## **Stock Price Prediction**

Akshay Gavandi [gavandi.a@husky.neu.edu](mailto:gavandi.a@husky.neu.edu) | Ajinkya Kunjir [kunjir.a@husky.neu.edu](mailto:kunjir.a@husky.neu.edu)

Kruti Lakhani [lakhani.kr@husky.neu.edu](mailto:lakhani.kr@husky.neu.edu) | Neelam Babel [babel.n@husky.neu.edu](mailto:babel.n@husky.neu.edu)

## **Abstract**

Stock price prediction is the process of trying to predict the future prices of a stock or any other financial instrument traded on an exchange. Stock price prediction has been a strong topic of discussion with many trying to predict prices to yield profits. Initially a far-fetched dream, data science has helped make huge strides in stock price prediction. The purpose of the project is to analyze and understand the advantages and disadvantages of various algorithms applicable. The dataset used is the yahoo stock data ranging over a period of nearly 2 decades. The closing price of the yahoo stock was predicted using multiple algorithms which included: - Moving Average Model - Autoregression Model - ARIMA, a time-series model and the Regression Models.

Index Terms—Stock Price, Time series, Auto-Regression, Moving Average, ARIMA, Regression

## **Introduction**

The stock market process is full of uncertainty, and fluctuations and is affected by myriad factors. Hence stock market prediction remains one of the hot areas of research for it’s importance in finance and business. The analysis is done using historical data of stock prices of different companies. The advent of Machine Learning and eventually Deep Learning proved out to be a massive push in this field resulting in companies hiring Data Scientists for Time Series Forecast and Analysis. Generally, the stock price data shows extensive variations owing to several factors like the government, financial situation of the company, actions of the competitors, etc. The challenge in hand of a Data Scientist is to take in account all the possible factors and predict the best possible price of the stock. Nowadays, the prediction model/system are advances enough that they help the Data Scientist by themselves making well informed choices and providing quick results which would generally require extensive calculations. Stock price prediction using Machine Learning depends on several factors including Trend, Momentum, Volume and Autocorrelation. A good Machine Learning model would take care of each of the factor and then predict the price. This would result in the best possible prediction. Several methods are available to analyze and take care of them.

**Dataset**

•The Historical data of the stock market is extracted for the years 2008 - 2019 using web scraping method on Yahoo finance website

•There are multiple variables in the dataset – date, open, high, low, close, Adj Close and Volume

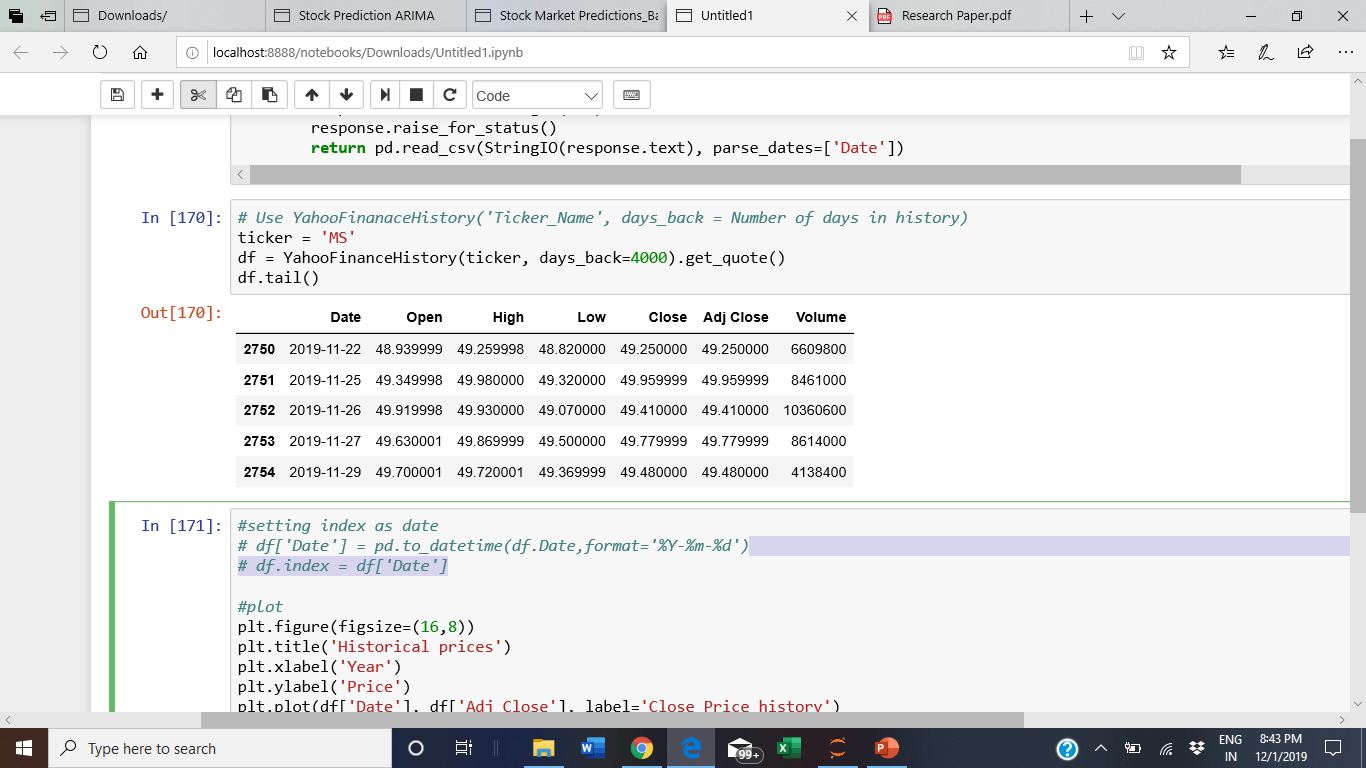
•The columns *Open* and *Close* represent the starting and final price at which the stock is traded on a day

•*High*, *Low* and Adj Close represent the maximum, minimum, and last closing price of the share for the day

•Volume is the number of shares bought or sold in the day

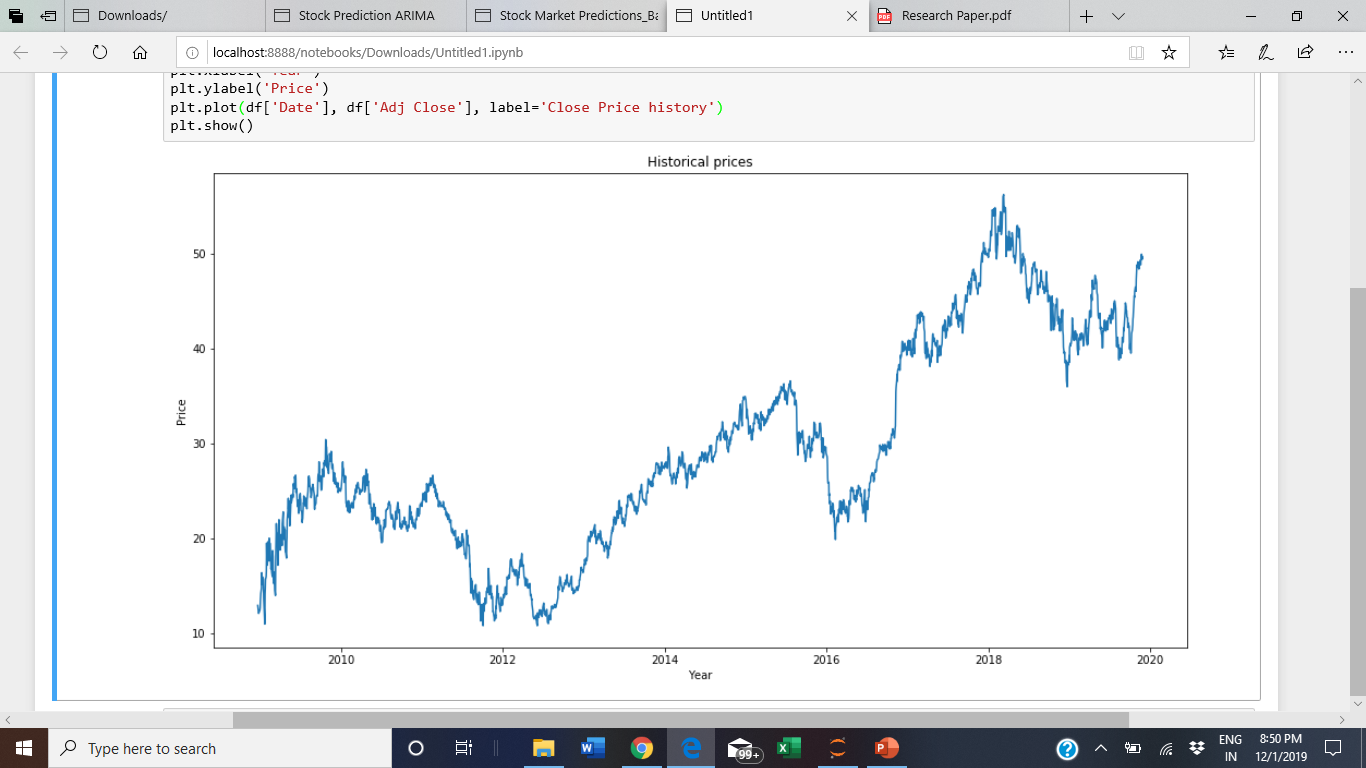
**Dependencies:**

* Pandas — to read the CSV file
* Numpy — perform calculations on data
* Scikit learn — build the predictive models
* Matplotlib — visualise the output

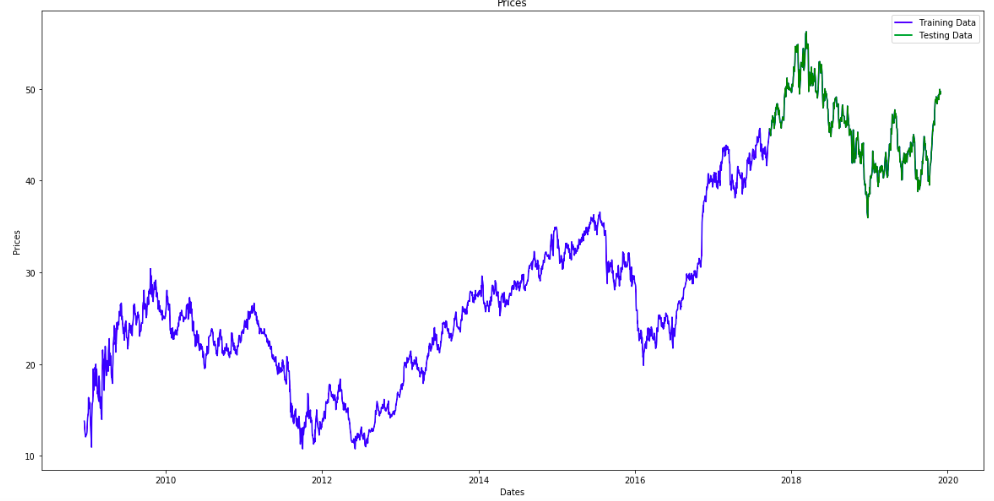
****

**Data Exploration**

When we take a look at the price movement over time by simply plotting the Closing price vs Time, we can already see that the price continuously increases over time and we can also estimate that trend could be linear.



Plot showing the historical prices from 2008-2019



Plot showing price distribution for training (blue) and test (green) data

In the upcoming sections, we try to explore these variables and use different techniques to predict the daily closing price of the stock

**Data Pre-processing**

Data Preprocessing can be split into 2 parts:

1. Trend, Seasonality and Stationarity Detection

2. Trend, Seasonality and Stationarity Elimination

Since time series is dependent on ‘time’, it can exhibit trend and seasonality, which in turn can make a time series non-stationary and have high autocorrelation which is undesirable. In this project, inconsistencies in the time series were detected by applying Dickey Fuller test to check the Stationarity, decomposing the time series in Residual, Trend and Seasonality and plotting ACF and PACF charts. A time series can be made stationary by applying Differencing and Transformation method. In this work, applying logarithmic function and taking the difference between the data points made the time series stationary and resulted in the p value reduced from 0.87 to 0.00.

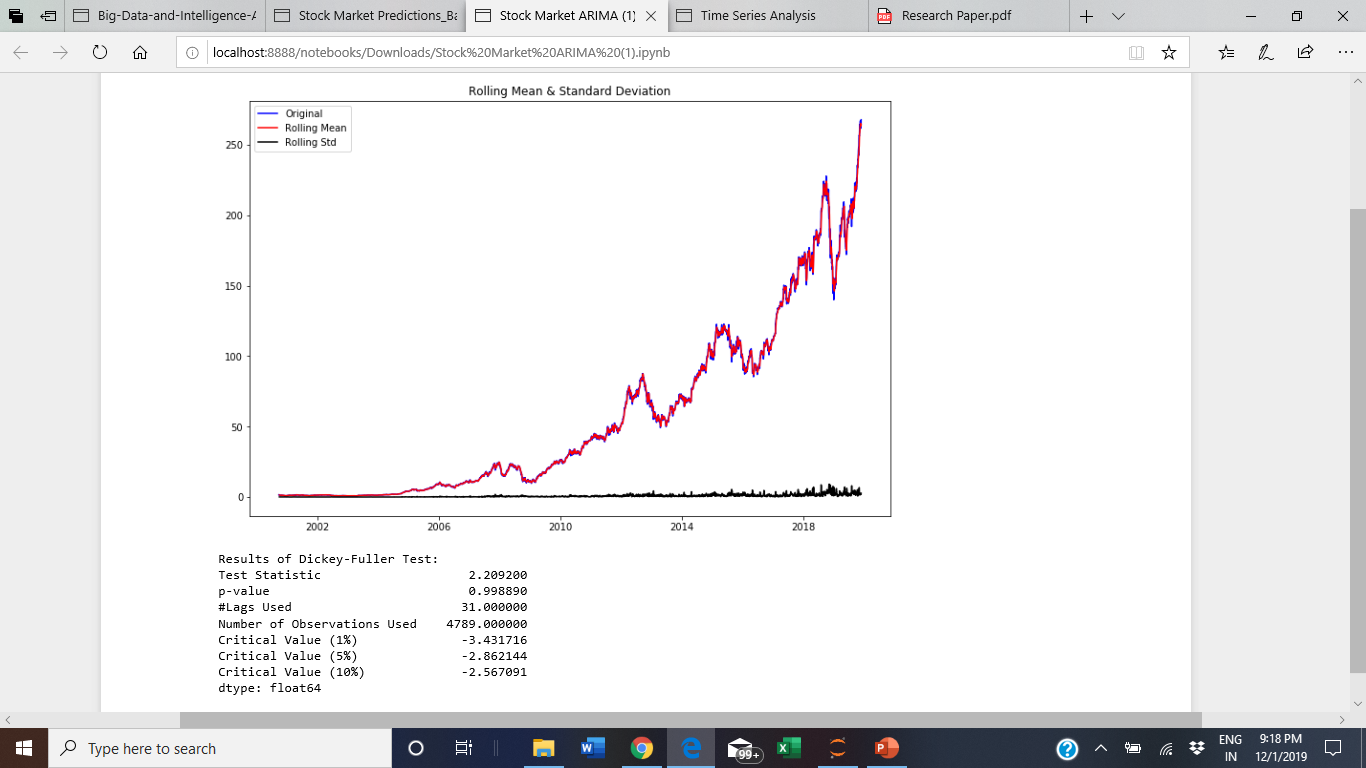
**CHECKING STATIONARITY OF TIME SERIES:**

A time series is said to be stationary if its statistical properties do not change with time. These properties, for example are mean, standard deviation, variation and autocorrelation. A non-stationary time series is a time series whose statistical properties vary over a period of time. A non-stationary time series first needs to be converted to a stationary time series. This can be done by removing the trend and seasonality from the time series so that further statistical analysis can be done on a detrended time series data.

We are using Dickey–Fuller test to check if our time series is stationary or not.

## **DICKEY–FULLER TEST:**

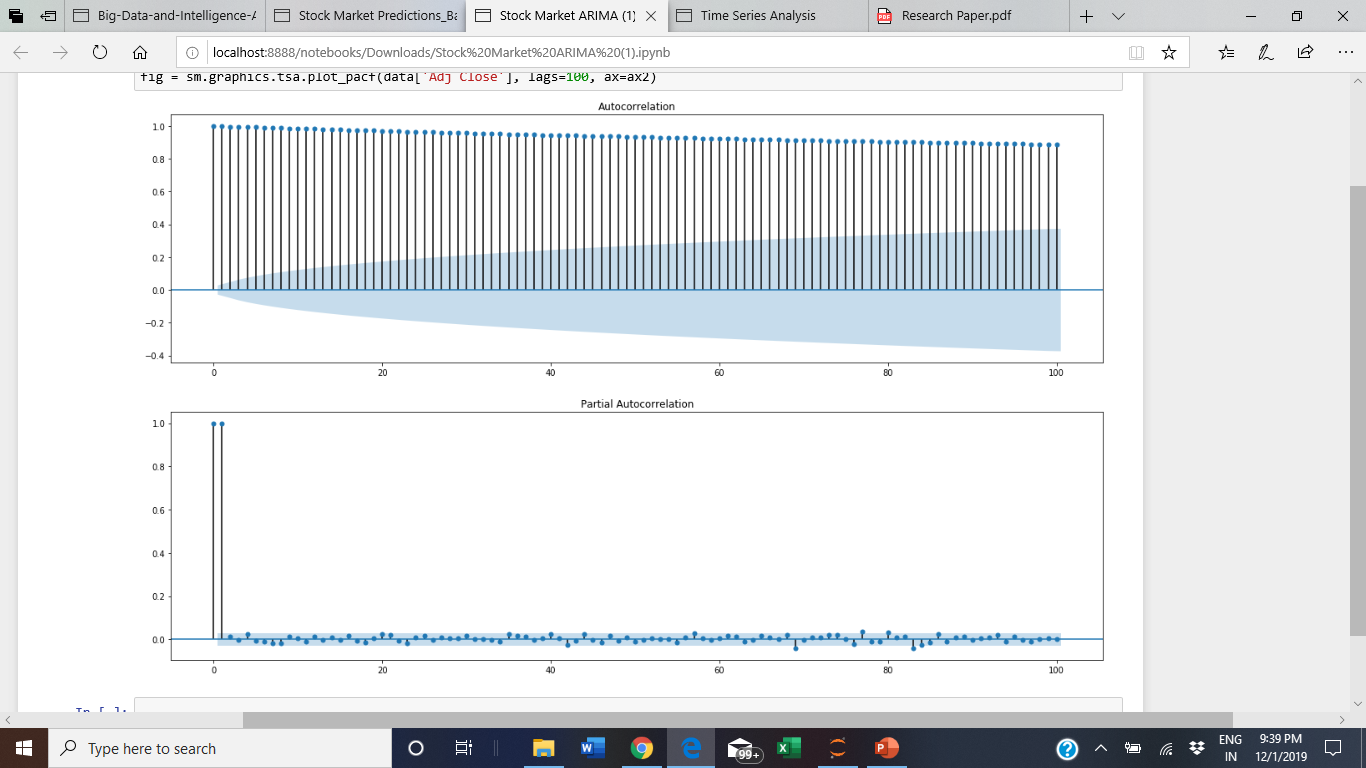
The test is named after American statisticians David Dickey and Wayne Fuller who developed the test in 1979. It is used to determine whether a unit root, a feature that can cause issues in statistical inference, is present in an autoregressive model. We reject the null when the p-value is less than or equal to a specified significance level of 0.05 (5%)

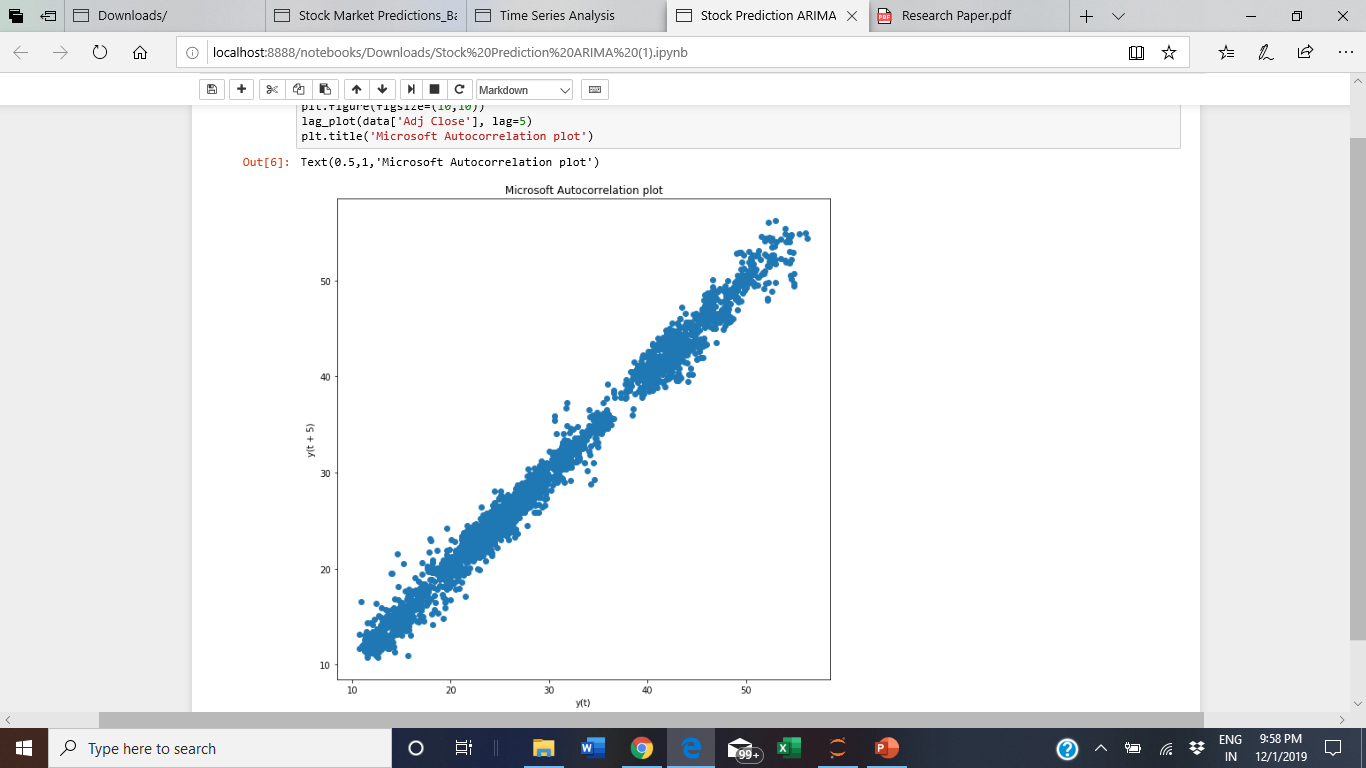


**ACF and PACF plots:**

ACF is an (complete) auto-correlation function which gives us values of auto-correlation of any series with its lagged values.PACF is a partial auto-correlation function. it finds correlation of the residuals with the next lag value hence ‘partial’ and not ‘complete’ as we remove already found variations before we find the next correlation.

Autocorrelation and partial autocorrelation plots are heavily used in time series analysis and forecasting.These are plots that graphically summarize the strength of a relationship with an observation in a time series with observations at prior time steps.





The results shown from the above plot show that ARIMA would be a good model to be applied to this type of data

**Model Architecture Design**

**Linear Regression**

Linear regression is a basic and commonly used type of predictive analysis. The overall idea of regression is to examine two things:

(1) does a set of predictor variables do a good job in predicting an outcome (dependent) variable?

(2) Which variables in particular are significant predictors of the outcome variable, and in what way do they–indicated by the magnitude and sign of the beta estimates–impact the outcome variable?

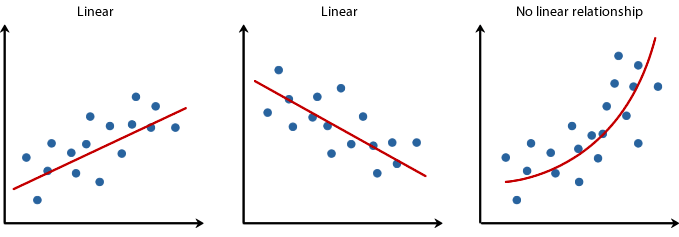
These regression estimates are used to explain the relationship between one dependent variable and one or more independent variables. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula

**y = c + b\*x**

where y = estimated dependent variable score, c = constant, b = regression coefficient, and x = score on the independent variable.

The three major use cases of regression analysis are:

1. the regression might be used to identify the strength of the effect that the independent variable(s) have on a dependent variable.
2. it can be used to forecast effects or impact of changes. That is, the regression analysis helps us to understand how much the dependent variable changes with a change in one or more independent variables.
3. regression analysis predicts trends and future values. The regression analysis can be used to get point estimates. A typical question is, “what will the price of gold be in 6 months?”



**Implementation**



**KNN (K Nearest Neighbours)**

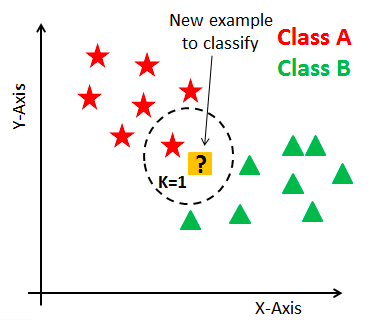
The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. The KNN algorithm hinges on this assumption being true enough for the algorithm to be useful. KNN captures the idea of similarity (sometimes called distance, proximity, or closeness) with some mathematics we might have learned in our childhood— calculating the distance between points on a graph. To select the K that’s right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm’s ability to accurately make predictions when it’s given data it hasn’t seen before.

**Advantages**

1. The algorithm is simple and easy to implement
2. There’s no need to build a model, tune several parameters, or make additional assumptions
3. The algorithm is versatile. It can be used for classification, regression, and search

**Disadvantage**

The algorithm gets significantly slower as the number of examples and/or predictors/independent variables increase.

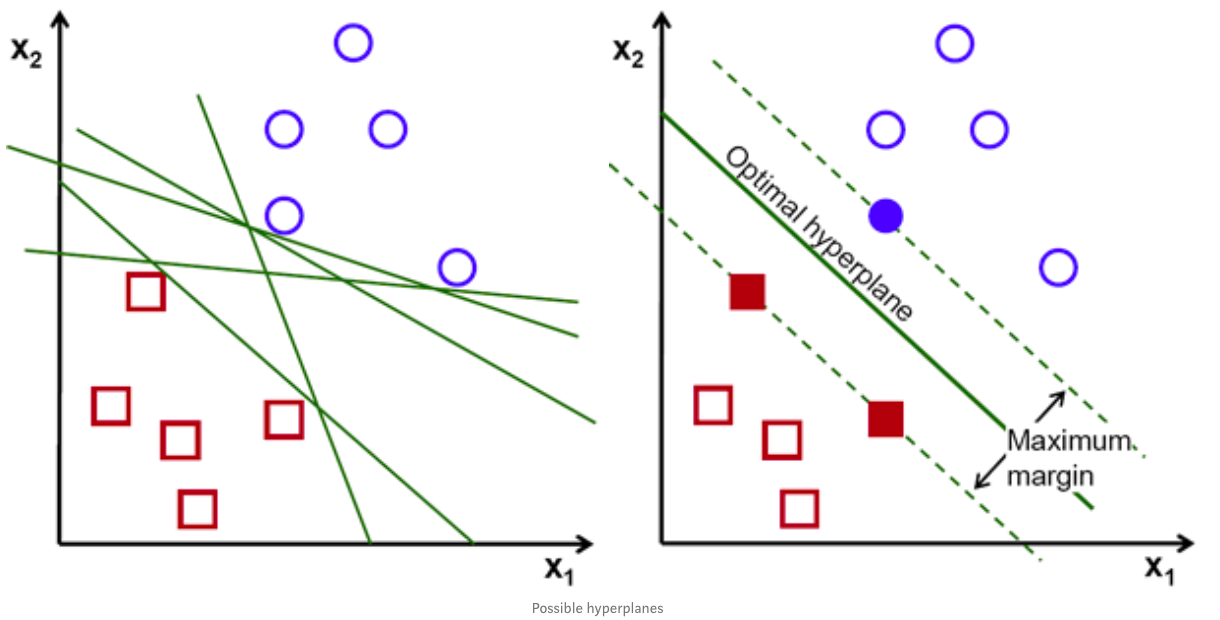


**Implementation**

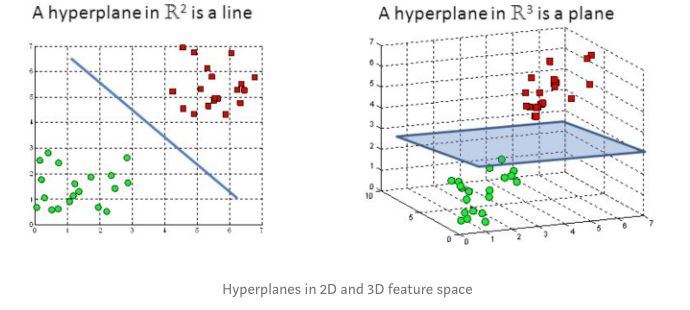


**SVM (Support Vector Machine)**

Support Vector Machine is used to find a Hyperplane in an N Dimensional Space (where N is the number of features) that distinctly classifies the data points.



Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.



Support vectors are data points that are closer to the hyperplane and influence the position and orientation of the hyperplane. Using these support vectors, we maximize the margin of the classifier. Deleting the support vectors will change the position of the hyperplane. These are the points that help us build our SVM.

**ARIMA**

ARIMA stands for Auto Regressive Integrated Moving Average.

**Auto Regressive** - The model that takes advantages of the connection between a predefined number of lagged observations and current one

### **Integrated -** Difference between raw observations

### **Moving Average** - The model that takes advantage of the relationship between residual error ad the observations.

The ARIMA model makes use of these three parameters :

p = number of lag observations

Purely autoregressive models resemble a linear regression where the predictive variables are P number of previous periods

d = the degree of differencing

d refers to the number of differencing transformations required by the time series to get stationary. Stationary time series is when the mean and variance are constant over time. It is easier to predict when the series is stationary.

q = the size of the moving average window.

Autocorrelation refers to how correlated a time series is with its past values whereas the ACF is the plot used to see the correlation between the points, up to and including the lag unit. In ACF, the correlation coefficient is in the x-axis whereas the number of lags is shown in the y-axis.

If there is a **Positive** autocorrelation at lag 1 then we use the **AR Model**

If there is a **Negative** autocorrelation at lag 1 the we use the **MA Model**

After plotting the ACF plot we move to Partial Autocorrelation Function plots (PACF). A partial autocorrelation is a summary of the relationship between an observation in a time series with observations at prior time steps with the relationships of intervening observations removed.

If the PACF plot drops off at lag n, then use an **AR(n) model** and if the drop in PACF is more gradual then we use the **MA term**

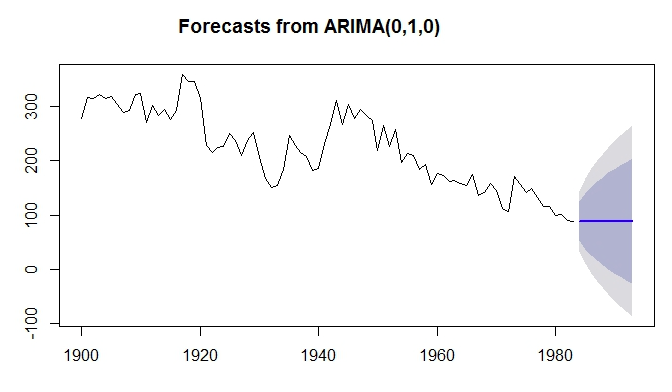
**Autoregressive component**: A purely AR model forecasts only using a combination of the past values sorta like linear regression where the number of AR terms used is directly proportional to the number of previous periods taken into consideration for the forecasting. Use AR terms in the model when :

* ACF plots show autocorrelation decaying towards zero
* PACF plot cuts off quickly towards zero
* ACF of a stationary series shows positive at lag-1

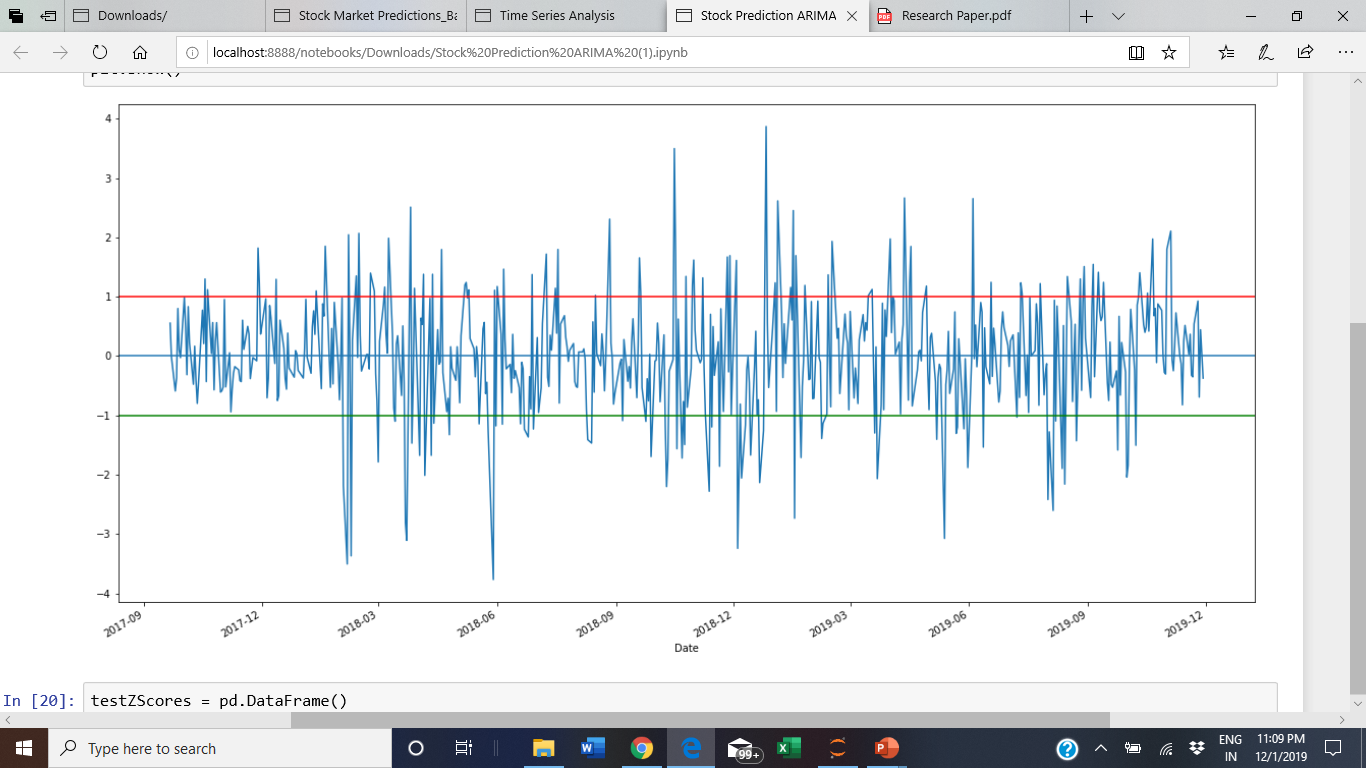
**Moving Averages**: Random jumps in the time series plot whose effect is felt in two or more consecutive periods. These jumps represent the error calculated in our ARIMA model and represent what the MA component would lag for. A purely MA model would smooth out these sudden jumps like the exponential smoothing method. Use MA terms in the Model when :

* Negatively Autocorrelated at Lag — 1
* ACF that drops sharply after a few lags
* PACF decreases more gradually

**Integrated component**: This component comes into action when the time series is not stationary. The number of times we have to difference the series to make it stationary is the parameter(i-term) for the integrated component



**Predicting Stock Market Movements**



Looking at the z-score chart, we can see that whenever the z-score feature gets too high, or too low, it tends to revert back. Let’s use +1/-1 as our thresholds for too high and too low, then we can use the following model to generate a trading signal:

* Ratio is buy (1) whenever the z-score is below -1.0 because we expect z score to go back up to 0, hence ratio to increase
* Ratio is sell(-1) when the z-score is above 1.0 because we expect z score to go back down to 0, hence ratio to decrease

Once we create functions and apply algorithm to our data assuming 1000$ as the initial amount, we can see that $1000 invested on 2017-09-22 gives us $1151.02 as returnsWe bought 22 shares at price of $45.17 with remaining amount as $6.37. If we had gone for a long term plan and removed the money at $49.48, the return would have been $1094.93

This mean we have upped the profit by 50% and we can say our trading recommendation algorithm performed 50% better than the market

We then tried to improve recommendations by using the optimal value of zthreshold and applied this trading algorithm on different stocks

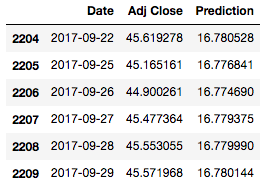
**Code with Documentation**

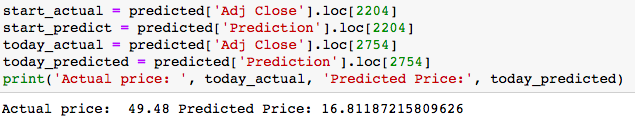
**Results**

Linear Regression -



Actual price vs Predictions



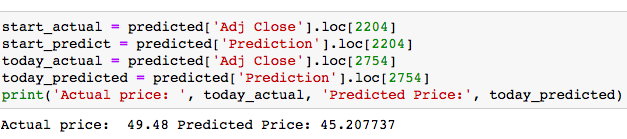


With Linear Regression, the accuracy of prediction is low, as can be seen with the above example. The actual closing price of the stock value is 49.48 whereas linear regression predicted the price to be 16.811

K-NN

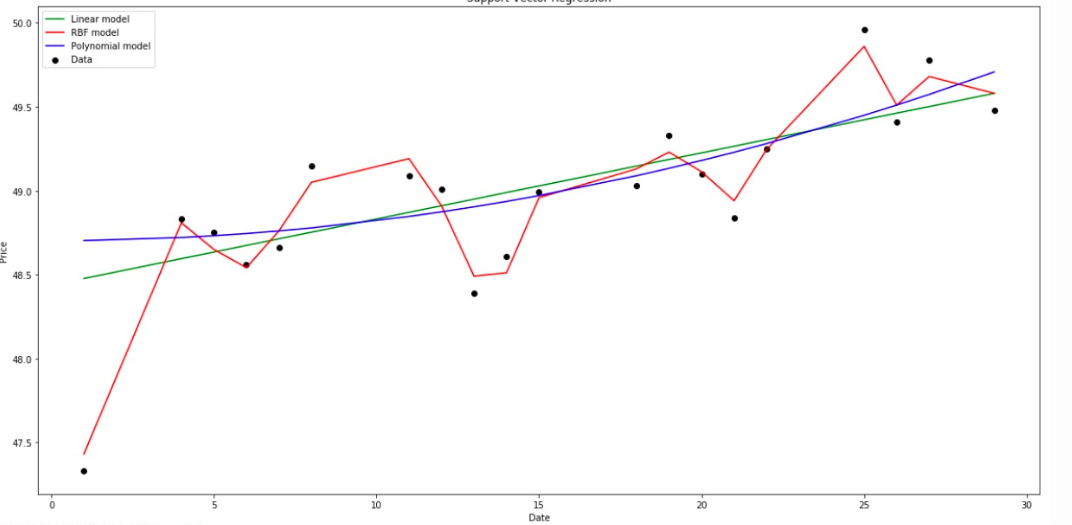


Actual price vs Predictions



With K-Nearest Neighbours, we get good predictions for some points while predictions are too off for the other points. This is due to Overfitting.

SVM -



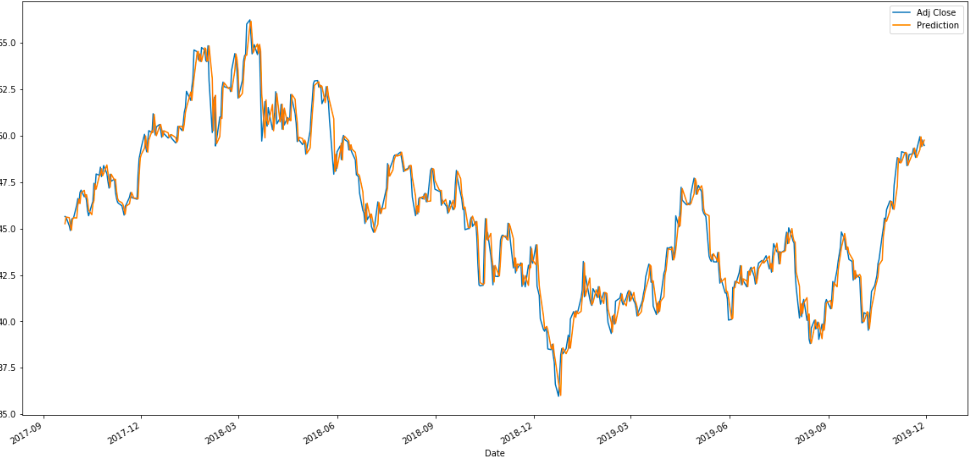
Support Vector Regression

Support Vector Regressor happens to be a better predictor than linear regression and K-NN, however the predictions still aren’t too accurate.

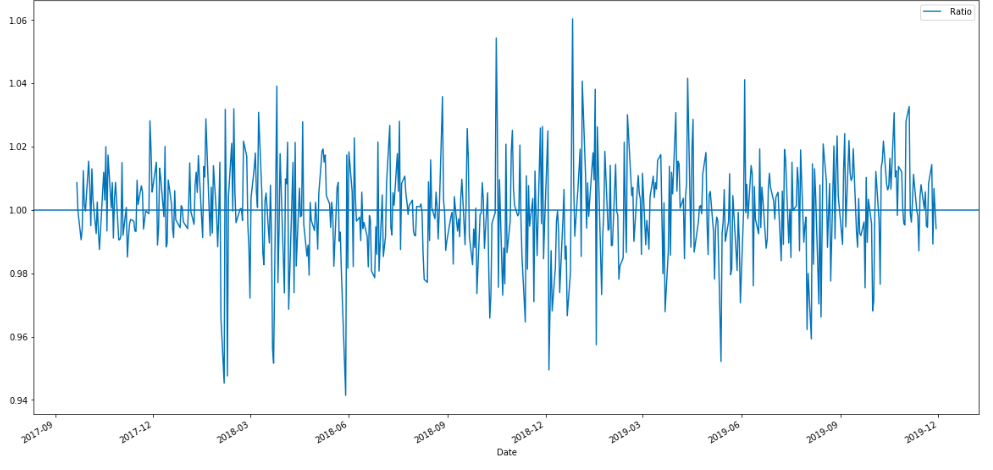
ARIMA -

****

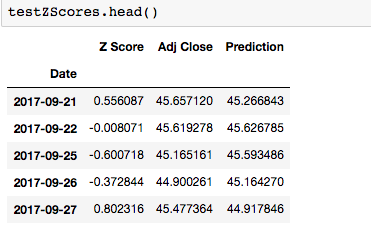
actual price vs predicted price

****

actual closing price vs predicted price



Z Score Graph



### **When to Buy, Sell or Hold**

**Buy** when stock is underpriced testZScores['Z Score'] <=-1.00

**Sell** when stock is overpriced testZScores['Z Score'] >= 1.00

**Hold** otherwise (testZScores['Z Score'] < 1.00) & (testZScores['Z Score'] > -1.00)



**Trading Algorithm**

We are creating a stock recommendation system that guides us to Buy, Sell or Hold stocks depending on whether the stocks are undervalued or overvalued. We recommend to buy in case the stock is undervalued than the predicted one otherwise in case it is overvalued, we sell.

The value of z threshold recommends this action.

We simulated the profits both by long term investment which is the standard way of investment in the market and using our algorithm.

The results show that our algorithm performs better.

1. Stock which increased in the given time
2. Stock which decreased in the given time
3. Stock that remained constant

Apple Inc. (AAPL) increase 72% since 2017. Using our trading algorithm, the increase in portfolio is almost 93%

Ford (F) reduced 10% since 2017. Using our trading algorithm, the increase in portfolio is 37%

General Motors (GM) didn’t change much -0.7% since 2017. Using our algorithm, the increase in portfolio is 80%

The results have proven that the algorithm works better in providing results than going for long term investments.

**Future Scope**

We live in a social media age and hence we cannot completely ignore the impact social media platforms like Twitter can have on stock price fluctuations for any company. The stocks of any company see an increase/decrease based on the tweets of influential people, and that inturn influences people in investing(buy/sell) on stocks. As a future scope to our implementation, we can expand the research to extract tweets for ‘yahoo-stocks’ by influential people and perform a sentiment analysis to see how positive or negative sentiments impact the trading.

**Acknowledgement**

We would like to show our gratitude to Professor Handan Liu and all the Teaching Assistants for guiding and encouraging us throughout the project

**References**

<https://towardsdatascience.com/time-series-forecasting-arima-models-7f221e9eee06>

<https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47>

<https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/>

<https://medium.com/predictive-analytics-for-stock-market-prediction/machine-learning-algorithm-for-stock-prediction-e75c678d05c7>

<https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm>

<https://scikit-learn.org/stable/modules/svm.html>

<https://medium.com/auquan/pairs-trading-data-science-7dbedafcfe5a>

<https://medium.com/auquan/pairs-trading-data-science-7dbedafcfe5a>

<https://datascienceplus.com/knn-classifier-to-predict-price-of-stock/>