# Stock Market Prediction using Linear Regression

## ECS784P DATA ANALYTICS COURSEWORK

**DATE: - 13TH APRIL 2018**


SUBMITTED BY

Shubham Chandel(170801265)
Mohit Sonania(170810234)
Majd Rafic Badran(171011720)

Advisor
Dr. Anthony Constantinou
Proff. Bhushan Chettri

# Table of Contents

# I. Abstract

The stock market is one of the most complex computational system in this era. Where the trading of such a huge amount of money is done on the basis of the market value of a company's stock. But for the long run we need to predict the future of the stock prices so that the investment occurs as per the planned way. As failure to this can cause huge loss of investor as well as for the company also. The stock market prices are difficult to be predicted by a human being, as there can be large number of factors which can cause the value of the stock going up and down.
One of the most effective way to predict the stock market prices is using computation which can be very complex.

As we are getting deeper into the technology, machine learning is increasing at a very fast pace. And in continuity to this, using the machine learning algorithms the prediction can be done at a very effective way. The algorithm which we will be using in this model is supervised machine learning algorithm – linear regression.

The code is implemented in jupyter notebook using the libraries such as scikit learn, matplotlib and numpy. And after this the plotting is done and the results are discussed.

# II. Project Briefing

## 1. Supervised Machine Learning

### 1.1 Literature Review

Supervised learning is a type of machine learning algorithm that uses a known dataset (called the training dataset) to make predictions. The training dataset includes input data and response values. From it, the supervised learning algorithm seeks to build a model that can make predictions of the response values for a new dataset. A test dataset is often used to validate the model. Using larger training datasets often yield models with higher predictive power that can generalize well for new datasets.
Supervised learning includes two categories of algorithms:

- **Classification**: for categorical response values, where the data can be separated into specific "classes"
- **Regression**: for continuous-response values. [1]

Some common regression algorithms are

- Linear Regression
- Support Vector Regression
- Neural Networks
- Naïve Bayes
- Decision Trees

In this coursework we will be using Linear Regression algorithm for the stock market prediction.

### 1.2 Linear Regression

Linear regression is a common statistical technique used to forecast values using the least squares fit method. This technique is applied in technical analysis to determine if the market trending up or down and what should the price be, given the recent trend of prices. In other words, the Linear Regression Forecast is a prediction of future prices charted in the present based on past quotes.

The Linear regression technique is used for calculating the anticipated value of one variable when one has the values of independent variable or variables M. This is in the form of a straight line which "best fits" the data points of prices available. In technical analysis, the Linear Regression line shows the principal market trend with respect to time. It is used to analyze when the market is deviating from the trend. The Linear Regression Forecast is drawn by calculating the regression based on a certain fixed regression time period (X). This is used to draw the regression line at each bar or time period based on the previous (X-1) time periods and optionally forecasting it for Y future time periods. The Linear Regression Forecast line thus drawn illustrates where prices "should" be trading. Any abnormal divergence from the regression line is likely to be temporary or it may signal that a major trend reversal is taking place. However, this is useful as an indicator only when markets are not highly volatile, and prices do not swing prodigiously around the trend. The less volatile the market, the better the fit of the equation is to the data, and therefore the more dependable is the linear trend. [2]

### 1.3 Motivation

Linear regression is an extremely simple method. It is very easy and intuitive to use and understand. A person with only the knowledge of high school mathematics can understand and use it. In addition, it works in most of the cases. Even when it doesn't fit the data exactly, we can use it to find the nature of the relationship between the two variables.

Linear Regression is a good supervised learning algorithm which is used to predictions problems, it find the target variable by finding a best suitable fit line between the independent and dependent variables.its main advantage is , the best fit line is the line with minimum error from all the points ,it has high efficiency but sometimes this high efficiency created disadvantage which is prone to over fitting of the data (i.e. some noisy data also considered as useful data), and also it can't be used when the relation between dependent and independent variable is not linear.[9]

# 2. Project

## 2.1 Dataset

Financial data consists of pieces or sets of information related to the financial health of a business. The pieces of data are used by internal management to analyze business performance and determine whether tactics and strategies must be altered.People and organizations outside a business will also use financial data reported by the business to judge its credit worthiness, decide whether to invest in the business, and determine whether the business is complying with government regulations.[9]

In this project, google stock market data set is used which is located in kaggle open source datasets. The dataset is in txt format and was transformed into pandas DataFrame where each row represents the data for one day and each column would represent a feature of the data set consisting of Open, Close, High, Low, Volume, and OpenInt.

Any financial dataset represents the above mentioned features.

Columns 1 and 4(Open and Close)- These are the open and close prices of the stock for the particular day.

Cloumns 2 and 3(High and low)- These are the highest and lowest prices at which a stock has traded over the particular day.

Cloumn 5(Volume)- This figure shows the total number of shares traded for the day.

Below is a snipped image of the DataFrame representing all the features and values.

| Date | Open | High | Low | Close | Volume | OpenInt |
|---|---|---|---|---|---|---|
| 2005-02-25 | 141.94 | 143.84 | 141.20 | 143.18 | 89902 | 0 |
| 2005-02-28 | 143.70 | 147.19 | 139.47 | 143.14 | 124965 | 0 |
| 2005-03-01 | 141.79 | 142.57 | 131.36 | 134.50 | 229663 | 0 |
| 2005-03-02 | 135.27 | 139.11 | 135.06 | 137.56 | 88113 | 0 |
| 2005-03-03 | 138.59 | 138.59 | 132.24 | 134.89 | 120294 | 0 |

## 2.2 Objective

The main objective of this project is to handle and explore the Google stock market data set, also perform Data Pre-processing (explained in section 2.4.2) to the dataset. In the end do stock market prediction using linear regression (supervised machine learning technique).

## 2.3 Python Libraries

### Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.[5]

A DataFrame in Pandas is a labelled 2D data matrix that supports many operations on it for convenience. The Series object represents a 1D series of values and they are used when setting DataFrame rows [6]. The pandas name itself is derived from panel data, an econometrics term for multidimensional structured data sets, and Python data analysis itself.

### Numpy

Numpy is a python package providing a powerful N-dimensional array object, tools for integrating C/C++ and Fortran code. For numerical data, NumPy arrays are a much more efficient way of storing and manipulating data than the other built-in Python data structures. Also, libraries written in a lower-level language, such as C or Fortran, can operate on the data stored in a NumPy array without copying any data.

### Matplotlib

Matplotlib is the most common and popular Python library for producing plots and other 2D data visualization. It provides comfortable interactive environment for ploting sand exploring the data. The plots generated through matplotlib are interactive, you can zoom in to a particular section of the plot using toolbar in the plot window.

### Scikit-learn

Scikit-Learn is a general purpose machine learning library for Python.[7]Scikit-learn is an extention to SciPy. It provides machine learning algorithms for the following purposes.

- Algorithms for supervised and unsupervised learning.
- Built on SciPy and Numpy.
- Standard Python Api interface.
- Sits on top of c libraries, LAPACK, LibSVM, and Cython.
- Open Source: BSD license (part of Linux).[8]

## 2.4 Implementation

### 2.4.1 Data Retrieval

- Required packages are imported using the commands mentioned below.

```
#libs imported
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import mpld3
mpld3.enable_notebook()
%matplotlib inline
import sklearn as skl
import math
import datetime
import time
from pandas.tools.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

- Data is imported in pandas using the pd.read_csv() method, parsing date as the index, thus by transforming the the data set to pandas DataFrame, so that it could be used by the Regression Algorithm explained below.

```
#read data in the pandas by parsing the date as index column

df = pd.read_csv('gogl.us.txt',
                 header=0,
                 index_col='Date',
                 parse_dates=True)
```

### 2.4.2 Pre-processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.
Steps included in Data Pre-processing.

- Data Cleaning**:** The data is cleaned by removing null values, filling in missing values, smoothing the noise, or resolving inconsistency in the data.
- Data Intregration: The data with different representation are combined to resolve the conflict with the data.
- Data Transformation: The Data is normalized, aggregated and generalized.
- Data Reduction: This step aims to present a reduced representation of the data in a data warehouse.
- Data Discretization: Involves the reduction of a number of values of a continuous attribute by dividing the range of attribute intervals.[4]

  Following Data Pre-Processing was done to achieve the desired results and prepare the data to implement the required algorithm.

- In the data frame OpenInt column is removed which was containing null values by using del df['OpenInt'].
- To remove the inconsistency in the decimal values, float data types are restricted to only 2 decimal values.
  Below is the pandas comand used to achieve the same.

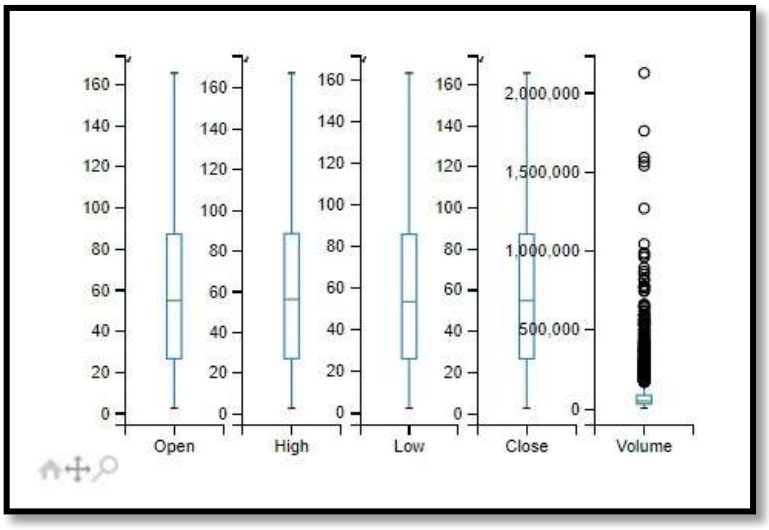
```
#float(dtype) to 2 decimal values to avoid error due to inconsistency in decimal values

df.Open = df.Open.round(2)
df.High = df.High.round(2)
df.Low = df.Low.round(2)
df.Close = df.Close.round(2)
```

- To get seamless visualization of matoplotlib plots, these libraries are imported. Thus providing a tool box below the plot enabling us to zoom on specific points in the plot and get a better understanding. Following mpld3 package was installed and following libraries were imported.

```
import mpld3
mpld3.enable_notebook()
%matplotlib inline
```

- Visualization of data using box plot, to get better understanding of the various attributes of the dataset. Box plot helps to explore the data much better, giving information regarding the mean of the

distribution. Here we can get the idea using the box plot regarding what has been the mean distribution of mentioned features in the google stock market.




### 2.4.3 Feature Selection

- **Percentage Change Implementation**
  To summarize the values of the stock attributes, Percentage change is calculated. Where OC_change is the percentage change between opening and closing values of the stock prices on the particular day and HL_change is the percentage change between the highest and the lowest values of the stock price on the particular day.

```
#Calculated the percentage change between the attributes

df['OC_change'] = (df['Open'] - df['Close'])/ df['Close']*100
df['HL_change'] = (df['High'] - df['Low'])/ df['Low']*100
```

- **Close Feature**
  Closing price generally refers to the last price at which a stock trades during a regular trading session.

- **Volume Feature**
  Volume is one of the most basic and beneficial concepts to understand when trading stocks. Volume is defined as, "the number of shares or contracts traded in a security or an entire market during a given period of time." What this means is that each time a person sells or buys shares of a stock, that is considered volume. [3]

- **Label**
  Close feature is shifted to forecast_out the label output.

```
forecast_out =  int(math.ceil(0.01*len (df)))
df['label'] = df[forecast_col].shift(-forecast_out)
```

Null Values are dropped using df.dropna(inplace=True)

### 2.4.4 Algorithm Implementation

X,Y labels are determined to implement the algorithm. Where X contains OC_change, HL_Change, Close, Volume and Y contains the output label.

```
X = np.array(df.drop(['label'],1))
Y = np.array(df['label'])
```

- Splitting the features and labels into test and train set, with test_size=0.2.
- x_train and y_train were fitted into the model.

```
clf.fit(x_train, y_train)

LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

- Stock prices are predicted for the Forecast_set.

```
#Predicting the stocks for the forecasted set
X = X[:-forecast_out]
X_lately = X[-forecast_out:]
Forecast_set = clf.predict(X_lately)
```

# III. Results and Analysis

## 3.1 Accuracy

The accuracy of the algorithm achieved was 93.34%.

```
#Accuracy of the algorithm
accuracy=clf.score(x_test, y_test)
accuracy
0.9334180049396671
```

## 3.2 Stock Prices Prediction

The forecast set contains the predicted stock prices. We were able to predict the stock prices for the following 33 days .

Below is a Snipped image of the Forecast set obtained after the implementation of the alogorithm.

```
df['Forecast'].tail(30)
```

```
Date
2017-12-04 23:00:00    7.711381
2017-12-05 23:00:00    7.572278
2017-12-06 23:00:00    7.173754
2017-12-07 23:00:00    7.231440
2017-12-08 23:00:00    6.909222
2017-12-09 23:00:00    7.354164
2017-12-10 23:00:00    6.698325
2017-12-11 23:00:00    7.737709
2017-12-12 23:00:00    7.340172
2017-12-13 23:00:00    8.199028
2017-12-14 23:00:00    8.213111
2017-12-15 23:00:00    8.603341
2017-12-16 23:00:00    8.018540
2017-12-17 23:00:00    7.969536
2017-12-18 23:00:00    8.481410
2017-12-19 23:00:00    8.177767
2017-12-20 23:00:00    8.625956
2017-12-21 23:00:00    8.587463
2017-12-22 23:00:00    7.844227
2017-12-23 23:00:00    8.754640
2017-12-24 23:00:00    8.336629
2017-12-25 23:00:00    8.719874
2017-12-26 23:00:00    8.803204
2017-12-27 23:00:00    8.665573
2017-12-28 23:00:00    8.815172
2017-12-29 23:00:00    8.523897
2017-12-30 23:00:00    9.133596
2017-12-31 23:00:00    8.757476
2018-01-01 23:00:00    9.602385
2018-01-02 23:00:00    9.410234
Name: Forecast, dtype: float64
```

### 3.3 Visualization of Predicted Stock Prices

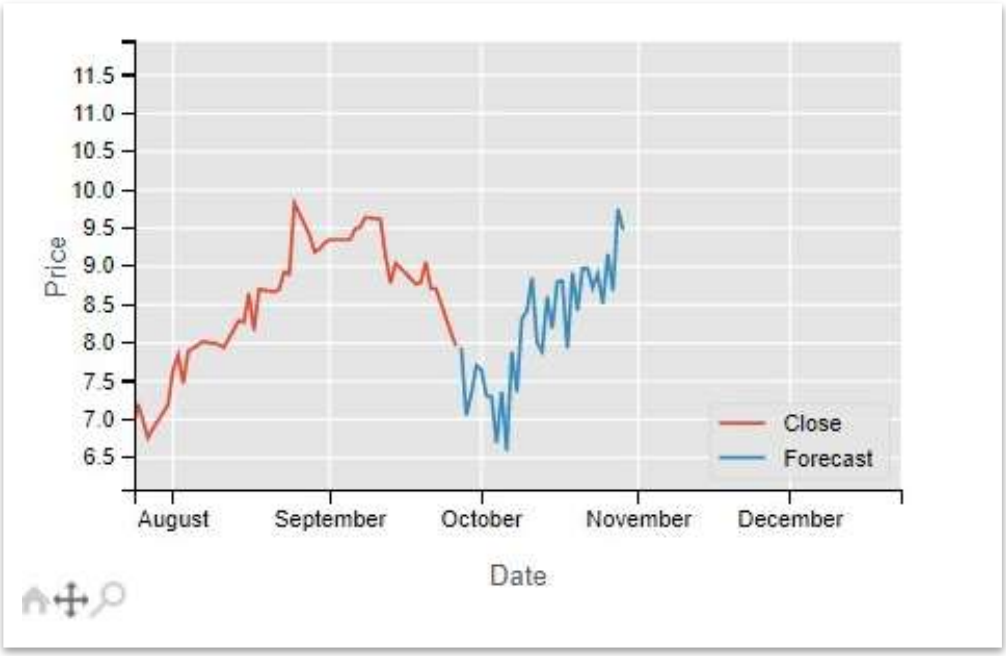
The predicted stock prices are visualized using the ggplot. Where X axis contains the predicted stock prices and Y axis contains the date. Here blue color represents the forecasted stock prices and red color represents the closing prices.

Below is mentioned the command used to plot the ggplot.

```
style.use('ggplot')

df['Forecast'] = np.nan

last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day

for i in Forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]

df['Close'].plot()
df['Forecast'].plot()
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
```

ggplot Code

Below is the snipped image of the ggplot, the plot was zoomed in using the tool enabled through the packages installed and the respective libraries imported, It was explained in the section 2.3.2.




Stock Price Prediction

## IV. Conclusion and Future Outcomes

As per the prediction we have achieved, the trend of the stocks goes down in late September and then a rapid increase can be seen in mid-October which carries over until the end of October. We have predicted the stock prices of October 2017. With the help of linear regression, we have achieved a fairly good accuracy and avoided over fitting of the model.

In continuity to this project, different algorithms such as SVM, Random Forest and Naves Bayes can also be applied, and the results can be compared with linear regression. Some other features can also be implemented to further enhance the model such as moving average and exponential moving average(EMA).

## V. Limitation and Challenges

- We had to pass the date attribute as index because, it was generating string type error while reading the data in the pandas.

```
header=0,
index_col='Date',
parse_dates=True
```

- As the plots generated have very large window size of the data, so the visualization was squeezed into a very small trend. We had to use the mpld3 package which actually allows us to zoom in and zoom out into the plots.
- As we know that we cannot predict the stock market accurately. There can be many factors which can affect the stock market such as political decisions, natural hazards etc.

# VI. Refrences


[1]. Mathworks, https://uk.mathworks.com/discovery/supervised-learning.html
[2]. Blastchart, http://www.blastchart.com/Community/IndicatorGuide/Overlays/LinearRegressionForecast.aspx
[3]. StockTrader, https://www.stocktrader.com/2006/03/21/volume-and-its-meaning/
[4].Techopedia, https://www.techopedia.com/definition/14650/data-preprocessing
[5].Pandas 0.22.0 documentation, https://pandas.pydata.org/pandas-docs/stable/
[6]. Pandas.pydata.org. (2017). Python Data Analysis Library — pandas: Python Data Analysis Library. [online] Available at: http://pandas.pydata.org/ [Accessed 11 Mar. 2017].
[7]. F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: Journal of Machine Learning Research 12 (2011), pp. 2825–2830.
[8]. https://qmplus.qmul.ac.uk/pluginfile.php/1295712/mod_resource/content/1/bc_ml_type2_scikit_first.pdf
[9].ResearchGate.https://www.researchgate.net/post/What_are_the_advantages_of_using_linear_regression_model_over_the_quadratic_regression_while_checking_linearity_in_a_quantitative_analysis

## VII. Appendices

### 1. CODE

```
# coding: utf-8
# In[146]:
#libs imported
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
import mpld3
mpld3.enable_notebook()
get_ipython().run_line_magic('matplotlib', 'inline')
import sklearn as skl
import math
import datetime
import time
from pandas.tools.plotting import scatter_matrix
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# In[147]:
#read data in the pandas by parsing the date as index column
df = pd.read_csv('gogl.us.txt',
                 header=0,
                 index_col='Date',
                 parse_dates=True)
# In[148]:
df.head()
# In[149]:
# df.drop(['OpenInt'], axis = 1).head()
del df['OpenInt']
# In[150]:
df.head()
# In[151]:
# visualzation of data using box plot
df.plot(kind='box',subplots=True,layout=(1,5),sharex=False,sharey=False)
plt.show()
# In[152]:
#data types analysed
df.dtypes
# In[153]:
#float(dtype) to 2 decimal values to avoid error due to inconsistency in decimal values
df.Open = df.Open.round(2)
df.High = df.High.round(2)
df.Low = df.Low.round(2)
df.Close = df.Close.round(2)
# In[154]:
```

```
#Calculated the percentage change between the attributes
df['OC_change'] = (df['Open'] - df['Close'])/ df['Close']*100
df['HL_change'] = (df['High'] - df['Low'])/ df['Low']*100
# In[155]:
df.tail()
# In[156]:
df.head()
# In[157]:
#populating the dataframe by the required fields
df = df[['OC_change','HL_change','Close','Volume']]
# In[158]:
#defining the features and lables
#shift the close feautre
forecast_col = 'Close'
forecast_out =  int(math.ceil(0.01*len (df)))
df['label'] = df[forecast_col].shift(-forecast_out)
df['label'].tail()
# In[159]:
#null values handled
df.dropna(inplace = True)
# In[160]:
df.tail()
# In[161]:
#Feature Selection
X = np.array(df.drop(['label'],1))
Y = np.array(df['label'])
# In[162]:
len(X)
len(Y)
# In[163]:
#dividing the features and lables into test and train set
x_train, x_test, y_train, y_test = train_test_split(X,Y,test_size = 0.2)
# In[164]:
#linear regression
clf = LinearRegression()
# In[165]:
clf.fit(x_train, y_train)
# In[166]:
#Accuracy of the algorithm
accuracy=clf.score(x_test, y_test)
accuracy
# In[167]:
#Predicting the stocks for the forecasted set
X = X[:-forecast_out]
X_lately = X[-forecast_out:]
Forecast_set = clf.predict(X_lately)
```

```
# In[168]:
#Predicted stock prices
Forecast_set
# In[169]:
style.use('ggplot')
df['Forecast'] = np.nan
last_date = df.iloc[-1].name
last_unix  = last_date.timestamp()
one_day = 86400
next_unix = last_unix+one_day
for i in Forecast_set:
    next_date = datetime.datetime.fromtimestamp(next_unix)
    next_unix += one_day
    df.loc[next_date] = [np.nan for _ in range(len(df.columns)-1)]+[i]
df['Close'].plot()
df['Forecast'].plot()
plt.legend(loc=4)
plt.xlabel('Date')
plt.ylabel('Price')
plt.show()
# In[170]:
df['Forecast'].tail(30)
# In[171]:
df.tail()
```

## 2. DATA :

The data of google stock can be accessed using this google drive link.
https://drive.google.com/open?id=1H_bbMGqQ3qVXQXaiwEB_TuD5LbXhn085