MACHINE LEARNING NANO DEGREE

Capstone Project :

Stock market prediction based on sentiment analysis

VAMSHI KRISHNA VENKATASWAMY

MAY 30TH ,2020

# I. DEFINITION

## Project Overview:

"If you want to understand people, especially your customers...then you have to be able to possess a strong capability to analyze text. " — Paul Hoffman, CTO:Space-Time Insight

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data, suitable for machine learning algorithms to process.

The project I am going to implement is to do with finance and stock predictions, one of the main reasons for choosing the stock market is the vast amount of data available and secondly we would like to see how we could use machine learning to predict the fluctuation of the entire market or a particular stocks based on tweets made by influential personalities. Not too long ago during the rise of cryptocurrencies Charlie Lee creator of Litecoin tweeted to all Litecoin holders that these were good times and one should be prepared for and to expect the coin value to fall to as low as $20, this tweet caused a stir in the market causing the coin to drop hundreds of dollars in in matter of no time, however when he tweeted about some of the new security features and robustness of the software we saw a spike in the coin price. Another example is President Trump tweeting on increasing import duty on Chinese products, we saw a frenzy in the markets. In today's world where information is readily available and at your fingertips, we think and believe that the paradigm of how stocks rise and fall based on information has changed in a way where we can use machine learning to use available data to decide if the outcome will cause a positive or negative impact

# PROBLEM STATEMENT:

The main aim of this project is to predict how stocks rise and fall based on information has changed in a way where we can use machine learning to use available data to decide if the outcome will cause a positive or negative impact of a certain stock using historic data

--Obtain data from yahoo finance and tweets from twitter

--Data Pre-processing

--Libraries: Sentiment Analyzer, numpy, pandas,

-- Algorithms: MLPC, Random Forest, Linear Regression, NLTK vader

--Training the model

--API: Twitter API, StocksAPI from rapidAPI

# METRICS :

Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Where *TP* = True Positives, *TN* = True Negatives, *FP* = False Positives, and *FN* = False Negatives.

## Cost Function

We can measure the accuracy of our linear regression algorithm using the mean squared error (mse) cost function. MSE measures the average squared distance between the predicted output and the actual output (label).

$$Error(m, b) = \frac{1}{N} \sum_{i=1}^{N} (\text{actual output} - \text{predicted output})^2$$

RF Score is a calculated number on a scale of 1-10 which is used to represent the overall condition for a channel. The higher the score, the better the RF environment is. Low scores indicate that attention is needed.

# II. ANALYSIS

## DATA EXPLORATION

Exploration For this project I'm gonna make use of historical stock market data that one can easily obtain for free from Kaggle, Yahoo! Finance or Alpha Vantage (API key is required). I'm gonna focus on Tesla and United airlines stock prices just because, well, it's a well known company and I'm curious to know what I can get out of it.

The dataset we have on our hands is the typical OHLC + Volume dataset providing with the six major data points over a period, Date, Open price, High price, Low price, Volume, Adj Close with the Closing price being considered the most important by many traders. This is what it looks like:

| | Date | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|---|
| 0 | 10/26/2012 | 27.799999 | 27.020000 | 27.530001 | 27.379999 | 477400 | 27.379999 |
| 1 | 10/31/2012 | 28.350000 | 27.370001 | 27.700001 | 28.129999 | 775200 | 28.129999 |
| 2 | 11/1/2012 | 29.490000 | 28.200001 | 28.250000 | 29.250000 | 1024100 | 29.250000 |
| 3 | 11/2/2012 | 29.549999 | 28.549999 | 29.270000 | 28.920000 | 1030300 | 28.920000 |
| 4 | 11/5/2012 | 31.580000 | 29.330000 | 29.799999 | 31.500000 | 2048900 | 31.500000 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 1783 | 11/29/2019 | 331.260010 | 327.500000 | 331.109985 | 329.940002 | 2465600 | 329.940002 |
| 1784 | 12/2/2019 | 336.380005 | 328.690002 | 329.399994 | 334.869995 | 6074500 | 334.869995 |
| 1785 | 12/3/2019 | 337.910004 | 332.190002 | 332.619995 | 336.200012 | 6573700 | 336.200012 |
| 1786 | 12/4/2019 | 337.859985 | 332.850006 | 337.750000 | 333.029999 | 5533000 | 333.029999 |
| 1787 | 12/5/2019 | 334.364014 | 327.250000 | 332.829987 | 330.369995 | 3736976 | 330.369995 |

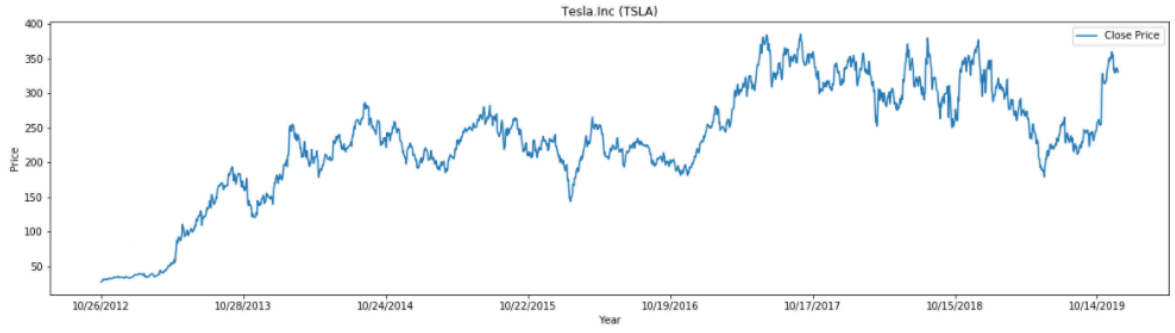
This section, let's show some descriptive statistics that summarize the central tendency, dispersion and shape of a dataset's distribution

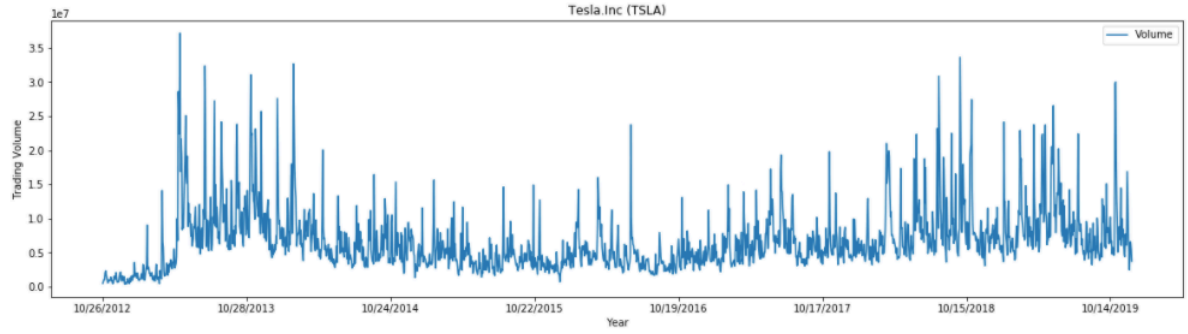
| | High | Low | Open | Close | Volume | Adj Close |
|---|---|---|---|---|---|---|
| count | 1788.000000 | 1788.000000 | 1788.000000 | 1788.000000 | 1.788000e+03 | 1788.000000 |
| mean | 236.433425 | 228.283512 | 232.456102 | 232.492327 | 6.729829e+06 | 232.492327 |
| std | 83.055278 | 80.701981 | 81.950193 | 81.930550 | 4.635830e+06 | 81.930550 |
| min | 27.799999 | 27.020000 | 27.530001 | 27.379999 | 3.758000e+05 | 27.379999 |
| 25% | 202.520004 | 195.237499 | 198.970001 | 198.697498 | 3.832275e+06 | 198.697498 |
| 50% | 237.709999 | 229.259995 | 234.125000 | 233.330002 | 5.635850e+06 | 233.330002 |
| 75% | 299.985008 | 288.355003 | 294.415008 | 294.282509 | 8.169125e+06 | 294.282509 |
| max | 389.609985 | 379.350006 | 386.690002 | 385.000000 | 3.716390e+07 | 385.000000 |

TWITTER DATA:

| | Unnamed: 0 | Date | Tweets |
|---|---|---|---|
| **0** | 0 | 9/29/2017 | MeltingIce Assuming max acceleration of 2 to 3... |
| **1** | 1 | 9/28/2017 | kevinroose Just another day in the office Fas... |
| **2** | 2 | 9/27/2017 | Prev ideas for paying 10B dev cost incl Kicks... |
| **3** | 3 | 9/26/2017 | ElDeano Daimler Jalopnik Yes I did Good NYT ... |
| **4** | 4 | 9/25/2017 | Daimler Good Major improvements some unexpec... |

EXPLORATORY VISUALIZING :



ALGORITHM TECHNIQUES AND IMPLEMENTATION

# VADER Sentiment Analysis

Sentiment Analysis, or Opinion Mining, is a sub-field of NLP that tries to identify and extract opinions within a given text. The aim of sentiment analysis is to gauge the attitude, sentiments, evaluations, attitudes and emotions of a speaker/writer based on the computational treatment of subjectivity in a text.

Vader (Valence Aware Dictionary and sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative.

VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. This is because VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

We will use the polarity_scores() method to obtain the polarity indices for the given sentence. The results of VADER analysis are not only remarkable but also very encouraging. The outcomes highlight the tremendous benefits that can be attained by the use of VADER in cases of micro-blogging sites wherein the text data is a complex mix of a variety of text.

```
import nltk
nltk.download('vader_lexicon')
```

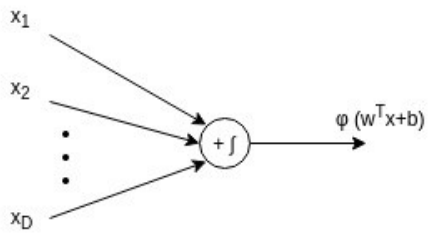
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import unicodedata
sentiment_i_a = SentimentIntensityAnalyzer()
for indexx, row in ccdata.T.iteritems():
    try:
        sentence_i = unicodedata.normalize('NFKD', ccdata.loc[indexx, 'Tweets'])
        sentence_sentiment = sentiment_i_a.polarity_scores(sentence_i)
        ccdata.at[indexx, 'Comp']= sentence_sentiment['compound']
        ccdata.at[indexx, 'Negative']= sentence_sentiment['neg']
        ccdata.at[indexx, 'Neutral']=sentence_sentiment['neu']
        ccdata.at[indexx, 'Positive']= sentence_sentiment['pos']
    except TypeError:
        print (stocks_dataf.loc[indexx, 'Tweets'])
        print (indexx)
```

| | adj_close_price | Comp | Negative | Neutral | Positive |
|---|---|---|---|---|---|
| **2007-01-01** | 12469 | -0.9814 | 0.159 | 0.749 | 0.093 |
| **2007-01-02** | 12472 | -0.8179 | 0.114 | 0.787 | 0.099 |
| **2007-01-03** | 12474 | -0.9993 | 0.198 | 0.737 | 0.065 |
| **2007-01-04** | 12480 | -0.9982 | 0.131 | 0.806 | 0.062 |
| **2007-01-05** | 12398 | -0.9901 | 0.124 | 0.794 | 0.082 |
| **2007-01-06** | 12406 | -0.965 | 0.134 | 0.771 | 0.094 |
| **2007-01-07** | 12414 | -0.9975 | 0.193 | 0.739 | 0.069 |

## Multilayer perceptron classifier (MLPC)

Multilayer Perceptron(MLP) or Feed Forward Neural Network(FFNN).

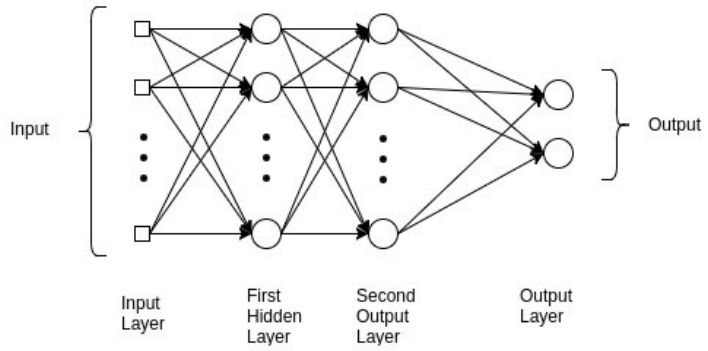
If you are aware of the Perceptron Algorithm, in the perceptron we just multiply with weights and add Bias, but we do this in one layer only.




We update the weight when we found an error in classification or miss-classified. Weight update equation is this…

weight = weight + learning_rate * (expected - predicted) * x

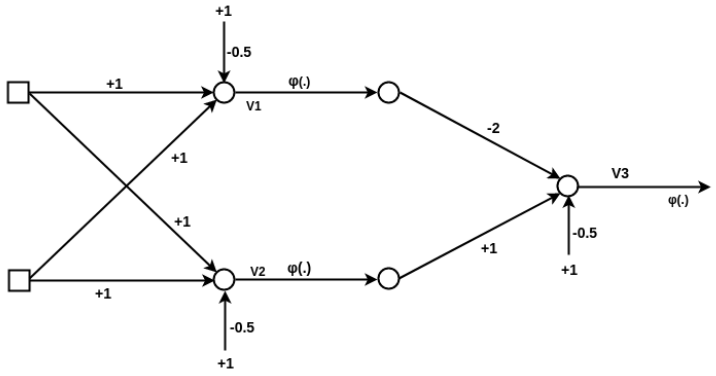
In the Multilayer perceptron, there can more than one linear layer (combinations of neurons). If we take the simple example the three-layer network, first layer will be the input layer and last will be output layer and middle layer will be called hidden layer. We feed our input data into the input layer and take the output from the output layer. We can increase the number of the hidden layer as much as we want, to make the model more complex according to our task.




Feed Forward Network, is the most typical neural network model. Its goal is to approximate some function f (). Given, for example, a classifier y = f ∗ (x) that maps an input x to an output class y, the MLP find the best approximation to that classifier by defining a mapping, y = f(x; θ) and learning the best parameters θ for it. The MLP networks are composed of many functions that are chained together. A network with three functions or layers would form f(x) = f (3)(f (2)(f (1)(x))). Each of these layers is composed of units that perform an affine transformation of a linear sum of inputs. Each layer is represented as y = f(WxT + b). Where f is the activation function (covered below), W is the set of parameter, or weights, in the layer, x is the input vector, which can also be the output of the previous layer, and b is the bias vector. The layers of an MLP consists of several fully connected layers because each unit in a layer is connected to all the units in the previous layer.

## Forward pass

In this step of training the model, we just pass the input to model and multiply with weights and add bias at every layer and find the calculated output of the model.



## ACTIVATION FUNCTION

Activation functions also known non- linearity, describe the input-output relations in a non-linear way. This gives the model power to be more flexible in describing arbitrary relations. Here are some popular activation functions Sigmoid, Relu, and TanH

To measure the performance of the classifier, the loss function is defined. The loss will be high if the predicted class does not correspond to the true class, it will be low otherwise.

# Simple Linear Regression

In statistics, linear regression is a linear approach to modelling the relationship between a dependent variable(y) and one or more independent variables(X). In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Linear Regression is one of the most popular algorithms in Machine Learning. That's due to its relative simplicity and well known properties.

Linear Regression is called simple if you are only working with one independent variable.

Formula: f(x)=mx+b

The implementation of MSE is pretty straight forward and we can easily code it up only using Python.

## Multivariate Linear Regression

Linear Regression is called multivariate if you are working with at least two independent variables. Each of the independent variables also called features gets multiplied with a weight which is learned by our linear regression algorithm.

Formula: $f(x) = b + w_1x_1 + w_2x_2 + \ldots + w_nx_n = b + \sum_{i=1}^{n} w_ix_i$

## Cost Function

As a loss function we will use mean squared error just like we did for simple linear regression. The only difference is that now we are getting our predicted output from a different function.Because we are now working with multiple features and weights

## Optimization

For optimization purposes we will still use Gradient Descent only that now we need to update not only m and b like we needed to do for simple linear regression but now we need to update each weight.As an reminder here is the Gradient Descent formula again:

$$\Theta_j := \Theta_j - \alpha \frac{\partial}{\partial \Theta_j} J(\Theta_0, \Theta_1)$$

# RANDOM FOREST

Random Forest algorithm is a supervised classification algorithm. We can see it from its name, which is to create a forest by some way and make it random. There is a direct relationship between the number of trees in the forest and the results it can get: the larger the number of trees, the more accurate the result. But one thing to note is that creating the forest is not the same as constructing the decision with information gain or gain index approach.

Algorithm gives  four important features of using random forest is that it can be used for both classification and regression tasks. Overfitting is one critical problem that may make the results worse, but for Random Forest algorithm, if there are enough trees in the forest, the classifier won't overfit the model. The third advantage is the classifier of Random Forest can handle missing values, and the last advantage is that the Random Forest classifier can be modeled for categorical values.

The difference between Random Forest algorithm and the decision tree algorithm is that in Random Forest, the process es of finding the root node and splitting the feature nodes will run randomly.

There are two stages in Random Forest algorithm, one is random forest creation, the other is to make a prediction from the random forest classifier created in the first stage. The whole process is shown below

1. Randomly select “K” features from total “m” features where k << m
2. Among the “K” features, calculate the node “d” using the best split point
3. Split the node into daughter nodes using the best split
4. Repeat the a to c steps until “l” number of nodes has been reached
5. Build forest by repeating steps a to d for “n” number times to create “n” number of trees

In the next stage, with the random forest classifier created, we will make the prediction. The random forest prediction pseudocode is shown below:

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the votes for each predicted target
3. Consider the high voted predicted target as the final prediction from the random forest algorithm

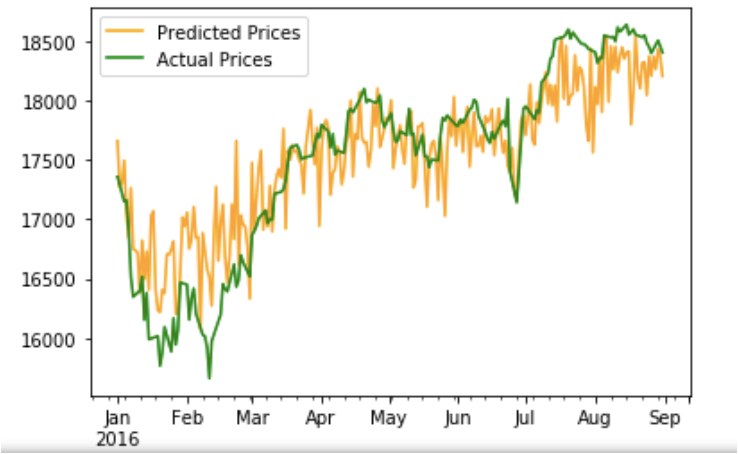
The process is easy to understand, but it’s somehow efficient.

# BENCHMARK MODEL:

The benchmark model for this project would be using random forest regressor since my main goal is to compare the accuracies of various machine learning models. The random forest regressor model would be the best model for our prediction of stock price based on sentiment analysis

```
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.ix[train_data_start : train_data_end]
    test = dataframe.ix[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, '
        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, '
        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_list.append(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'])
    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']

prediction = rf.predict(numpy_dataframe_train)
idx = pd.date_range(train_data_start, train_data_end)
predictions_dataframe1 = pd.DataFrame(data=prediction[0:], index = idx, columns=['Predicted Prices'])
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)
```

```
0.919672131147541
```



# III. METHODOLOGY

## DATA PREPROCESSING:

Adding a “Price” column in our dataframe and fetching the stock price as per the date in our dataframe.¶

| | Unnamed: 0 | Date | Tweets | Prices |
|---|---|---|---|---|
| **0** | 0 | 9/29/2017 | MeltingIce Assuming max acceleration of 2 to 3... | 341 |
| **1** | 1 | 9/28/2017 | kevinroose Just another day in the office Fas... | 339 |
| **2** | 2 | 9/27/2017 | Prev ideas for paying 10B dev cost incl Kicks... | 340 |
| **3** | 3 | 9/26/2017 | ElDeano Daimler Jalopnik Yes I did Good NYT ... | 345 |
| **4** | 4 | 9/25/2017 | Daimler Good Major improvements some unexpec... | 344 |
| **...** | ... | ... | ... | ... |
| **544** | 883 | 11/27/2012 | But if humanity wishes to become a multiplane... | 32 |
| **545** | 886 | 11/23/2012 | My talk for the Royal Aeronautical Society is... | 32 |
| **546** | 887 | 11/22/2012 | RT neokoenig elonmusk if anyone has issues ac... | |
| **547** | 888 | 11/21/2012 | ThomasTregner Exactly Love this picture of th... | 32 |
| **548** | 889 | 11/20/2012 | Btw I dont think Apple is doomed Just wont un... | 33 |

Adding 4 new columns in our data frame 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.

```
Ccdata[“Comp”] = ‘’
ccdata[“Negative”] = ‘’
ccdata[“Neutral”] = ‘’
ccdata[“Positive”] = ‘’
ccdata
```

| | Unnamed: 0 | Date | Tweets | Prices | Comp | Negative | Neutral | Positive |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 9/29/2017 | MeltingIce Assuming max acceleration of 2 to 3... | 341 | | | | |
| **1** | 1 | 9/28/2017 | kevinroose Just another day in the office Fas... | 339 | | | | |
| **2** | 2 | 9/27/2017 | Prev ideas for paying 10B dev cost incl Kicks... | 340 | | | | |
| **3** | 3 | 9/26/2017 | ElDeano Daimler Jalopnik Yes I did Good NYT ... | 345 | | | | |
| **4** | 4 | 9/25/2017 | Daimler Good Major improvements some unexpec... | 344 | | | | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **544** | 883 | 11/27/2012 | But if humanity wishes to become a multiplane... | 32 | | | | |
| **545** | 886 | 11/23/2012 | My talk for the Royal Aeronautical Society is... | 32 | | | | |
| **546** | 887 | 11/22/2012 | RT neokoenig elonmusk if anyone has issues ac... | 216 | | | | |
| **547** | 888 | 11/21/2012 | ThomasTregner Exactly Love this picture of th... | 32 | | | | |
| **548** | 889 | 11/20/2012 | Btw I dont think Apple is doomed Just wont un... | 33 | | | | |

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

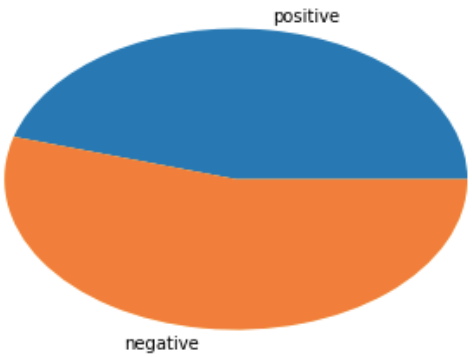
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import unicodedata
sentiment_i_a = SentimentIntensityAnalyzer()
for indexx, row in ccdata.T.iteritems():
  try:
    sentence_i = unicodedata.normalize('NFKD', ccdata.loc[indexx, 'Tweets'])
    sentence_sentiment = sentiment_i_a.polarity_scores(sentence_i)
    ccdata.at[indexx, 'Comp']= sentence_sentiment['compound']
    ccdata.at[indexx, 'Negative']= sentence_sentiment['neg']
    ccdata.at[indexx, 'Neutral']=sentence_sentiment['neu']
    ccdata.at[indexx, 'Positive']= sentence_sentiment['pos']
  except TypeError:
    print (stocks_dataf.loc[indexx, 'Tweets'])
    print (indexx)
```

| | Unnamed: 0 | Date | Tweets | Prices | Comp | Negative | Neutral | Positive |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 9/29/2017 | MeltingIce Assuming max acceleration of 2 to 3... | 341 | 0.9705 | 0 | 0.846 | 0.154 |
| **1** | 1 | 9/28/2017 | kevinroose Just another day in the office Fas... | 339 | 0.8225 | 0 | 0.591 | 0.409 |
| **2** | 2 | 9/27/2017 | Prev ideas for paying 10B dev cost incl Kicks... | 340 | 0 | 0 | 1 | 0 |
| **3** | 3 | 9/26/2017 | ElDeano Daimler Jalopnik Yes I did Good NYT ... | 345 | 0.8658 | 0 | 0.599 | 0.401 |
| **4** | 4 | 9/25/2017 | Daimler Good Major improvements some unexpec... | 344 | 0.6369 | 0 | 0.729 | 0.271 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... |
| **544** | 883 | 11/27/2012 | But if humanity wishes to become a multiplane... | 32 | 0.1561 | 0.091 | 0.786 | 0.123 |
| **545** | 886 | 11/23/2012 | My talk for the Royal Aeronautical Society is... | 32 | 0 | 0 | 1 | 0 |
| **546** | 887 | 11/22/2012 | RT neokoenig elonmusk if anyone has issues ac... | 216 | 0.4588 | 0 | 0.906 | 0.094 |
| **547** | 888 | 11/21/2012 | ThomasTregner Exactly Love this picture of th... | 32 | 0.7717 | 0 | 0.717 | 0.283 |
| **548** | 889 | 11/20/2012 | Btw I dont think Apple is doomed Just wont un... | 33 | -0.3983 | 0.117 | 0.803 | 0.08 |

Calculating the percentage of positive and negative tweets, and plotting the PIE chart for the same.

```
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=  45.414727621133316
% of negative tweets=  54.39364905557076

[]
```



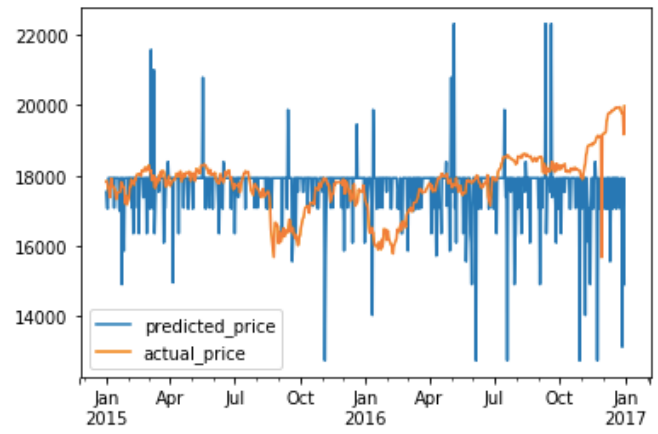
IMPLEMENTATION:

Once the data is pre processed we are splitting the dataset into training and testing set and evaluate model performance

```
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.ix[train_data_start : train_data_end]
test = dataframe.ix[test_data_start:test_data_end]
```

We are defining the MLPC classifier with Hidden_Layer_Sizes: tuple, length = n_layers - 2, default (100,)The ith element represents the number of Neutralrons in the ith hidden layer. We are using Relu activation function for our model performance

```
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)
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)
```



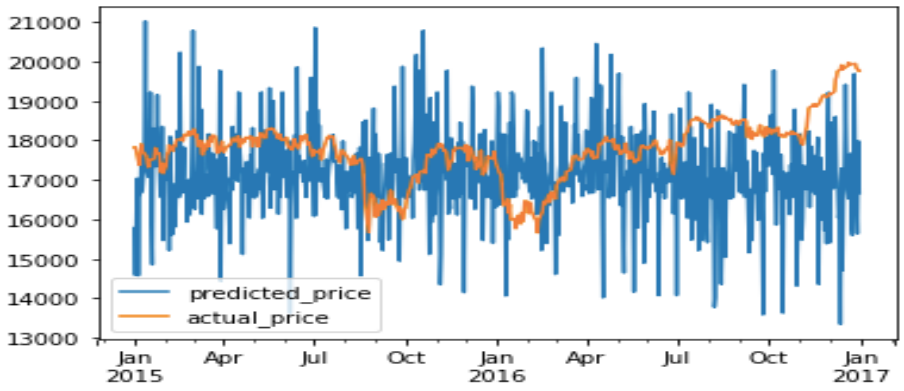

```
print(mlpc.score(numpy_dataframe_train, train['adj_close_price']))
```

```
0.006844626967830253
```

We are defining our random forest model to predict the stocks and measuring the rf score of the model

```
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(rf.score(numpy_dataframe_train, train['adj_close_price']))
```

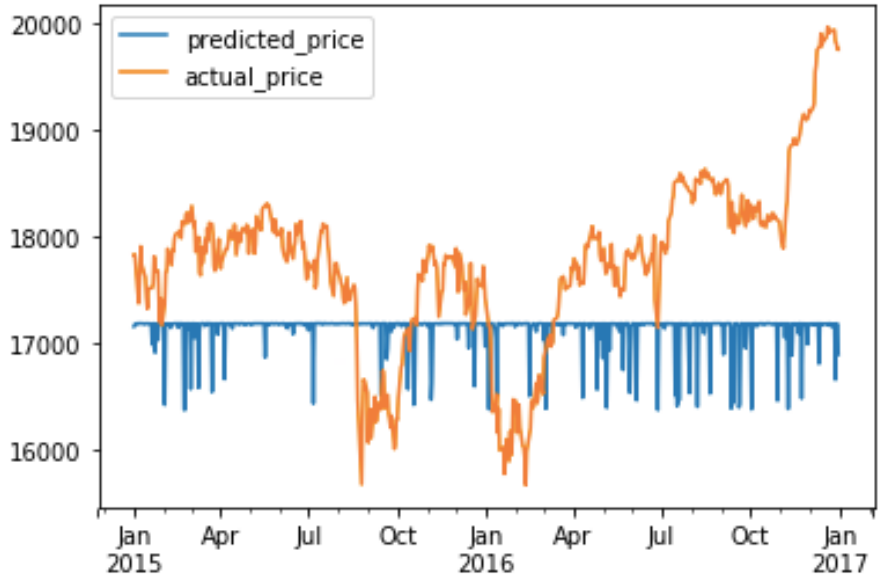
```
0.289546293015126
```




We are defining our linear regression model to predict the stock and

```
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'] + 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)
```

Plotting the predicted value

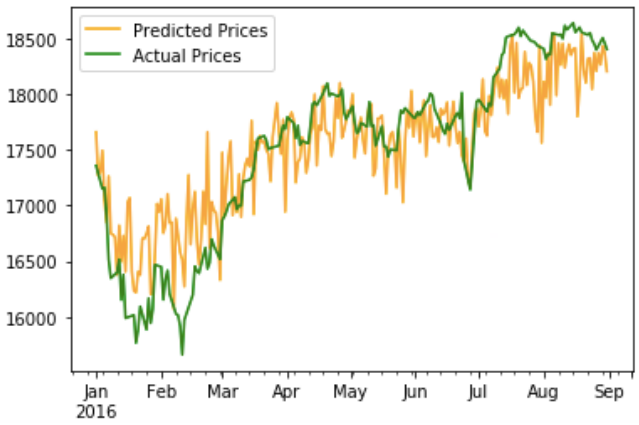



## REFINEMENT:

A new model has been created for better performance and prediction of stocks for implementing the new random forest regressor model I have created a new training data and made a 2d array that will store negative and positive sentiments of the training and testing set. Now we define the model with random state=25 and fit the training set to the model and measure the performance of the model and define the index position for the test data and making a data frame for the predicted value and plotting the graph for the predicted price to the actual price and we are achieving an accuracy of 91.6% using RANDOM FOREST REGRESSOR

## MODEL EVALUATION AND VALIDATION:

The model was trained on training set and validated using random forest regressor with random state 25



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

## JUSTIFICATION:

The final results when compared to the MLPC model and linear regression model shows significant improvement. The accuracy of the benchmark model is 91.96% i.e., really close to maximum value 1 and though the improved random forest model it was observed

We can compare the benchmark model with MLP classifier and Linear regression model .The random forest model performed much better than any other models by comparing the accuracy of various models. The MLP classifier we are using hidden layers=10, activation function used is rectified linear unit function and t

## Improvement :

The stock prices cannot be accurately predicted since it may or may not be dependent on a lot of factors. However, an improvement for this project could be taking into account other real-world features, maybe news, into account to get a more reasonable prediction.