FACULTY OF SCIENCE & TECHNOLOGY
Department of Computing & Informatics
Forensic Computing & Security

Tools & Technologies of Data Science **February 6th 2020**
Word Count: 2800

Paul-David Jarvis
s5115232@bournemouth.ac.uk

New York Stock Exchange & Microsoft Corporation Stock
Big Data Analytics & Technologies Tools

Table of Contents

Table of Figures

Abstract

I have decided that my big data problem is how I could use big data and big data sets such as historical financial stock data sourced from Kaggle that was sourced from the New York Stock Exchange (NYSE) and how I can use data analytics like Python and Machine Learning algorithms (MLA) like Support Vector Regression (SVR) and Linear Regression (LR) models to predict Microsoft Corporation's (MSFT) stocks. In this project, I will be researching into the NYSE and MSFT as well several different modules in Python like Pandas, Matplotlib, Scikit-Learn and others that I could use to analyse the information in the data set and predict future and unknown stock prices. This project and other similar projects are important within the financial section and especially within the world of trading as in the 21st century most jobs within the NYSE now are investment analysts looking, learning and designing algorithms that can be used to make stock trades.

Introduction

There are four paradigms of science: Empirical Evidence, Theories, Computational, Data Intensive. The first paradigm of science is empirical evidence is information gathered by the means senses. This information is gathered particularly by observing and documenting potential patterns and behaviour of a certain experiment. The second paradigm of science is theories. This is an explanation of specific requirements such as measurement and observation of an experiment that can be repeatedly tested and verified. A theory could be ultimately rejected if new scientific evidence is gathered and can't be explained. Computational is the third paradigm of science. It is often referred to as "scientific computing". This paradigm uses advanced computing power and its capability to solve complex problems found in nature. The information gathered is understood through the analysis and mathematical modelling, extremely different from empirical and theories. Data intensive is the fourth paradigm of science. The idea of this paradigm was first conceived during the 90s due to the internet and the volume and velocity of data that is being generated and needing processing, analysing and sharing.

1.0 Big Data Problem

1.1 New York Stock Exchange & Its Economic Relevance

The NYSE is open from Monday to Friday and is only closed on the weekends and certain NYSE's holidays. The NYSE produces roughly 2 terabytes every day so we can calculate that it produces roughly 500 terabytes or half a petabyte every year. Due to the volume and velocity of the data that the NYSE is producing there will be certain technical issues that will arise.

Microsoft was founded in 1975 and went public just over a decade later in 1986 at $21 per share and now in 2020 they are $100 plus, this can be seen in figure 1. Because MSFT is only one in thousands of other companies listed on the NYSE means that the volume of data for MSFT is not necessarily big data but has the potential to grow due to the velocity of the data.

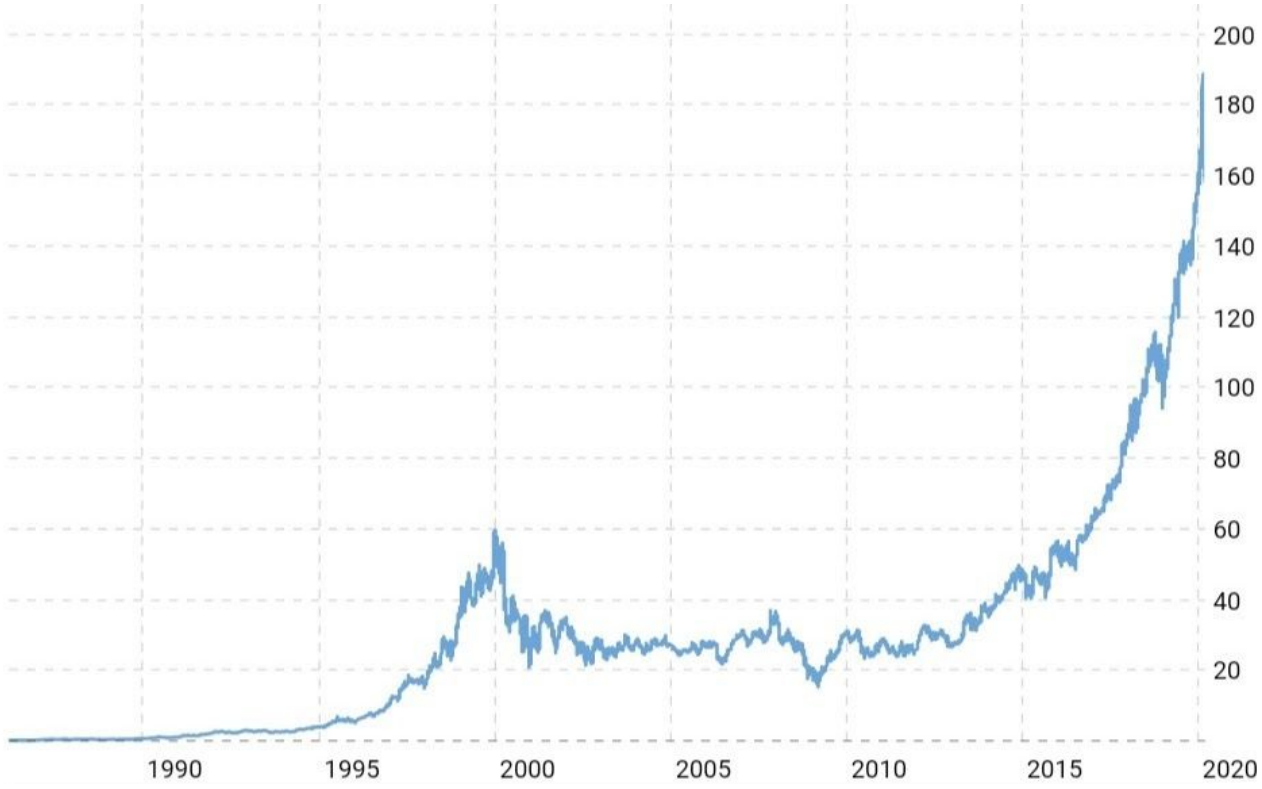


Figure 1: Microsoft Corporation Stock Price Chart. - (Macrotrends.net, 2020)

Microsoft Historical Annual Stock Price Data

Year	Average Stock Price	Year Open	Year High	Year Low	Year Close	Annual % Change
2020	170.5631	160.6200	188.7000	157.5800	161.5700	2.45%
2019	130.3820	101.1200	158.9600	97.4000	157.7000	55.26%
2018	101.0340	85.9500	115.6100	85.0100	101.5700	18.74%
2017	71.9840	62.5800	86.8500	62.3000	85.5400	37.66%
2016	55.2593	54.8000	63.6200	48.4300	62.1400	12.00%
2015	46.7136	46.7600	56.5500	40.2900	55.4800	19.44%
2014	42.4533	37.1600	49.6100	34.9800	46.4500	24.16%
2013	32.4915	27.6200	38.9400	26.4600	37.4100	40.06%
2012	29.8203	26.7650	32.8500	26.3700	26.7097	2.89%
2011	26.0522	27.9800	28.8700	23.7050	25.9600	-6.99%
2010	27.0584	30.9500	31.3900	23.0100	27.9100	-8.43%
2009	22.9766	20.3300	31.3900	15.1500	30.4800	56.79%
2008	26.6475	35.2200	35.3700	17.5300	19.4400	-45.39%
2007	30.4459	29.8600	37.0600	26.7200	35.6000	19.22%
2006	26.2908	26.8400	30.1900	21.5100	29.8600	14.19%
2005	25.8710	26.7400	28.1600	23.9200	26.1500	-2.13%
2004	27.1247	27.4500	29.9800	24.1500	26.7200	-2.37%
2003	26.1014	26.8600	29.9600	22.8000	27.3700	5.88%
2002	27.2745	33.5200	34.9300	21.4150	25.8500	-21.96%
2001	31.2712	21.6900	36.8400	21.6900	33.1250	52.72%
2000	38.1098	58.2800	58.2800	20.7500	21.6900	-62.84%
1999	43.7663	35.2500	59.5600	35.2500	58.3750	68.36%
1998	24.6307	16.3913	35.8900	15.8750	34.6725	114.61%
1997	15.1192	10.2025	18.6875	10.2025	16.1563	56.44%
1996	7.5441	5.6094	10.6875	5.0119	10.3275	88.31%
1995	5.1904	3.7619	6.8125	3.6875	5.4844	43.55%
1994	3.2051	2.5038	4.0388	2.4569	3.8206	51.65%
1993	2.6087	2.6600	3.0078	2.2188	2.5194	-5.56%
1992	2.5059	2.3750	2.9688	2.0859	2.6678	15.11%
1991	1.6104	1.0382	2.3177	1.0208	2.3177	121.77%
1990	0.8645	0.6163	1.1007	0.5981	1.0451	72.97%
1989	0.4355	0.3724	0.6146	0.3229	0.6042	63.39%
1988	0.3854	0.3889	0.4835	0.3194	0.3698	-1.83%
1987	0.3413	0.1658	0.5486	0.1658	0.3767	124.90%

Figure 2: Microsoft Corporation Annual Stock Data. - (Macrotrends.net, 2020)

1.2 5Vs

1.2.1 Volume

The NYSE produces roughly 2 terabytes a day at the end of the closing bell and 500 terabytes roughly a year, this volume of data will introduce technical challenges when it comes to storing the processing the data. MSFT's historical financial data set that I am using which was sourced from Kaggle isn't necessarily big data or a big data set but has the potential to grow every day.

1.2.2 Velocity

The NYSE produces data every weekday at the end of the day. The stock exchange is closed on weekends and certain holidays so it's opened roughly 260 days a year. The velocity of my data set that I am using is that it grows depending on the type of data set of MSFT stock. For example, If you use daily data then the velocity is that it's growing daily or if you use annual historical data then it grows every year, you can also import real time data from the NYSE with tickers that is every millisecond.

1.2.3 Variety

The historical data set that I am using off Kaggle that was sourced from the NYSE's data variety is unstructured and is a TXT file. I can easily convert them into semi structured CSV files that made it much easier to use Pandas library in Python and it's method read_csv. This was also efficient as the primary dataset being the full over 200 records CSV file was less efficient than using the second dataset I created which is the historical MSFT's stock data but cut down to roughly 20 records. The data set I am using has 7 columns: Date, Open, High, Low, Close, Adj Close and Volume.

1.2.4 Veracity

The data that the NYSE produces is extremely trustworthy. Plenty of Financial companies share their data with the NYSE and the SEC The data that I used was sourced from Kaggle which was sourced from the NYSE and the NASDAQ stock market and therefore making it trustworthy. However, it should be noted that the stock market could be possibly manipulated by artificially inflating or deflating the price of the stocks for personal gain, there are a few ways to manipulate the price of the stock but one way is for a company to place hundreds of small orders at a low price in an attempt to give the impression to anyone looking at this specific stock that there is something wrong.

1.2.5 Value

The data is extremely valuable as the data that is produced by NYSE is used to provide valuable information to traders so they can be informed with all the facts when it comes to making their trades. The data set that I am using to predict MSFT's stock differs in its value, it's historic data collected over several years so at first it might be discarded as unuseful due to it being outdated, however, I used this historical data to analyse it and I was able to plot the data on charts and discover future trends that were extremely on point.

1.3 Project Objectives

The main objective of this project is to analyse MSFT's stocks to identify trends that can be used to predict future stock. I first downloaded the historical MSFT stock from a trustworthy source like Kaggle and then converted the downloaded unstructured MSFT TXT file into a semi structured CSV file. We import the downloaded historical MSFT's stock CSV file into Python and then use the necessary libraries such as Pandas, Numpy and Sklearn to create MLA and SVR and LR to analyse the stock and Matplotlib to visualise and plot the data produced graphs and charts and finally document our results and discuss the results regarding if I achieved these objectives and to discuss any future work with the prime objective to make this project more efficient using other big data technology.

2.0 Big Data Tools & Technologies Specification

I have identified several tools and technologies I could use in order to solve my big data problem. For example, I could use Hadoop that can be used by loading the big data and data sets onto servers or machines that are running Hadoop software and the large data sets are then split into pieces and spread across the Hadoop software running servers. I then send the code to each server and the results are returned whole, this being MapReduce. This addresses efficient storage and efficient processing of the big datasets we use when they grow in volume, velocity and variety.

I could also use Python 3 and some of Python modules like Scikit-Learn and Matplotlib to develop scripts that include MLA which will allow me to efficiently address the processing of the data as Scikit-Learn will allow me to generate SVR and LR models that will allow me to read and import the data into the algorithms and process it effectively to get the results needed, I can also use Python's modules like Matplotlib as well to visualise my data that the SVR and LR models generated and the results to identify trends that can be used to predict future and unknown price that would solve my proposed big data problem.

I could also use AWS and their wide range of big data and analytic services such as Amazon Redshift which is an analytic tool and data warehousing. Amazon Redshift is a fully managed petabyte-scale cloud-based data warehouse designed to store and analyse big data and big data sets. Amazon Redshift delivers fast performance for querying data due to its column-oriented database. Each Amazon Redshift data warehouse contains a collection of nodes which are computing resources put together to form a cluster. Each one of these clusters runs its own Amazon Redshift engine and contains databases filled with clients data sets. It solves the technical issue that occurs regarding the volume and velocity of data due to its fast processing and scalability with the ability to buy more clusters of data.

3.0 Big Data Tools and Technologies Implementation

I used Python 3 and Google Colab to develop the scripts needed in order to solve my big data problem. I used the Pandas method read_csv to read the MSFT historical stock data and create a data frame that I can use to call the data. I used Scikit-Learn to import MLA SVR and LR Model, using Sklearn, I created 3 SVR Models: Radial Basis Function (RBF), Polynomial (Poly) and Linear (Lin) and trained them by fitting it with dates and prices, in addition to this, I also used Sklearn to create and train an LR model. I then used Matplotlib to plot the results from the data returned from the 3 SVR models and 1 LR model as well as MSFT stock data to see which model fits the MSFT stock data best.

```
# This program attempts to predict Microsoft Corp stock for a specific day
# by importing histroical MSFT stock data and using Machine Learning algorithm
# called Support Vector Regression (SVR) & Linear Regression.

# Using inspiration from several sources such as Youtube
# Code was used from https://www.youtube.com/watch?v=AF8zgxLukg4
```

```
#Importing Packages & Dependancies
import pandas as pd
import numpy as np
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
```

Figure 3: Code Dependencies & Libraries.

Figure 4 is a screenshot of the python code that was used to generate the 3 SVR models: Linear, Polynomial and Radial Basis Function and the LR model. This also shows the code that is used to train the SVR and LR models by feeding them the dates and prices arrays. You can also see within the screenshot, the code that is used to create the chart and graphs that includes all the information and data that the SVR and LR models returns as well as the scatter data used to plot the MSFT's opening stock price using Matplotlib.

```
[82] def predict_prices(dates, prices, x):

       #Creates 3 SVR models
       svr_lin = SVR(kernel='linear', C=1e3)
       svr_poly = SVR(kernel='poly', C=1e3)
       svr_rbf = SVR(kernel='rbf', C=1e3)

       #Trains the SVR models
       svr_lin.fit(dates, prices)
       svr_poly.fit(dates, prices)
       svr_rbf.fit(dates, prices)

       #Creates & Trains the Linear Regression Model
       lin_reg = LinearRegression()
       lin_reg.fit(dates, prices)

       #Plots all models and data onto a graph
       plt.scatter(dates, prices, color='black', label='Data')
       plt.plot(dates, svr_rbf.predict(dates), color='red', label='SVR RBF')
       plt.plot(dates, svr_poly.predict(dates), color='blue', label='SVR POLY')
       plt.plot(dates, svr_lin.predict(dates), color='green', label='SVR LINEAR')
       plt.plot(dates, lin_reg.predict(dates), color='orange', label='LINEAR REG')
       plt.xlabel('Days')
       plt.ylabel('Price')
       plt.title('Regression')
       plt.legend()
       plt.show()

       return svr_rbf.predict(x)[0], svr_lin.predict(x)[0], svr_poly.predict(x)[0], lin_reg.predict(x)[0]
```

Figure 4: Support Vector Regression & Linear Regression Models.

4.0 Results and Discussions

The original MSFT stock dataset that I used which was sourced from Kaggle had roughly 200 records of historical stock prices and in developing the project further I discovered that I needed to reduce and cut it down to roughly 20 records, this doesn't change anything or reduce the likeliness of success for this project as the 20 records were enough data to predict an unknown future price.

Figure 5 shows a screenshot of the code that prints the last record and the record itself for the 24th of February 2020 that is stored in the MSFT historical stock price CSV file. I will be using this last record so I can test and confirm instantly whether or not the SVR and LR models have successfully predicted a float number close enough to the open price shown in figure 5.

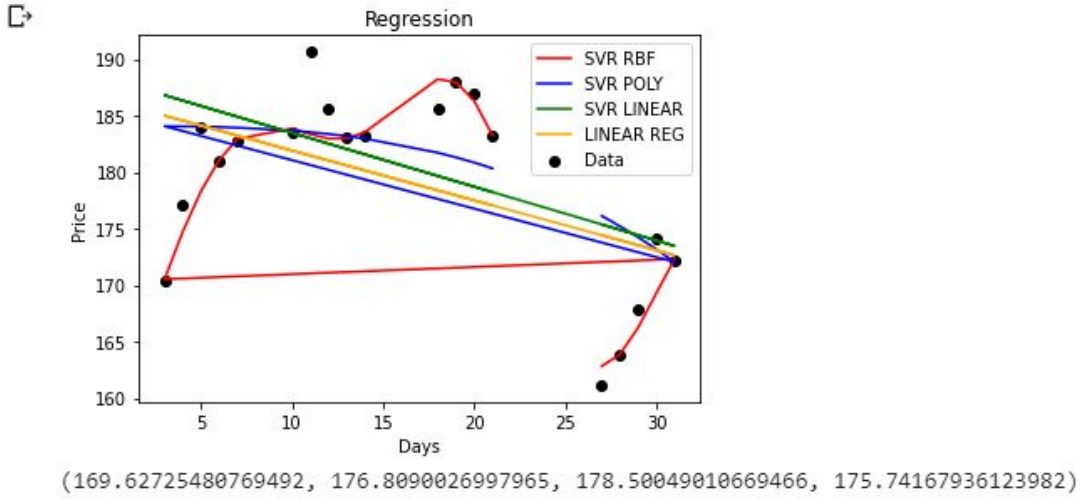
```
[76] #Prints the last row and column within the dataset
     df.tail(1)
```

	Date	Open	High	Low	Close	Adj Close	Volume
19	2020-02-24	167.770004	174.550003	163.229996	170.889999	170.889999	68311100

Figure 5: February 24th Stock Record.

Figure 6 is a Regression chart that was produced by the three SVR and LR models that I created and trained. The array number ([[“24”]]) is for the date which is the last record in the reduced data set shown in figure 5, this is one of the reasons why I predicted the opening stock price for the 24th of February because it allowed me to easily check whether or not the all four models predicted a float number anywhere near the actual opening price. The best SVR model that aligns with the scatter data that shows the MSFT stock opening prices is the RBF and the first float number (“169.62725480769492”) is the predicted price for the 24th of February 2020 which was $169 and was extremely close to the opening price that you can see in figure 5 and was $167 or 167.770004 for that date. My SVR RBF model prediction was only $2 or 1.09% (1.09490117599%) off.

```
[83] predicted_price = predict_prices(dates, prices, [[24]])
     print(predicted_price)
```




```
(169.62725480769492, 176.8090026997965, 178.50049010669466, 175.74167936123982)
```

Figure 6: February 24th Stock Open Price Prediction.

In conclusion, regarding my objectives, I successfully completed my primary objective by using Python’s Scikit-Learn module and SVR and LR models to predict MSFT’s 24th of February 2020 opening stock price with an error of $2. In addition to this, I was also able to complete my other objectives by using Matplotlib to virtualise all my information and data on graphs like the Regression chart that includes the SVR, LR models and MSFT’s opening prices like the one in figure 6. I was also able to produce another chart like figure 7 which is a chart of MSFT’s stock moving average which can be used to identify the direction of the trend and also determine risks, this is done by smoothing out the price and filter out anomaly fluctuations caused by random short term price. I also and looked into producing a candlestick graph for Microsoft Corp’s stock but Matplotlib.finance was removed and was the module I needed that would allow me to produce the candlestick chart as seen in figure 8 that was sourced from online and has the same data plotted on it.

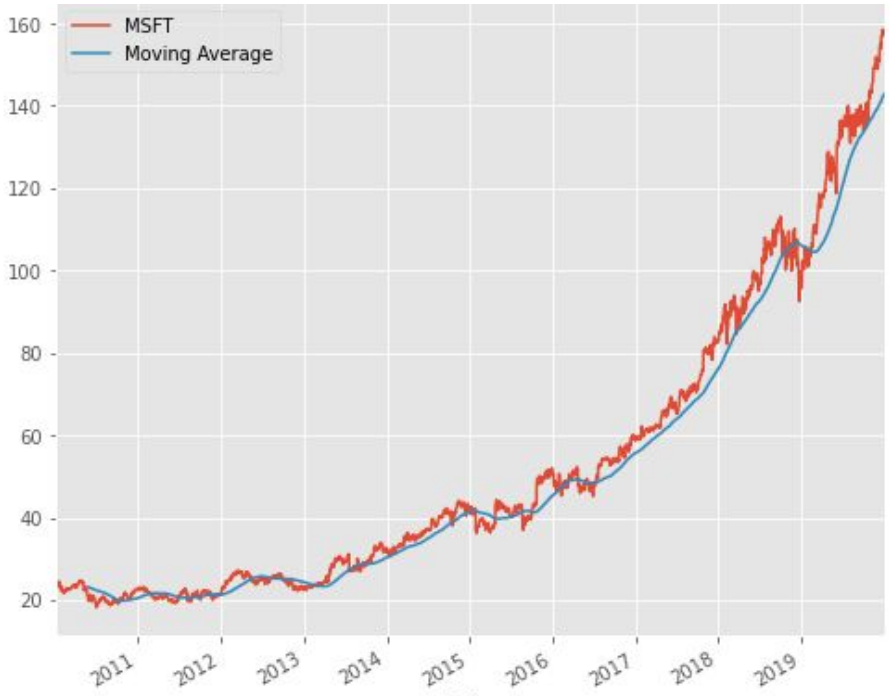


Figure 7: Microsoft Corporation's Stock Moving Average.

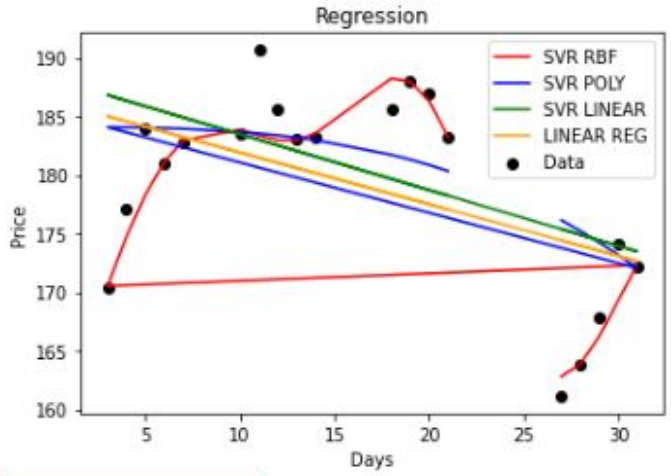
Figure 8: Microsoft Corporation Candle Stick Graph - (MSFT - candlestick chart analysis of Microsoft Corporation, 2001)

I wanted to reinforce the success of my primary objectives by performing several more tests and attempting to predict more than a single opening price for MSFT. When looking over MSFT's historical stock prices, I saw that on the 11th of February there was a sudden rise in the opening stock price by roughly $5 to $10, this can be seen in figure 9. I wanted to know which SVR model or the LR model would predict closest to the anomaly opening stock. The SVR RBF model was $7 off and can be seen in figure 10. Taking into consideration that the models are trained by using this data, it makes sense that RBF returned a prediction that aligns more with the previous prices.

8	2020-02-06	180.970001
9	2020-02-07	182.850006
10	2020-02-10	183.580002
11	2020-02-11	190.649994
12	2020-02-12	185.580002
13	2020-02-13	183.080002
14	2020-02-14	183.250000
15	2020-02-18	185.610001
16	2020-02-19	188.059998
17	2020-02-20	186.949997
18	2020-02-21	183.169998

Figure 9: Microsoft Corporation's Stock Opening Price from 06/02 to 21/02

```
predicted_price = predict_prices(dates, prices, [[11]])
print(predicted_price)
```



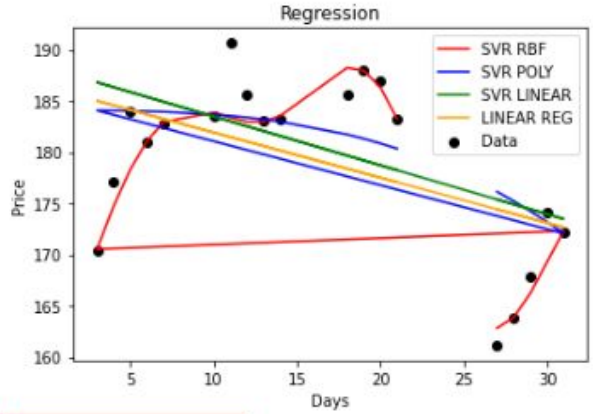

```
(183.3633031104008, 183.00350205018225, 183.54347672712686, 181.47644342052203
```

Figure 10: Microsoft Corporation's Stock Opening Price for 11/02 Prediction

In addition to predicting the opening stock price for the 11th of January or February. I also threw the prediction models on the 20th and 21st of February The SVR RBF model predicted both of these dates opening stock price to near almost exact regarding the price rounded up to no decimal places. These results can be seen in figure 11 and figure 12.

17	2020-02-20	186.949997	187.250000	181.100006	184.419998	184.419998	368624
18	2020-02-21	183.169998	183.500000	177.250000	178.589996	178.589996	485726

```
predicted_price = predict_prices(dates, prices, [[20]])
print(predicted_price)
```

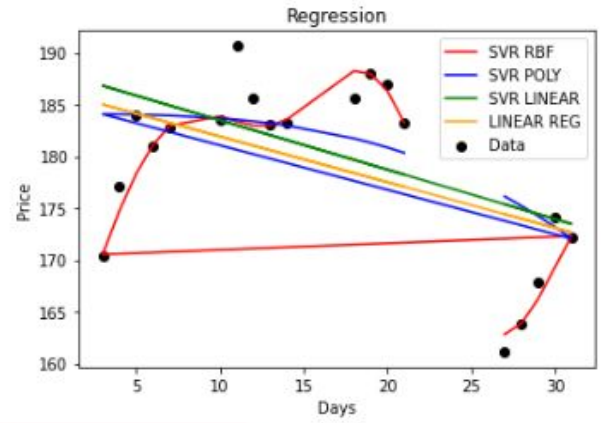



```
(186.31890369124088, 178.71500249998684, 180.85143496194507, 177.50622214871126)
```

Figure 11: Microsoft Corporation's Stock Opening Price for 20/02 Prediction

18	2020-02-21	183.169998	183.500000	177.250000	178.589996	178.589996	485726

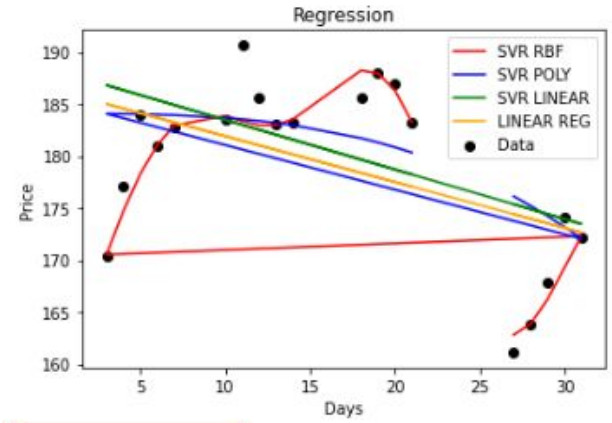
```
predicted_price = predict_prices(dates, prices, [[21]])
print(predicted_price)
```




```
(183.27011935330083, 178.2385025494736, 180.34241341947347, 177.0650864518434)
```

Figure 12: Microsoft Corporation's Stock Opening Price for 21/02 Prediction

I also wanted to see what would happen when I tried to predict using the MLA models MSFT's opening price with the same day. I ran the MLA models on the 27th. I wanted to try and predict the opening stock price for the 27th of February which is a date that wasn't in my revised MSFT's historical stock price data set. The SVR RBF model returned $162 that can be seen in figure 13. The opening stock price for the 27th of February according to Yahoo finance was $163 and this is shown in figure 14. I also noticed the opening stock price for the 27th of January was $161 that can be seen in figure 15. I am not certain what month the model predicted.



(162.82119187250217, 175.37950284965376, 176.13541697736366, 174.41827227063624)

Figure 13: Microsoft Corporation's Stock predicting for the 27th

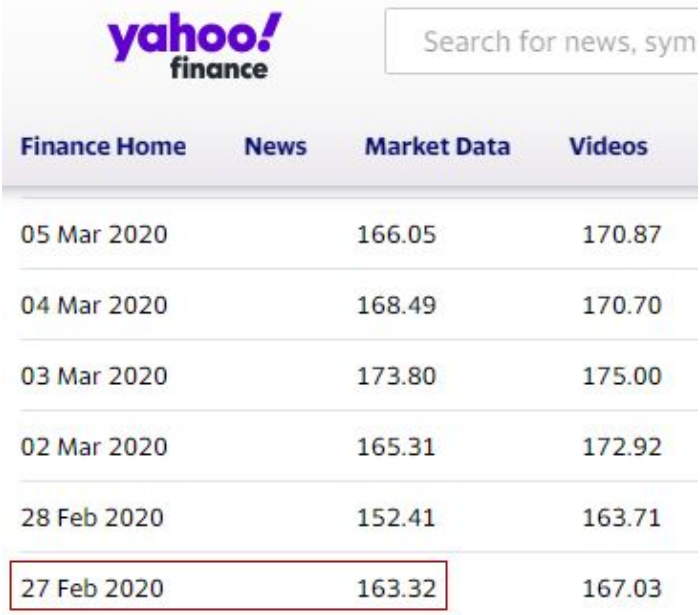


Figure 14: Microsoft Corporation's Stock Opening Price for 27/02 on Yahoo

Figure 15: Microsoft Corporation's Stock Opening Price for 27/01

5.0 Conclusions and Future Work

I enjoyed this project overall. It was rocky at the start but I found tons of great resources that helped a lot from youtube and official python documents and official python module documents. I enjoyed reading and learning about how the NYSE actually works and was interested in looking into an essential company like Microsoft. I learnt a lot about the libraries that I used like Matplotlib but more importantly, Scikit-Learn and the MLA SVR and LR. I was also happy to achieve the objectives I set out when first researching and beginning this project, those being able to predict future and unknown stock as well as plotting the data on charts produced by Matplotlib.

One of my future objectives of this project and something that I think would be incredible to achieve would be changing the way the ML SVR and LR models are coded and trained in order to predict more specific dates. One of my tests was to throw the MLA a date that is used more than once like the specific day such as the 27th. I think it would be amazing to be able to use the specific day, month and year to predict the opening stock price as this would be more efficient and would solve my problem shown in figure 13, figure 14 and figure 15 as I am unsure whether the MLA model predicted the 27th of January or February.

In regards to future work, It was interesting to see how we could possibly use the NYSE and big data sets in order to make this entire process more efficient with big data tools like Hadoop, AWS or Azure. When researching how to make this process more efficient using industry tools like Hadoop and AWS, we could potentially do some incredible things like predict future stock every day for a specific company or even maybe we could potentially feed all stock data from the NYSE or NASDAQ stock market and predict the stock for multiple companies every day.

6.0 References

Macrotrends.net. (2020). Microsoft - 34 Year Stock Price History | MSFT. [online] Available at: https://www.macrotrends.net/stocks/charts/MSFT/microsoft/stock-price-history [Accessed 8 Mar. 2020].

Hotcandlestick.com. 2001. *MSFT - Candlestick Chart Analysis Of Microsoft Corporation*. [online] Available at: <https://www.hotcandlestick.com/MSFT> [Accessed 22 March 2020].

Apache Hadoop & Big Data 101: The Basics. 2015. [video] https://www.youtube.com/watch?v=AZovvBgRLIY&t=1s: Cloudera, Inc.

Apache Hadoop Tutorial | Hadoop Tutorial For Beginners | Big Data Hadoop | Hadoop Training | Edureka. 2017. [video] https://www.youtube.com/watch?v=mafw2-CVYnA&t=4s: edureka!.

Big Data Tutorial For Beginners | What Is Big Data | Big Data Tutorial | Hadoop Training | Edureka. 2017. [DVD] https://www.youtube.com/watch?v=zez2Tv-bcXY&t=234s: edureka!.

Create Your Own Google Stock Prediction Program Using Python And Machine Learning. 2019. [DVD] https://www.youtube.com/watch?v=AF8zgxLukg4&t=693s.: Computer Science.

Matplotlib Tutorial 1 - Introduction and Line. 2015. [video] https://www.youtube.com/watch?v=q7Bo_J8x_dw&t=30s: sentdex.

Support Vector Machines, Clearly Explained!!!. 2019. [video] https://www.youtube.com/watch?v=efR1C6CvhmE: StatQuest with Josh Starmer.

What Is Big Data | AWS Big Data Tutorial For Beginners | Big Data Tutorial | Hadoop Training. 2018. [video] https://www.youtube.com/watch?v=ijF2vI5efVs&t=63s: CLOUD GURU.