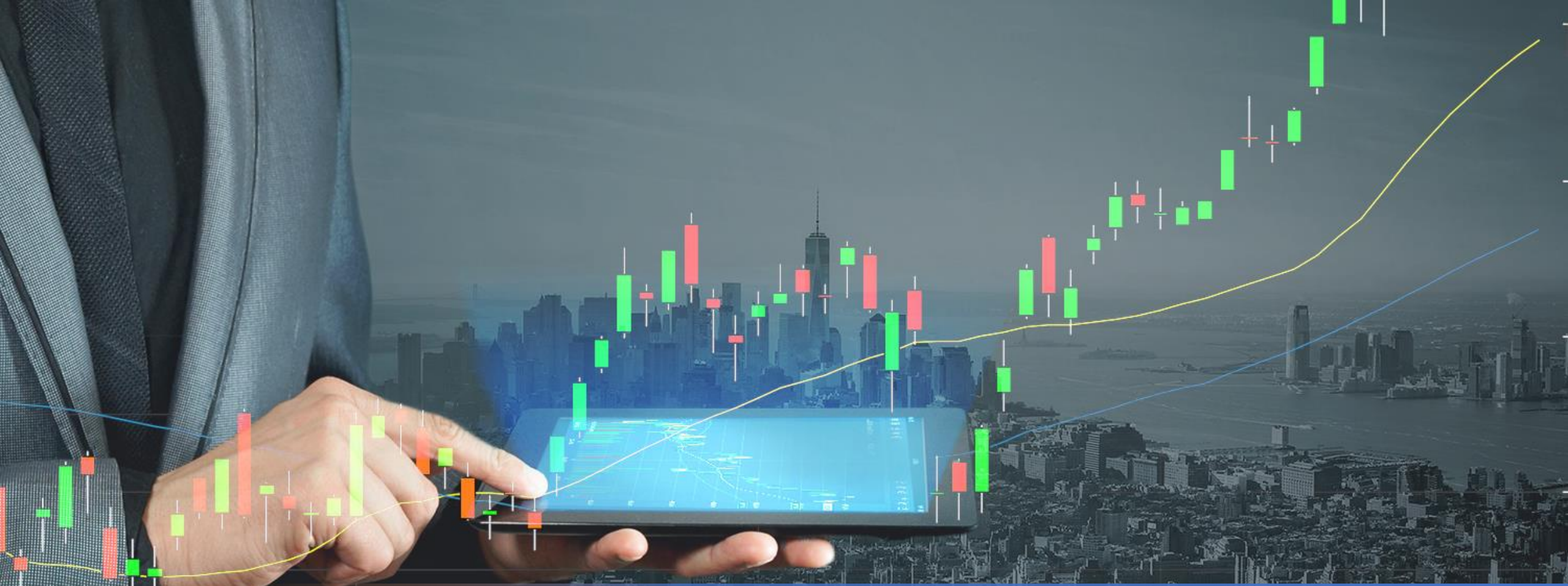


Stock Market Prediction

Akshay Gavandi
001825055

Ajinkya Kunjir
001869353

Kruti Lakhani
001883036

Neelam Babel
001837464

Agenda

- Introduction
- Machine Learning in stock prediction
- Objective
- Step by step process
- Data Acquisition and Dataset
- Linear Regression
- K-Nearest Neighbor
- Support Vector Machine (SVM)
- ARIMA
- When to buy, sell or hold ?
- Trading Simulation Model
- Conclusion
- References

Introduction

- Stock price prediction is the process of predicting 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
- Data mining techniques have been successfully shown to generate high forecasting accuracy of stock price Movement

Machine learning in Stock Prediction

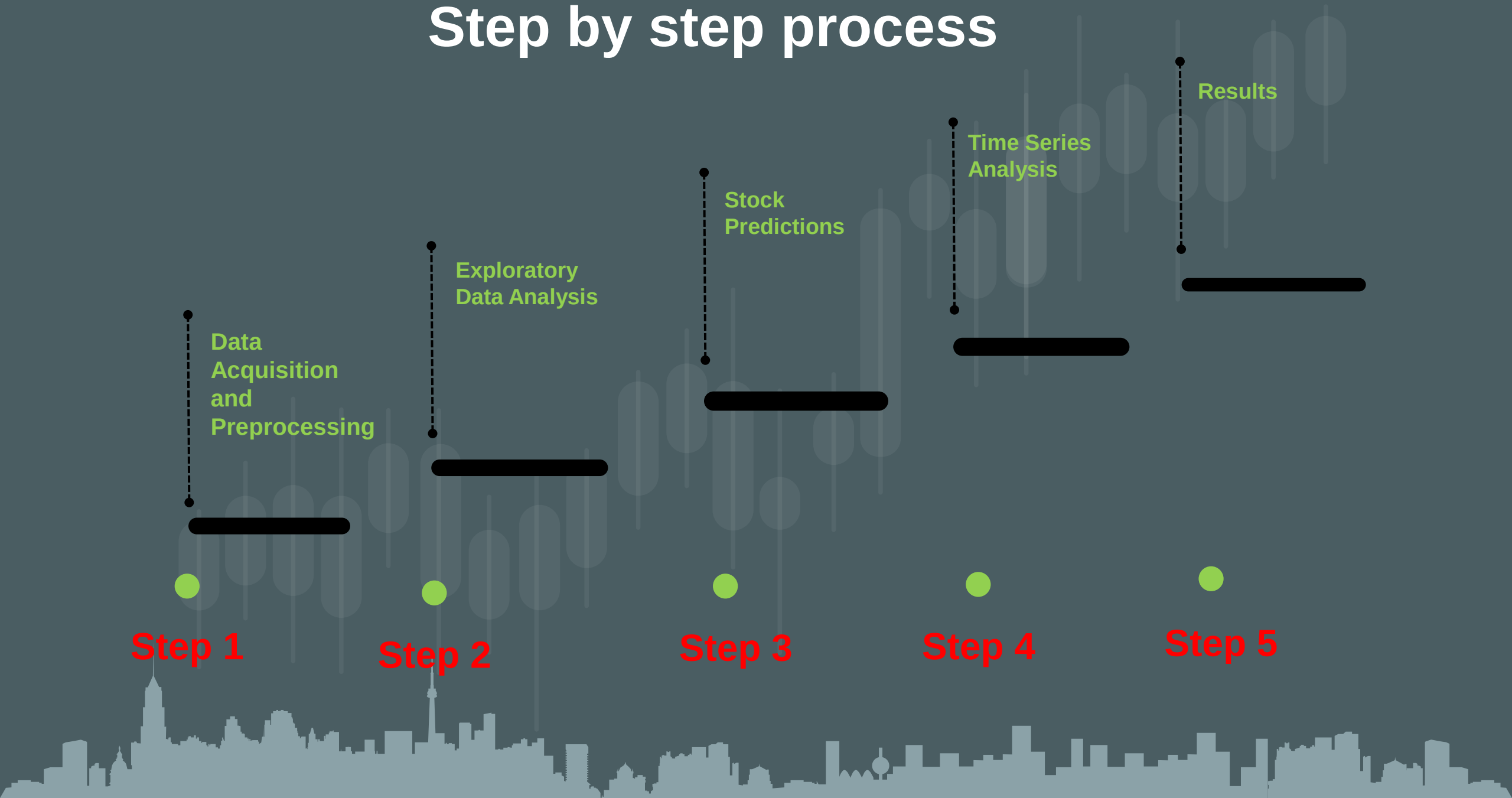
- The field of machine learning is vast and plays a key role in a wide range of critical applications
- Predicting the stock market involves predicting the closing prices of a company's stock for any given number of days ahead
- The advent of Machine Learning and eventually Deep Learning proved out to be a massive push in this field resulting in companies hiring Data Scientists for Time Series Forecast and Analysis
- Machine learning has the potential to ease the whole process by analyzing large chunks of data, spotting significant patterns and generating a single output that navigates traders towards a decision based on predicted asset prices

Objective

- To analyze and understand advantages and disadvantages of various algorithms applicable on the dataset

- To answer the questions like:
  - Can I predict the price of a stock and if yes, to what extent?
  - When to buy, hold or sell a stock?

Step by step process



Data Acquisition and Dataset

```
data.head()
```

Date	Open	High	Low	Close	Adj Close	Volume
2008-12-17	15.52	17.930000	14.80	16.50	13.790027	65914400
2008-12-18	16.82	17.680000	16.10	16.23	13.564372	34326400
2008-12-19	16.33	17.290001	15.45	15.45	12.912481	36407100
2008-12-22	15.62	15.630000	14.54	14.58	12.185370	19350500
2008-12-23	14.73	15.090000	14.40	14.44	12.068365	12686200

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

- Following are the columns in the scrapped data:
  - Open: Starting price of the stock on a specific day
  - Close: Final price of the stock on a specific day
  - High: Maximum price of the share on a specific day
  - Low: Minimum price of share on a specific day
  - Adj Close: Last closing price of share on a specific day
  - Volume: Number of shares bought or sold in a day

Linear Regression

- A Linear Regression Line is a straight line that best fits the prices between a starting price point and an ending price point. A "best fit" means that a line is constructed where there is the least amount of space between the price points and the actual Linear Regression Line

- The linear regression model returns an equation that determines the relationship between the independent variables and the dependent variable

- The equation for linear regression can be written as: **$Y = \beta 0 + \beta 1 X$**
  where Y is the predicted value of dependent variable
  β is the y-intercept
  $\beta 1$ is the slope
  X is the value of the independent variable

For our dataset, we are considering “X” as Date and “Y” to be Closing Price

Predicted VS Actual Price Plot

Results

The results clearly states that Linear Regression does not work efficiently for our purpose

```
start_actual = predicted['Adj Close'].loc[2204]
start_predict = predicted['Prediction'].loc[2204]
today_actual = predicted['Adj Close'].loc[2754]
today_predicted = predicted['Prediction'].loc[2754]
print('Actual price: ', today_actual, 'Predicted Price:', today_predicted)
```

```
Actual price:  49.48 Predicted Price: 16.81187215809626
```

	Date	Adj Close	Prediction
2204	2017-09-22	45.619278	16.780528
2205	2017-09-25	45.165161	16.776841
2206	2017-09-26	44.900261	16.774690
2207	2017-09-27	45.477364	16.779375
2208	2017-09-28	45.553055	16.779990
2209	2017-09-29	45.571968	16.780144
2210	2017-10-02	46.281521	16.785904
2211	2017-10-03	46.622101	16.788669
2212	2017-10-04	46.366669	16.786596
2213	2017-10-05	46.953228	16.791358
2214	2017-10-06	47.076218	16.792356
2215	2017-10-09	46.716717	16.789438
2216	2017-10-10	46.820782	16.790282
2217	2017-10-11	46.527496	16.787901
2218	2017-10-12	45.969318	16.783370
2219	2017-10-13	45.694958	16.781142

K-Nearest Neighbor

```
start_actual = predicted['Adj Close'].loc[2204]
start_predict = predicted['Prediction'].loc[2204]
today_actual = predicted['Adj Close'].loc[2754]
today_predicted = predicted['Prediction'].loc[2754]
print('Actual price: ', today_actual, 'Predicted Price:', today_predicted)
```

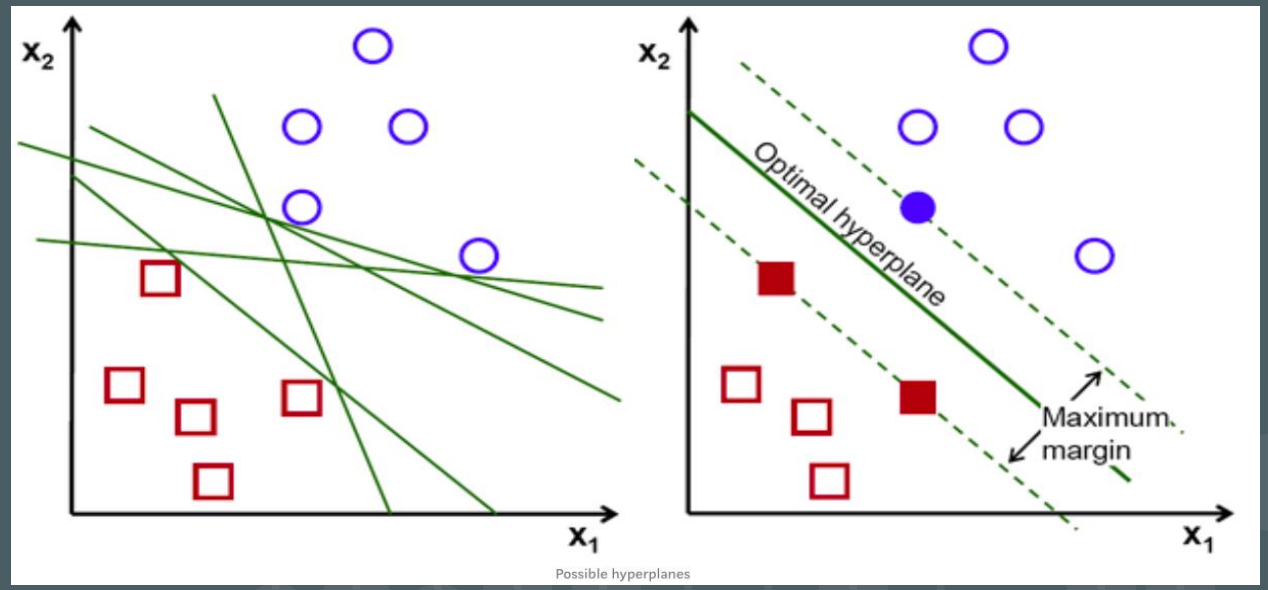
```
Actual price:  49.48 Predicted Price: 45.207737
```

- There’s a reason k-nearest neighbors is more useful for classification problems and small-scale regression.
- This appears to be a classic case of overfitting. Because KNN is just calculating distances from each point to another, it was completely unable to figure out the trend of where the prices are going

K-Nearest Neighbor

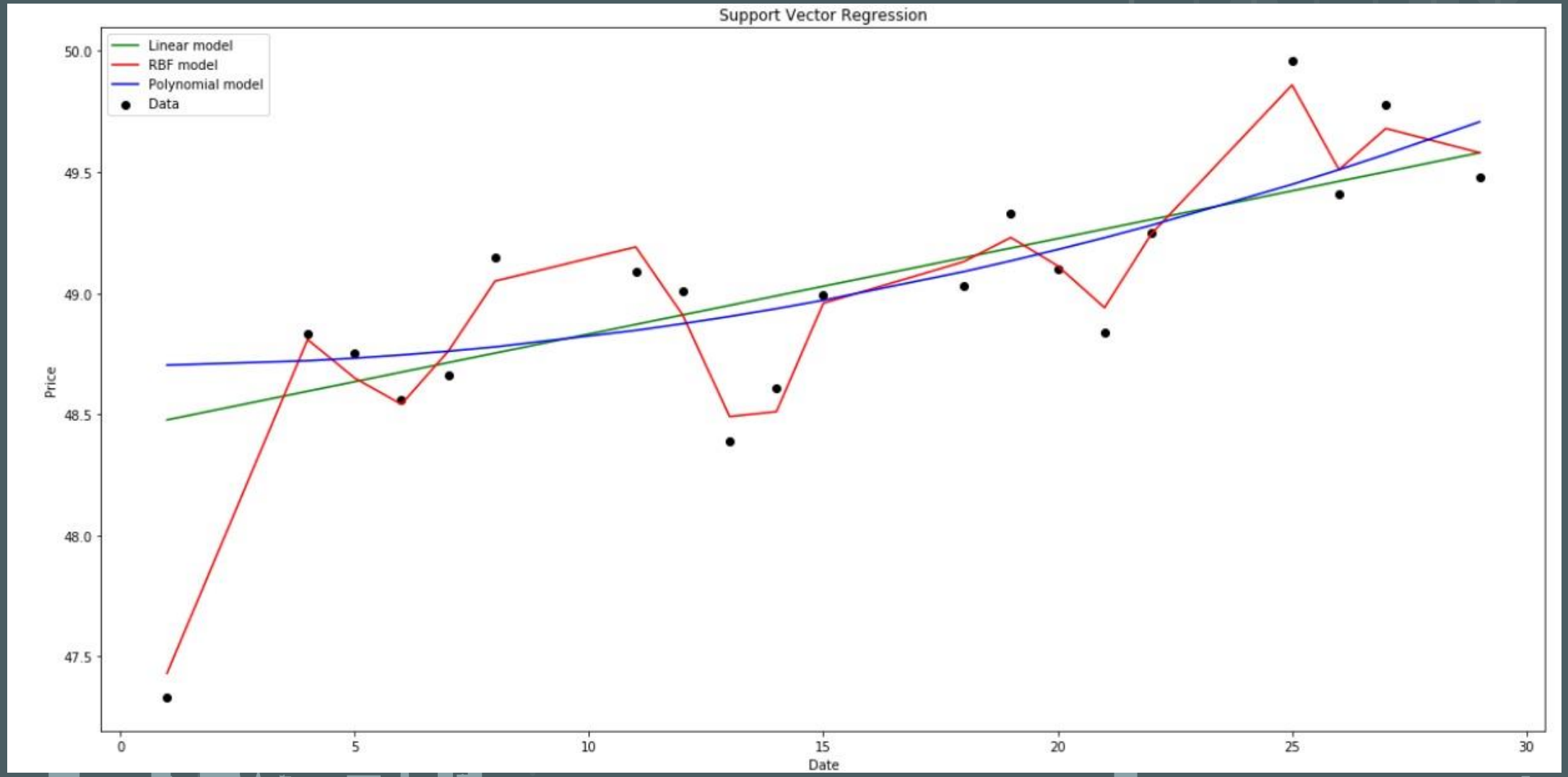
The results states that even K-Nearest Neighbor does not work efficiently for our purpose as it gives close prediction value only for some points

Support Vector Machine (SVM)



- 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.
- 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.

Support Vector Machine (SVM)



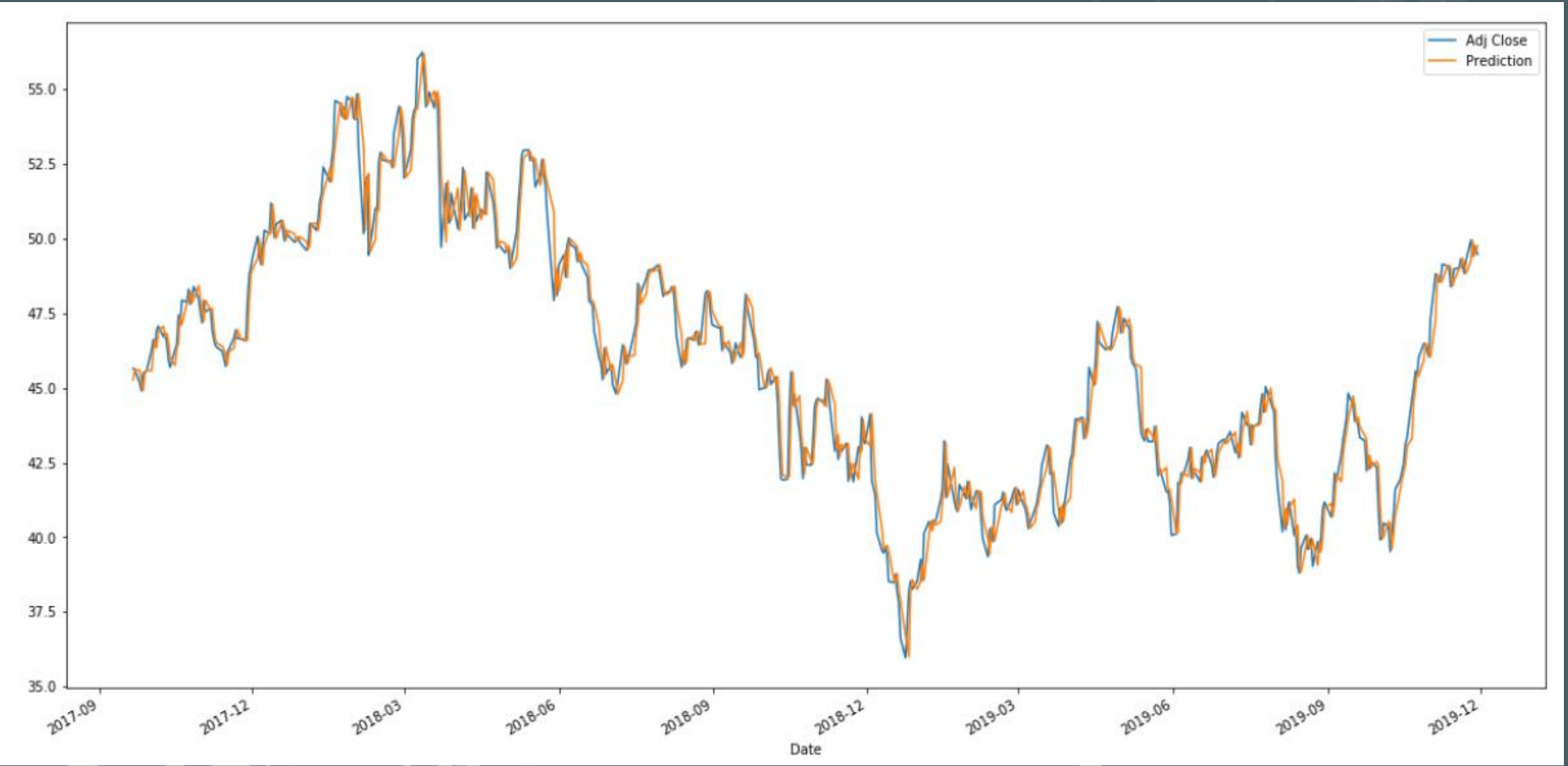
ARIMA

- Arima (Auto Regressive Integrated Moving Average) models are widely used in time series forecasting which provides complementary approaches to the problem

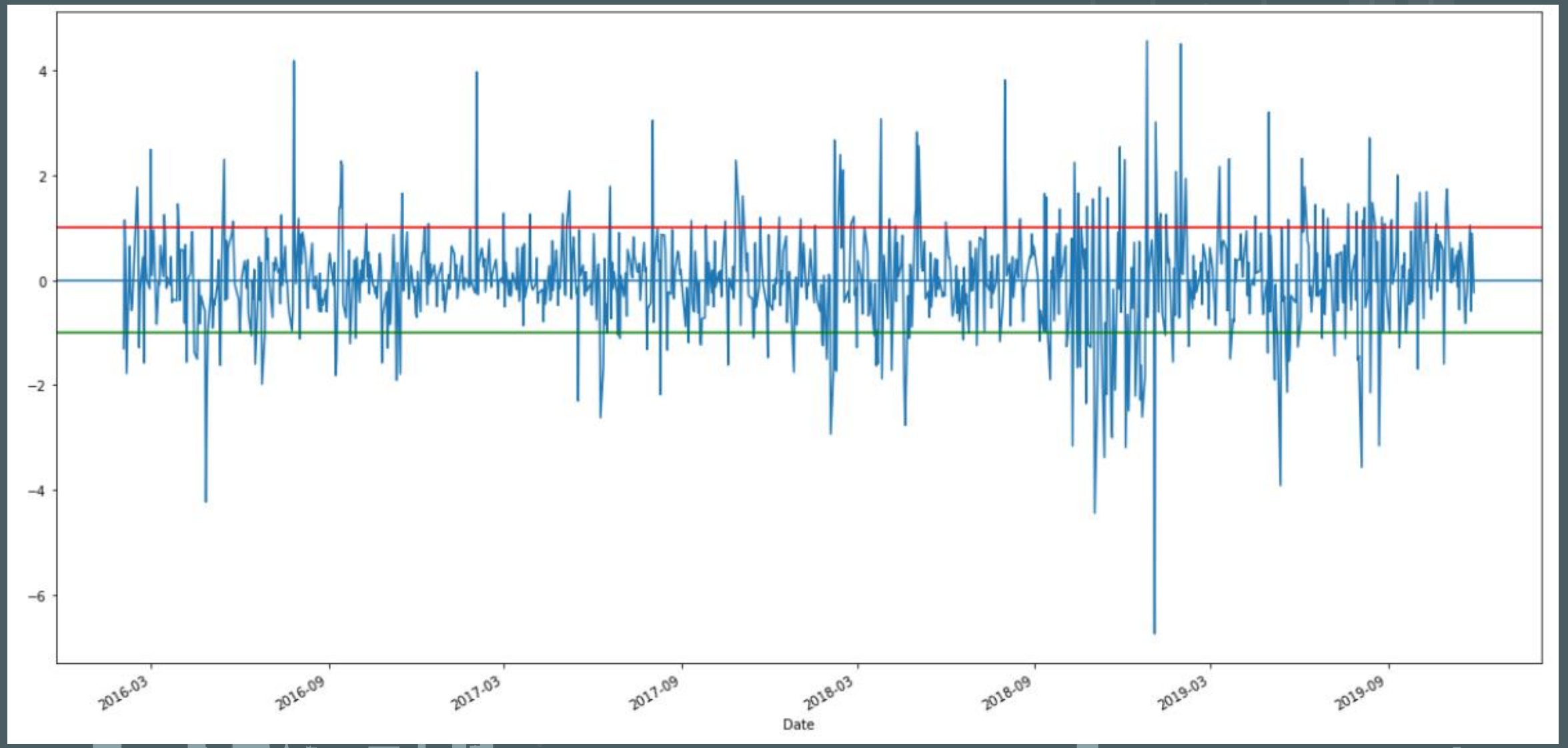
- They are of 2 types:
  - Seasonal model
  - Non-seasonal model

- We have used Non-seasonal ARIMA model which consists of 3 variables namely:
  - P = Periods to lag for
  - D = Number of differencing transformations
  - Q = Variable denoting lag of error component

Adjacent Close Price VS Predicted Price



Z score graph



When to buy, sell or hold ?

- The stock hits your buy value
- When it is undervalued
- When a stock goes on sale

HOLD

- The stock hits the target price
- You achieve a specific goal
- You need money for an emergency

$100.50

BUY

SELL

$203.00

Date	Z Score	Adj Close	Prediction
2017-09-21	0.555707	45.66	45.27
2017-09-22	-0.011547	45.62	45.63
2017-09-25	-0.589027	45.17	45.59
2017-09-26	-0.367161	44.90	45.16
2017-09-27	0.803026	45.48	44.92

- 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)

Date	Adj Close	Action
2017-09-21	45.66	Sell
2017-09-22	45.62	Hold
2017-09-25	45.17	Buy
2017-09-26	44.90	Buy
2017-09-27	45.48	Sell

Trading Simulation Model

Date	Action	Money	Current	Price
2017-09-21 00:00:00	Sell	1000	0.0	45.66
2017-09-22 00:00:00	Hold	1000	0.0	45.62
2017-09-25 00:00:00	Buy	1000	0.0	45.17
2017-09-26 00:00:00	Buy	6.37	987.81	44.9
2017-09-27 00:00:00	Sell	6.37	1000.5	45.48
2017-09-28 00:00:00	Hold	1006.87	0.0	45.55
2017-09-29 00:00:00	Hold	1006.87	0.0	45.57
2017-10-02 00:00:00	Sell	1006.87	0.0	46.28
2017-10-03 00:00:00	Sell	1006.87	0.0	46.62
2017-10-04 00:00:00	Buy	1006.87	0.0	46.37
2017-10-05 00:00:00	Sell	33.17	986.02	46.95
2017-10-06 00:00:00	Sell	1019.19	0.0	47.08
2017-10-09 00:00:00	Buy	1019.19	0.0	46.72
2017-10-10 00:00:00	Hold	38.14	983.24	46.82
2017-10-11 00:00:00	Buy	38.14	977.08	46.53
2017-10-12 00:00:00	Buy	38.14	965.36	45.97
2017-10-13 00:00:00	Buy	38.14	959.59	45.69
2017-10-16 00:00:00	Sell	38.14	972.31	46.3
2017-10-17 00:00:00	Sell	38.14	975.89	46.47
2017-10-18 00:00:00	Sell	38.14	996.35	47.45
2017-10-19 00:00:00	Buy	1034.48	0.0	47.1
2017-10-20 00:00:00	Sell	45.29	1006.88	47.95
2017-10-23 00:00:00	Hold	1052.17	0.0	47.88
2017-10-24 00:00:00	Sell	1052.17	0.0	48.32
2017-10-25 00:00:00	Buy	1052.17	0.0	47.8
2017-10-26 00:00:00	Sell	0.46	1056.7	48.03
2017-10-27 00:00:00	Sell	1057.16	0.0	48.4
2017-10-30 00:00:00	Buy	1057.16	0.0	47.97
2017-10-31 00:00:00	Buy	1.76	1045.78	47.54

- A trading simulation model which helps the user to make a decision based on prediction.
- The model tells us about mainly 4 things:
  - Action: whether to buy, sell or hold stock
  - Money: Initial amount $1000 which changes over the time
  - Current: Current amount of money after every action
  - Price: Price of stock

Conclusion

- Linear Regression and K-Nearest Neighbor failed to give accurate results
- ARIMA being time series forecasting model gave much more accurate results than regression algorithms
- Trading Simulation model gave an overview of when to buy, sell and hold share

References

- https://towardsdatascience.com/time-series-forecasting-arima-models-7f221e9eee06
- https://www.thebalance.com/know-when-to-buy-or-sell-any-stock-4023014
- https://www.kaggle.com/ravishankars/nyse-stock-data-arima-model
- https://www.analyticsvidhya.com/blog/2018/10/predicting-stock-price-machine-learningnd-deep-learning-techniques-python/
- https://datascienceplus.com/knn-classifier-to-predict-price-of-stock/
- https://medium.com/datadriveninvestor/machine-learning-for-stock-market-investing-f90ad3478b64
- https://medium.com/auquan/pairs-trading-data-science-7dbedafcfe5a


Thank You