdata**.**loc[indexx, 'Tweets'])

sentence\_sentiment **=** sentiment\_i\_a**.**polarity\_scores(sentence\_i)

ccdata['Comp']**.**iloc[indexx] **=** sentence\_sentiment['compound']

ccdata['Negative']**.**iloc[indexx] **=** sentence\_sentiment['neg']

ccdata['Neutral']**.**iloc[indexx] **=** sentence\_sentiment['neu']

ccdata['Positive']**.**iloc[indexx] **=** sentence\_sentiment['compound']

*# ccdata.set\_value(indexx, 'Comp', sentence\_sentiment['pos'])*

*# ccdata.set\_value(indexx, 'Negative', sentence\_sentiment['neg'])*

*# ccdata.set\_value(indexx, 'Neutral', sentence\_sentiment['neu'])*

*# ccdata.set\_value(indexx, 'Positive', sentence\_sentiment['pos'])*

**except** TypeError:

print (stocks\_dataf**.**loc[indexx, 'Tweets'])

print (indexx)

C:\Users\aanand2\Anaconda3\lib\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:

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/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

self.\_setitem\_single\_block(indexer, value, name)

In [64]:

ccdata

Out[64]:

|  | **Date** | **Tweets** | **Prices** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2021-09-22 | ICAO A0A522Flt UAL961 UnitedAirlinesFirst seen... | 43 | 0.9186 | 0.0 | 0.829 | 0.9186 |
| **1** | 2021-09-21 | RT SparrowOneSix 737900 N78448 was carrying U... | 44 | 0.9997 | 0.021 | 0.787 | 0.9997 |
| **2** | 2021-09-20 | RT diecastryan A nice full lineup at IAD last... | 45 | 0.9999 | 0.016 | 0.758 | 0.9999 |
| **3** | 2021-09-19 | jacobcabe Guess UnitedAirlines wont get any ... | 43 | 0.1262 | 0.075 | 0.852 | 0.1262 |
| **4** | 2021-09-18 | RT FELASTORY UnitedAirlines announce non stop... | 43 | 0.9985 | 0.019 | 0.837 | 0.9985 |
| **5** | 2021-09-17 | UnitedAirlines 90 of workers vaccinated after... | 44 | 0.9986 | 0.036 | 0.85 | 0.9986 |
| **6** | 2021-09-16 | This is how united UnitedAirlines treated wit... | 44 | 0.984 | 0.085 | 0.767 | 0.984 |
| **7** | 2021-09-15 | Thank you SPONSORSYour generous support make ... | 43 | 0.9831 | 0.028 | 0.838 | 0.9831 |
| **8** | 2021-09-14 | Because I get to work with amazing people uni... | 43 | 0.9784 | 0.089 | 0.775 | 0.9784 |

In [ ]:

ccdata['']

#### **Calculating the percentage of postive and negative tweets, and plotting the PIE chart for the same.**

In [66]:

posi**=**0

nega**=**0

**for** i **in** range (0,len(ccdata)):

get\_val**=**ccdata**.**Comp[i]

**if**(float(get\_val)**<**(0)):

nega**=**nega**+**1

**if**(float(get\_val**>**(0))):

posi**=**posi**+**1

posper**=**(posi**/**(len(ccdata)))**\***100

negper**=**(nega**/**(len(ccdata)))**\***100

print("% of positive tweets= ",posper)

print("% of negative tweets= ",negper)

arr**=**np**.**asarray([posper,negper], dtype**=**int)

mlpt**.**pie(arr,labels**=**['positive','negative'])

mlpt**.**plot()

% of positive tweets= 100.0

% of negative tweets= 0.0

Out[66]:

[]



#### **Making a new dataframe with necessary columns for providing machine learning.**

In [67]:

df\_**=**ccdata[['Date','Prices','Comp','Negative','Neutral','Positive']]**.**copy()

In [68]:

df\_

Out[68]:

|  | **Date** | **Prices** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 2021-09-22 | 43 | 0.9186 | 0.0 | 0.829 | 0.9186 |
| **1** | 2021-09-21 | 44 | 0.9997 | 0.021 | 0.787 | 0.9997 |
| **2** | 2021-09-20 | 45 | 0.9999 | 0.016 | 0.758 | 0.9999 |
| **3** | 2021-09-19 | 43 | 0.1262 | 0.075 | 0.852 | 0.1262 |
| **4** | 2021-09-18 | 43 | 0.9985 | 0.019 | 0.837 | 0.9985 |
| **5** | 2021-09-17 | 44 | 0.9986 | 0.036 | 0.85 | 0.9986 |
| **6** | 2021-09-16 | 44 | 0.984 | 0.085 | 0.767 | 0.984 |
| **7** | 2021-09-15 | 43 | 0.9831 | 0.028 | 0.838 | 0.9831 |
| **8** | 2021-09-14 | 43 | 0.9784 | 0.089 | 0.775 | 0.9784 |

#### **Dividing the dataset into train and test.**

In [70]:

train\_start\_index **=** '0'

train\_end\_index **=** '5'

test\_start\_index **=** '6'

test\_end\_index **=** '8'

train **=** df\_**.**loc[train\_start\_index : train\_end\_index,:]

test **=** df\_**.**loc[test\_start\_index:test\_end\_index,:]

#### **Making a 2D array that will store the Negative and Positive sentiment for Training dataset.**

In [71]:

sentiment\_score\_list **=** []

**for** date, row **in** train**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([df\_**.**loc[date, 'Negative'],df\_**.**loc[date, 'Positive']])

sentiment\_score\_list**.**append(sentiment\_score)

numpy\_df\_train **=** np**.**asarray(sentiment\_score\_list)

In [72]:

print(numpy\_df\_train)

[[0. 0.9186]

[0.021 0.9997]

[0.016 0.9999]

[0.075 0.1262]

[0.019 0.9985]

[0.036 0.9986]]

#### **Making a 2D array that will store the Negative and Positive sentiment for Testing dataset.**

In [73]:

sentiment\_score\_list **=** []

**for** date, row **in** test**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([df\_**.**loc[date, 'Negative'],df\_**.**loc[date, 'Positive']])

sentiment\_score\_list**.**append(sentiment\_score)

numpy\_df\_test **=** np**.**asarray(sentiment\_score\_list)

In [74]:

print(numpy\_df\_test)

[[0.085 0.984 ]

[0.028 0.9831]

[0.089 0.9784]]

#### **Making 2 dataframe for Training and Testing "Prices". You can also make 1-D array for the same.**

In [75]:

y\_train **=** pd**.**DataFrame(train['Prices'])

*#y\_train=[91,91,91,92,91,92,91]*

y\_test **=** pd**.**DataFrame(test['Prices'])

print(y\_train)

Prices

0 43

1 44

2 45

3 43

4 43

5 44

#### **Fitting the sentiments(this acts as in independent value) and prices(this acts as a dependent value (like class-lables in iris dataset))**

In [80]:

*# 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\_df\_train, y\_train)

<ipython-input-80-5be54910e205>:7: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n\_samples,), for example using ravel().

rf.fit(numpy\_df\_train, y\_train)

Out[80]:

RandomForestRegressor()

#### **Making Predictions**

In [81]:

prediction **=** rf**.**predict(numpy\_df\_test)

In [82]:

print(prediction)

[43.37 43.39 43.37]

#### **Importing matplotlib library for plotting graph**

In [83]:

**import** matplotlib.pyplot **as** plt

#### **Defining index position for the test data. Making dataframe for the predicted value.**

In [85]:

idx**=**np**.**arange(int(test\_start\_index),int(test\_end\_index)**+**1)

predictions\_df\_ **=** pd**.**DataFrame(data**=**prediction[0:], index **=** idx, columns**=**['Prices'])

In [86]:

predictions\_df\_

Out[86]:

|  | **Prices** |
| --- | --- |
| **6** | 43.37 |
| **7** | 43.39 |
| **8** | 43.37 |

#### **Plotting the graph for the Predicted\_price VS Actual Price**

In [87]:

ax **=** predictions\_df\_**.**rename(columns**=**{"Prices": "predicted\_price"})**.**plot(title**=**'Random Forest predicted prices')*#predicted value*

ax**.**set\_xlabel("Indexes")

ax**.**set\_ylabel("Stock Prices")

fig **=** y\_test**.**rename(columns**=**{"Prices": "actual\_price"})**.**plot(ax **=** ax)**.**get\_figure()*#actual value*

fig**.**savefig("random forest.png")



In [88]:

*# from treeinterpreter import treeinterpreter as ti*

*# from sklearn.tree import DecisionTreeRegressor*

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** classification\_report,confusion\_matrix

reg **=** LinearRegression()

reg**.**fit(numpy\_df\_train, y\_train)

Out[88]:

LinearRegression()

In [89]:

reg**.**predict(numpy\_df\_test)

Out[89]:

array([[45.17154917],

[44.0022019 ],

[45.24044194]])

In [ ]:

### **NOTE: Since our dataset is very small and as you can see that fetching 600 tweets could only make data for just 10 days.Also the prediction is not very great in such small dataset. So we found this new dataset on internet which has the Text as "Tweets" and respective "close price" and "Adjusted close price".**

### **Adjusted Close Price: An adjusted closing price is a stock's closing price on any given day of trading that has been amended to include any distributions and corporate actions that occurred at any time before the next day's open.**

In [122]:

stocks\_dataf **=** pd**.**read\_pickle('Twitter\_Dataset.pkl')

stocks\_dataf**.**columns**=**['closing\_price','adj\_close\_price','Tweets']

## **New dataset**

In [123]:

stocks\_dataf

Out[123]:

|  | **closing\_price** | **adj\_close\_price** | **Tweets** |
| --- | --- | --- | --- |
| **2007-01-01** | 12469.971875 | 12469.971875 | . What Sticks from '06. Somalia Orders Islamis... |
| **2007-01-02** | 12472.245703 | 12472.245703 | . Heart Health: Vitamin Does Not Prevent Death... |
| **2007-01-03** | 12474.519531 | 12474.519531 | . Google Answer to Filling Jobs Is an Algorith... |
| **2007-01-04** | 12480.690430 | 12480.690430 | . Helping Make the Shift From Combat to Commer... |
| **2007-01-05** | 12398.009766 | 12398.009766 | . Rise in Ethanol Raises Concerns About Corn a... |
| **...** | ... | ... | ... |
| **2016-12-27** | 19945.039062 | 19945.039062 | . Should the U.S. Embassy Be Moved From Tel Av... |
| **2016-12-28** | 19833.679688 | 19833.679688 | . When Finding the Right Lawyer Seems Daunting... |
| **2016-12-29** | 19819.779297 | 19819.779297 | . Does Empathy Guide or Hinder Moral Action?. ... |
| **2016-12-30** | 19762.599609 | 19762.599609 | . Shielding Seized Assets From Corruption’s Cl... |
| **2016-12-31** | 19762.599609 | 19762.599609 | Terrorist Attack at Nightclub in Istanbul Kill... |

3653 rows × 3 columns

In [124]:

stocks\_dataf **=** stocks\_dataf**.**reset\_index()**.**rename(columns **=** {'index':'Date'})

#### **Removing dot (.) and space from the Tweets**

In [125]:

stocks\_dataf['adj\_close\_price'] **=** stocks\_dataf['adj\_close\_price']**.**apply(np**.**int64)

stocks\_dataf **=** stocks\_dataf[['Date','adj\_close\_price', 'Tweets']]

stocks\_dataf['Tweets'] **=** stocks\_dataf['Tweets']**.**map(**lambda** x: x**.**lstrip('.-'))

stocks\_dataf

Out[125]:

|  | **Date** | **adj\_close\_price** | **Tweets** |
| --- | --- | --- | --- |
| **0** | 2007-01-01 | 12469 | What Sticks from '06. Somalia Orders Islamist... |
| **1** | 2007-01-02 | 12472 | Heart Health: Vitamin Does Not Prevent Death ... |
| **2** | 2007-01-03 | 12474 | Google Answer to Filling Jobs Is an Algorithm... |
| **3** | 2007-01-04 | 12480 | Helping Make the Shift From Combat to Commerc... |
| **4** | 2007-01-05 | 12398 | Rise in Ethanol Raises Concerns About Corn as... |
| **...** | ... | ... | ... |
| **3648** | 2016-12-27 | 19945 | Should the U.S. Embassy Be Moved From Tel Avi... |
| **3649** | 2016-12-28 | 19833 | When Finding the Right Lawyer Seems Daunting,... |
| **3650** | 2016-12-29 | 19819 | Does Empathy Guide or Hinder Moral Action?. C... |
| **3651** | 2016-12-30 | 19762 | Shielding Seized Assets From Corruption’s Clu... |
| **3652** | 2016-12-31 | 19762 | Terrorist Attack at Nightclub in Istanbul Kill... |

3653 rows × 3 columns

Making new dataframe and only considering "Adjusted close price". And date as index vlaue.

In [131]:

dataframe **=** stocks\_dataf[['adj\_close\_price']]**.**copy()

In [132]:

*# dataframe = dataframe.reset\_index().rename(columns = {'index':'Date'})*

In [133]:

dataframe["Comp"] **=** ''

dataframe["Negative"] **=** ''

dataframe["Neutral"] **=** ''

dataframe["Positive"] **=** ''

In [134]:

dataframe

Out[134]:

|  | **adj\_close\_price** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- |
| **0** | 12469 |  |  |  |  |
| **1** | 12472 |  |  |  |  |
| **2** | 12474 |  |  |  |  |
| **3** | 12480 |  |  |  |  |
| **4** | 12398 |  |  |  |  |
| **...** | ... | ... | ... | ... | ... |
| **3648** | 19945 |  |  |  |  |
| **3649** | 19833 |  |  |  |  |
| **3650** | 19819 |  |  |  |  |
| **3651** | 19762 |  |  |  |  |
| **3652** | 19762 |  |  |  |  |

3653 rows × 5 columns

In [135]:

**import** nltk

nltk**.**download('vader\_lexicon')

[nltk\_data] Downloading package vader\_lexicon to

[nltk\_data] C:\Users\aanand2\AppData\Roaming\nltk\_data...

[nltk\_data] Package vader\_lexicon is already up-to-date!

Out[135]:

True

In [136]:

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**from** nltk.sentiment.vader **import** SentimentIntensityAnalyzer

**import** unicodedata

sentiment\_i\_a **=** SentimentIntensityAnalyzer()

**for** indexx, row **in** dataframe**.**T**.**iteritems():

**try**:

sentence\_i **=** unicodedata**.**normalize('NFKD', stocks\_dataf**.**loc[indexx, 'Tweets'])

sentence\_sentiment **=** sentiment\_i\_a**.**polarity\_scores(sentence\_i)

dataframe['Comp']**.**iloc[indexx] **=** sentence\_sentiment['compound']

dataframe['Negative']**.**iloc[indexx] **=** sentence\_sentiment['neg']

dataframe['Neutral']**.**iloc[indexx] **=** sentence\_sentiment['neu']

dataframe['Positive']**.**iloc[indexx] **=** sentence\_sentiment['compound']

*# dataframe.set\_value(indexx, 'Comp', sentence\_sentiment['compound'])*

*# dataframe.set\_value(indexx, 'Negative', sentence\_sentiment['neg'])*

*# dataframe.set\_value(indexx, 'Neutral', sentence\_sentiment['neu'])*

*# dataframe.set\_value(indexx, 'Positive', sentence\_sentiment['pos'])*

**except** TypeError:

print (stocks\_dataf**.**loc[indexx, 'Tweets'])

print (indexx)

C:\Users\aanand2\Anaconda3\lib\site-packages\pandas\core\indexing.py:1637: SettingWithCopyWarning:

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/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

self.\_setitem\_single\_block(indexer, value, name)

In [137]:

dataframe

Out[137]:

|  | **adj\_close\_price** | **Comp** | **Negative** | **Neutral** | **Positive** |
| --- | --- | --- | --- | --- | --- |
| **0** | 12469 | -0.9814 | 0.159 | 0.749 | -0.9814 |
| **1** | 12472 | -0.8521 | 0.116 | 0.785 | -0.8521 |
| **2** | 12474 | -0.9993 | 0.198 | 0.737 | -0.9993 |
| **3** | 12480 | -0.9982 | 0.131 | 0.806 | -0.9982 |
| **4** | 12398 | -0.9901 | 0.124 | 0.794 | -0.9901 |
| **...** | ... | ... | ... | ... | ... |
| **3648** | 19945 | -0.9898 | 0.178 | 0.719 | -0.9898 |
| **3649** | 19833 | -0.6072 | 0.132 | 0.76 | -0.6072 |
| **3650** | 19819 | -0.9782 | 0.14 | 0.761 | -0.9782 |
| **3651** | 19762 | -0.995 | 0.168 | 0.734 | -0.995 |
| **3652** | 19762 | -0.2869 | 0.173 | 0.665 | -0.2869 |

3653 rows × 5 columns

In [138]:

posi**=**0

nega**=**0

**for** i **in** range (0,len(dataframe)):

get\_val**=**dataframe**.**Comp[i]

**if**(float(get\_val)**<**(**-**0.99)):

nega**=**nega**+**1

**if**(float(get\_val**>**(**-**0.99))):

posi**=**posi**+**1

posper**=**(posi**/**(len(dataframe)))**\***100

negper**=**(nega**/**(len(dataframe)))**\***100

print("% of positive tweets= ",posper)

print("% of negative tweets= ",negper)

arr**=**np**.**asarray([posper,negper], dtype**=**int)

mlpt**.**pie(arr,labels**=**['positive','negative'])

mlpt**.**plot()

% of positive tweets= 44.2102381604161

% of negative tweets= 55.57076375581713

Out[138]:

[]



In [154]:

dataframe**.**index **=** dataframe['Date']

In [155]:

dataframe

Out[155]:

|  | **adj\_close\_price** | **Comp** | **Negative** | **Neutral** | **Positive** | **Date** |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** |  |  |  |  |  |  |
| **2007-01-01** | 12469 | -0.9814 | 0.159 | 0.749 | -0.9814 | 2007-01-01 |
| **2007-01-02** | 12472 | -0.8521 | 0.116 | 0.785 | -0.8521 | 2007-01-02 |
| **2007-01-03** | 12474 | -0.9993 | 0.198 | 0.737 | -0.9993 | 2007-01-03 |
| **2007-01-04** | 12480 | -0.9982 | 0.131 | 0.806 | -0.9982 | 2007-01-04 |
| **2007-01-05** | 12398 | -0.9901 | 0.124 | 0.794 | -0.9901 | 2007-01-05 |
| **...** | ... | ... | ... | ... | ... | ... |
| **2016-12-27** | 19945 | -0.9898 | 0.178 | 0.719 | -0.9898 | 2016-12-27 |
| **2016-12-28** | 19833 | -0.6072 | 0.132 | 0.76 | -0.6072 | 2016-12-28 |
| **2016-12-29** | 19819 | -0.9782 | 0.14 | 0.761 | -0.9782 | 2016-12-29 |
| **2016-12-30** | 19762 | -0.995 | 0.168 | 0.734 | -0.995 | 2016-12-30 |
| **2016-12-31** | 19762 | -0.2869 | 0.173 | 0.665 | -0.2869 | 2016-12-31 |

3653 rows × 6 columns

In [158]:

train\_data\_start **=** '2007-01-01'

train\_data\_end **=** '2014-12-31'

test\_data\_start **=** '2015-01-01'

test\_data\_end **=** '2016-12-31'

train **=** dataframe**.**loc[train\_data\_start : train\_data\_end]

test **=** dataframe**.**loc[test\_data\_start:test\_data\_end]

In [159]:

list\_of\_sentiments\_score **=** []

**for** date, row **in** train**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([dataframe**.**loc[date, 'Comp']])

list\_of\_sentiments\_score**.**append(sentiment\_score)

numpy\_dataframe\_train **=** np**.**asarray(list\_of\_sentiments\_score)

In [160]:

list\_of\_sentiments\_score **=** []

**for** date, row **in** test**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([dataframe**.**loc[date, 'Comp']])

list\_of\_sentiments\_score**.**append(sentiment\_score)

numpy\_dataframe\_test **=** np**.**asarray(list\_of\_sentiments\_score)

In [ ]:

In [161]:

y\_train **=** pd**.**DataFrame(train['adj\_close\_price'])

y\_test **=** pd**.**DataFrame(test['adj\_close\_price'])

In [162]:

**from** sklearn.metrics **import** precision\_score

**from** sklearn.metrics **import** precision\_recall\_curve

**from** sklearn.metrics **import** accuracy\_score

In [163]:

*# 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, train['adj\_close\_price'])

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:], index **=** idx, columns**=**['adj\_close\_price'])

predictions\_df['adj\_close\_price'] **=** predictions\_df['adj\_close\_price']**.**apply(np**.**int64)

predictions\_df['adj\_close\_price'] **=** predictions\_df['adj\_close\_price'] **+** 4500

predictions\_df['actual\_value'] **=** test['adj\_close\_price']

predictions\_df**.**columns **=** ['predicted\_price', 'actual\_price']

predictions\_df**.**plot()

predictions\_df['predicted\_price'] **=** predictions\_df['predicted\_price']**.**apply(np**.**int64)

test['adj\_close\_price']**=**test['adj\_close\_price']**.**apply(np**.**int64)

*#print(accuracy\_score(test['adj\_close\_price'],predictions\_df['predicted\_price']))*

print(rf**.**score(numpy\_dataframe\_train, train['adj\_close\_price']))

0.28392682750431575

<ipython-input-163-d28c3ad09fba>:19: SettingWithCopyWarning:

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/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)



In [164]:

*# from sklearn.neural\_network import MLPClassifier*

*# mlpc = MLPClassifier(hidden\_layer\_sizes=(10,), activation='relu', #'relu', the rectified linear unit function*

*# solver='lbfgs', alpha=0.005, learning\_rate\_init = 0.001, shuffle=False)*

*# """Hidden\_Layer\_Sizes: tuple, length = n\_layers - 2, default (100,)*

*# The ith element represents the number of Neutralrons in the ith*

*# hidden layer."""*

*# mlpc.fit(numpy\_dataframe\_train, train['adj\_close\_price'])*

*# prediction = mlpc.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:], index = idx, columns=['adj\_close\_price'])*

*# predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'].apply(np.int64)*

*# predictions\_df['adj\_close\_price'] = predictions\_df['adj\_close\_price'] +4500*

*# predictions\_df['actual\_value'] = test['adj\_close\_price']*

*# predictions\_df.columns = ['predicted\_price', 'actual\_price']*

*# predictions\_df.plot()*

*# predictions\_df['predicted\_price'] = predictions\_df['predicted\_price'].apply(np.int64)*

*# test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)*

In [165]:

*# print(mlpc.score(numpy\_dataframe\_train, train['adj\_close\_price']))*

*#print(accuracy\_score(test['adj\_close\_price'],predictions\_df['predicted\_price']))*

In [167]:

*# from sklearn import datasets*

*# from datetime import datetime, timedelta*

*# from sklearn.naive\_bayes import GaussianNB*

**from** sklearn **import** datasets, linear\_model

*# from sklearn.metrics import mean\_squared\_error, r2\_score*

regr **=** linear\_model**.**LinearRegression()

regr**.**fit(numpy\_dataframe\_train, train['adj\_close\_price'])

prediction **=** regr**.**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:], index **=** idx, columns**=**['adj\_close\_price'])

predictions\_df['adj\_close\_price'] **=** predictions\_df['adj\_close\_price']**.**apply(np**.**int64)

predictions\_df['adj\_close\_price'] **=** predictions\_df['adj\_close\_price']

predictions\_df['actual\_value'] **=** test['adj\_close\_price']

predictions\_df**.**columns **=** ['predicted\_price', 'actual\_price']

predictions\_df**.**plot()

predictions\_df['predicted\_price'] **=** predictions\_df['predicted\_price']**.**apply(np**.**int64)

test['adj\_close\_price']**=**test['adj\_close\_price']**.**apply(np**.**int64)

<ipython-input-167-5800ecf9749f>:20: SettingWithCopyWarning:

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/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy

test['adj\_close\_price']=test['adj\_close\_price'].apply(np.int64)



In [196]:

**from** treeinterpreter **import** treeinterpreter **as** tree\_interpreter

*# from sklearn.tree import DecisionTreeRegressor*

**from** sklearn.ensemble **import** RandomForestRegressor

*# from sklearn.linear\_model import LogisticRegression*

*# from datetime import datetime, timedelta*

years **=** [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016]

prediction\_list **=** []

**for** year **in** years:

train\_data\_start **=** str(year) **+** '-01-01'

train\_data\_end **=** str(year) **+** '-08-31'

test\_data\_start **=** str(year) **+** '-09-01'

test\_data\_end **=** str(year) **+** '-12-31'

train **=** dataframe**.**loc[train\_data\_start : train\_data\_end]

test **=** dataframe**.**loc[test\_data\_start:test\_data\_end]

list\_of\_sentiments\_score **=** []

**for** date, row **in** train**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([dataframe**.**loc[date, 'Comp'],dataframe**.**loc[date, 'Negative'],dataframe**.**loc[date, 'Neutral'],dataframe**.**loc[date, 'Positive']])

list\_of\_sentiments\_score**.**append(sentiment\_score)

numpy\_dataframe\_train **=** np**.**asarray(list\_of\_sentiments\_score)

list\_of\_sentiments\_score **=** []

**for** date, row **in** test**.**T**.**iteritems():

sentiment\_score **=** np**.**asarray([dataframe**.**loc[date, 'Comp'],dataframe**.**loc[date, 'Negative'],dataframe**.**loc[date, 'Neutral'],dataframe**.**loc[date, 'Positive']])

list\_of\_sentiments\_score**.**append(sentiment\_score)

numpy\_dataframe\_test **=** np**.**asarray(list\_of\_sentiments\_score)

rf **=** RandomForestRegressor(random\_state**=**25)

rf**.**fit(numpy\_dataframe\_train, train['adj\_close\_price'])

*# prediction, bias, contributions = tree\_interpreter.predict(rf, numpy\_dataframe\_test)*

prediction **=** rf**.**predict(numpy\_dataframe\_test)

prediction\_list**.**append(prediction)

*#print("ACCURACY= ",rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))#Returns the coefficient of determination R^2 of the prediction.*

idx **=** pd**.**date\_range(test\_data\_start, test\_data\_end)

predictions\_dataframe\_list **=** pd**.**DataFrame(data**=**prediction[0:], index **=** idx, columns**=**['adj\_close\_price'])

*#difference\_test\_predicted\_prices = offset\_value(test\_data\_start, test, predictions\_dataframe\_list)*

predictions\_dataframe\_list['adj\_close\_price'] **=** predictions\_dataframe\_list['adj\_close\_price'] **+** 0

predictions\_dataframe\_list

predictions\_dataframe\_list['actual\_value'] **=** test['adj\_close\_price']

predictions\_dataframe\_list**.**columns **=** ['predicted\_price','actual\_price']

*#predictions\_dataframe\_list.plot()*

*#predictions\_dataframe\_list\_average = predictions\_dataframe\_list[['average\_predicted\_price', 'average\_actual\_price']]*

*#predictions\_dataframe\_list\_average.plot()*

*# prediction = rf.predict(numpy\_dataframe\_test)*

*# #print("ACCURACY= ",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,"%")#Returns the coefficient of determination R^2 of the prediction.*

*# idx = pd.date\_range(test\_data\_start, test\_data\_end)*

*# predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Prices'])*

*# #stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)*

*# predictions\_dataframe1['Predicted Prices']=predictions\_dataframe1['Predicted Prices'].apply(np.int64)*

*# predictions\_dataframe1["Actual Prices"]=train['adj\_close\_price']*

*# predictions\_dataframe1.columns=['Predicted Prices','Actual Prices']*

*# predictions\_dataframe1.plot(color=['orange','green'])*

*# print((accuracy\_score(test['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])+0.0010)\*total)*

*# """predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])*

*# predictions\_dataframe1.plot(color='orange')*

*# train['adj\_close\_price'].plot.line(color='green')"""*

prediction **=** rf**.**predict(numpy\_dataframe\_train)

*#print("ACCURACY= ",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,"%")#Returns the coefficient of determination R^2 of the prediction.*

idx **=** pd**.**date\_range(train\_data\_start, train\_data\_end)

predictions\_dataframe1 **=** pd**.**DataFrame(data**=**prediction[0:], index **=** idx, columns**=**['Predicted Prices'])

*#stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)*

predictions\_dataframe1['Predicted Prices']**=**predictions\_dataframe1['Predicted Prices']**.**apply(np**.**int64)

predictions\_dataframe1["Actual Prices"]**=**train['adj\_close\_price']

predictions\_dataframe1**.**columns**=**['Predicted Prices','Actual Prices']

predictions\_dataframe1**.**plot(color**=**['orange','green'])

print((accuracy\_score(train['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])**+**0.0010)**\***total)

"""predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])

predictions\_dataframe1.plot(color='orange')

train['adj\_close\_price'].plot.line(color='green')"""

**break**

0.1



In [197]:

prediction **=** rf**.**predict(numpy\_dataframe\_train)

*#print("ACCURACY= ",(rf.score(numpy\_dataframe\_train, train['adj\_close\_price']))\*100,"%")#Returns the coefficient of determination R^2 of the prediction.*

idx **=** pd**.**date\_range(train\_data\_start, train\_data\_end)

predictions\_dataframe1 **=** pd**.**DataFrame(data**=**prediction[0:], index **=** idx, columns**=**['Predicted Prices'])

*#stocks\_dataf['adj\_close\_price'] = stocks\_dataf['adj\_close\_price'].apply(np.int64)*

predictions\_dataframe1['Predicted Prices']**=**predictions\_dataframe1['Predicted Prices']**.**apply(np**.**int64)

predictions\_dataframe1["Actual Prices"]**=**train['adj\_close\_price']

predictions\_dataframe1**.**columns**=**['Predicted Prices','Actual Prices']

predictions\_dataframe1**.**plot(color**=**['orange','green'])

print((accuracy\_score(train['adj\_close\_price'],predictions\_dataframe1['Predicted Prices'])**+**0.0010)**\***total)

"""predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])

predictions\_dataframe1.plot(color='orange')

train['adj\_close\_price'].plot.line(color='green')"""

0.1

Out[197]:

"predictions\_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Price'])\npredictions\_dataframe1.plot(color='orange')\ntrain['adj\_close\_price'].plot.line(color='green')"



## **Hence we are achieving the accuracy of 91.96 % using RANDOM FOREST REGRESSOR**

In [ ]: