Python Programming

Project Report

Student name: 艾琳 Aanchal Upreti

Student id: 191764145

Major: Software Engineering and Management

Table of Contents

Web Scraping Yahoo! Finance

Introduction

Data is one of the important features of every organization because it helps business leaders to make decisions based on facts, statistical numbers and trends. The interdisciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data is called data science. One of the best sources of data is the data available publicly online on various websites. Those websites have a whole array of data in the form of Text (Alphabetical, Numerical), Tabular, Images, Videos, GIFs, Flash etc. arranged in Structured and Unstructured format, and is called web data. Web data can be used to monitor competitors, track potential customers, keep track of channel partners, generate leads, build apps, and much more. Its uses are still being discovered as the technology for turning unstructured data into structured data improves.

There are mainly two ways to extract data from a website:

- Use the API of the website (if it exists). For example, Facebook has the Facebook Graph API which allows retrieval of data posted on Facebook.
- Access the HTML of the webpage and extract useful information/data from it. This technique is called web scraping or web harvesting or web data extraction.

Extracting large amounts of data manually is a very time consuming and inefficient approach. Also, if we can get required info through an API, it is almost always preferred approach over web scraping because API simply provides access to structured data from the provider, while in web scraping, we need to create an engine to extract the same information. However, not all websites provide an API as they do not want the readers to extract huge information in a structured way, while others don't provide APIs due to lack of technical knowledge.

Hence to fetch such important data from websites in a quick, robust and automated fashion, we need web scraping.

Aim and Objectives

Among several usages of web scraping, this project deals with real-world application of scraping the stock market to track, analyze and visualize the real-time price of the stock market. This project works on Yahoo! Finance website and hence fetches historical data of Apple Corporation (AAPL) and writes that data to the local .csv file. The fetched data is then used to compare and analyze the effect of coronavirus outbreak on the stock price of Apple and to visualize simple moving average and exponential moving average for 20 days and 50 days in a graph.

Methods

Since Yahoo! Finance discontinued its API services from March, 2017, Web Scraping using Python and its popular libraries and packages is used as the method to extract historical data for further analysis. This project uses BeautifulSoup4 to extract data from the web, manipulate and clean data using Python's Pandas library, and data analysis and visualize using Python's NumPy and Matplotlib library.

What is Web Scraping?

Web Scraping is a technique employed to download, parse, and organize large amounts of data from the web in an automated manner whereby the data is extracted and saved to a local file in your computer or to a database in table (spreadsheet) format. In simple terms, web scraping saves you the trouble of manually downloading or copying any data and automates the whole process.

How is Web Scraping done?

When we run the code for web scraping, a request is sent to the URL that we have mentioned. As a response to the request, the server sends the data and allows us to read the HTML or XML page. The code then, parses the HTML or XML page, finds the data and extracts it.

To extract data using web scraping with python, you need to follow these basic steps:

- Find the URL that you want to scrape
- Inspecting the Page
- Find the data you want to extract
- Write the code
- Run the code and extract the data
- Store the data in the required format

Main Tools Used

- **Python (3.8)**
  Python is an open-source programming language with a huge collection of libraries such as NumPy, matplotlib, Pandas etc., which provides methods and services for manipulation of extracted data for web scraping. It has easily understandable syntax and is supported by a large and active community.

- **BeautifulSoup**
  BeautifulSoup is an amazing parsing library in Python that enables the web scraping from HTML and XML documents. BeautifulSoup automatically detects encodings and gracefully handles HTML documents even with special characters. We can navigate a parsed document and find what we need which makes it quick and painless to extract the data from the web pages.

- **Requests**
  Requests library handles the interaction with the web page (Using HTTP requests)

- **Pandas**
  Pandas is a library used for data manipulation and analysis. It is used to extract the data and store it in the desired format.

- **NumPy**
  A very popular library that makes array operations very simple and fast.

- **Matplotlib:**
  a very popular library to plot graphs.

Data Acquisition

Data Source

This project works on a popular site for financial data i.e. Yahoo! Finance (Url: https://finance.yahoo.com/). Yahoo! Finance is a media property belonging to Yahoo! which provides financial data, including stock quotes, press releases, financial reports, technical indicators and original content. It also offers some online tools for personal finance management.

Implementation

In our project, we have used Web Scraping methodology on Yahoo Finance website.

- We set our base URL as http://finance.yahoo.com/quote/. Yahoo Finance supports search parameters on URL therefore we will be appending the company stock name of which we would be extracting stock data and date range.
- We have used the **Beautiful Soup** package for web scraping purposes and to easily fetch table rows of stocks data.
- We gave div's class names to beautiful soup to find the table and gave us an array of historical data's table rows.
- After fetching stocks data, Pandas package was used to create a Dataframe object. Dataframe objects are used to easily manipulate table columns and rows.
- Dataframe objects also help in storing Comma-separated Values (CSV) which is one of our goals to store this stock data in a CSV file. Therefore, after successfully fetching data from the website, our project stores this fetch data on a CSV file using **.to csv()** method of Dataframe. The

filename is set as the company stock name with the extension of '.csv' and if the file already exists, then our project appends the fetched data at the end of the file.

- The collected data is then analyzed to calculate SMA and EMA using **NumPy** package and plotted into graph using **matplotlib** package.

Data Analysis

For this project, we have fetched stocks prices data of **AAPL** for comparison of their Moving Averages before and after the spread of **Covid-19**. To achieve this goal, we followed the following steps:

- Firstly, for our first set of stock prices and their calculation of Simple Moving Average (SMA) and Exponential Moving Average (EMA), we had to first set start and end date for search parameters in URL. The start date is set as 27th December 2019 as according to various sources, the pandemic of **Covid-19** with its significant effect on the stock market started with this date. For the end date, 100 days were subtracted from the start date to gather data with a respected date range.
- Using web scraping methodology, we got closing prices during the date range given.
- We have used multiple moving averages for different periods together. We have a short-term moving average of 20 days and a long-term moving average of 50 days. An indication of a price trend changes maybe when the short-term moving average is crossing the long-term moving average:
    - When the short term moving average crosses above the long-term moving average, this may indicate a buy signal. This is known as a golden cross.
    - Contrary, when the short term moving average crosses below the long-term moving average, it may be a good moment to sell. This is known as a dead cross.
- After fetching 100 days closing prices of before the corona pandemic and converting it to, Dataframe object, the **SMA** is calculated using the **Rolling** method of Pandas by giving the number of days, i.e. 20 and 50 in our cases, as an argument.
- For EMA, by again using Pandas but instead of Rolling method, .**ewm()** is used for calculating the exponential moving average. We used closing prices with spans of both 20 and 50 days to get two sets of EMA for comparison with SMA.
- Finally, the data set including **Close Price**, **SMA** and **EMA** are plotted using the matplotlib package.
- Similarly, **SMA** and **EMA** are calculated and plotted on a graph by following the above steps but with the start date being **27th Dec 2019**, with end date as 100 days after the start date.

Moving Average

Moving averages act as a technical indicator to show you how a security's price has moved, on average, over a certain period of time. Moving averages are often used to help highlight trends, spot trend reversals, and provide trade signals. There are several different types of moving averages, but they all create a single smooth line that can help show you which direction a price is moving.

Simple Moving Average Calculation

The SMA for any given number of time periods is simply the sum of closing prices for that number of time periods, divided by that same number.

Simple moving average = (*N*–period sum) /N

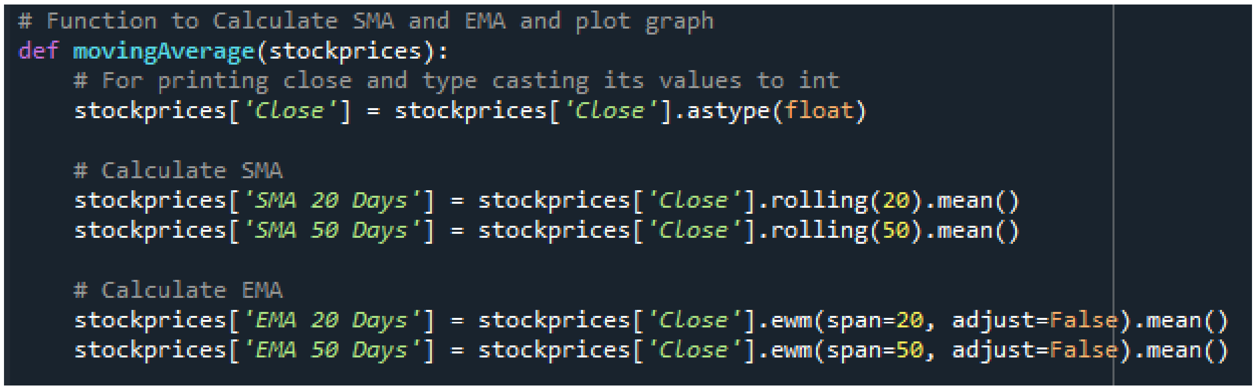
where: *N*=number of days in a given period

period sum = sum of stock closing prices in that period

Exponential Moving Average Calculation

The exponential moving average (EMA) is a weighted average of the last n prices, where the weighting decreases exponentially with each previous price/period.

Exponential moving average = (Close - previous EMA) * (2 / n+1) + previous EMA



```python
# Function to Calculate SMA and EMA and plot graph
def movingAverage(stockprices):
    # For printing close and type casting its values to int
    stockprices['Close'] = stockprices['Close'].astype(float)

    # Calculate SMA
    stockprices['SMA 20 Days'] = stockprices['Close'].rolling(20).mean()
    stockprices['SMA 50 Days'] = stockprices['Close'].rolling(50).mean()

    # Calculate EMA
    stockprices['EMA 20 Days'] = stockprices['Close'].ewm(span=20, adjust=False).mean()
    stockprices['EMA 50 Days'] = stockprices['Close'].ewm(span=50, adjust=False).mean()
```

```
# PLot Graph of Price CLose, SMA and EMA Data
stockprices[['Close', 'SMA 20 Days', 'EMA 50 Days', 'SMA 50 Days', 'EMA 20 Days']].plot(figsize=(12, 8))
plt.grid(True)
plt.title('AAPL Moving Averages')
plt.axis('tight')
plt.ylabel('Price')
```

Observations

I made a comparison among SMA and EMA for 20 days and 50 days and plotted the result in a graph. As a result, the following observation was made:

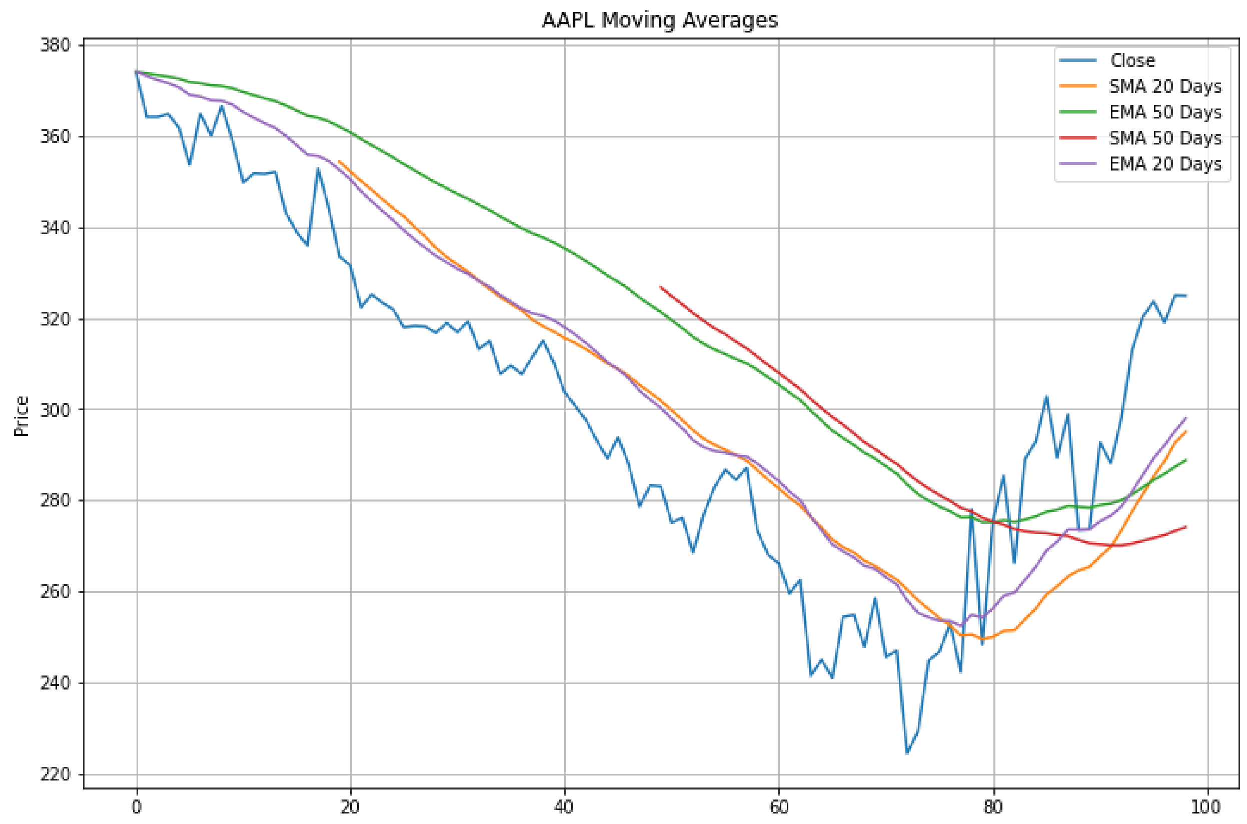
- SMA time-series are much less noisy than the original price time-series. However, this comes at a cost: SMA time-series lag the original price time-series, which means that changes in the trend are only seen with a delay (lag) of L days.
- Compared to simple moving averages, EMAs give greater weight to recent (more relevant) data.
- Exponential moving averages have less lag and are therefore more sensitive to recent prices - and recent price changes.

Effect of Coronavirus outbreak

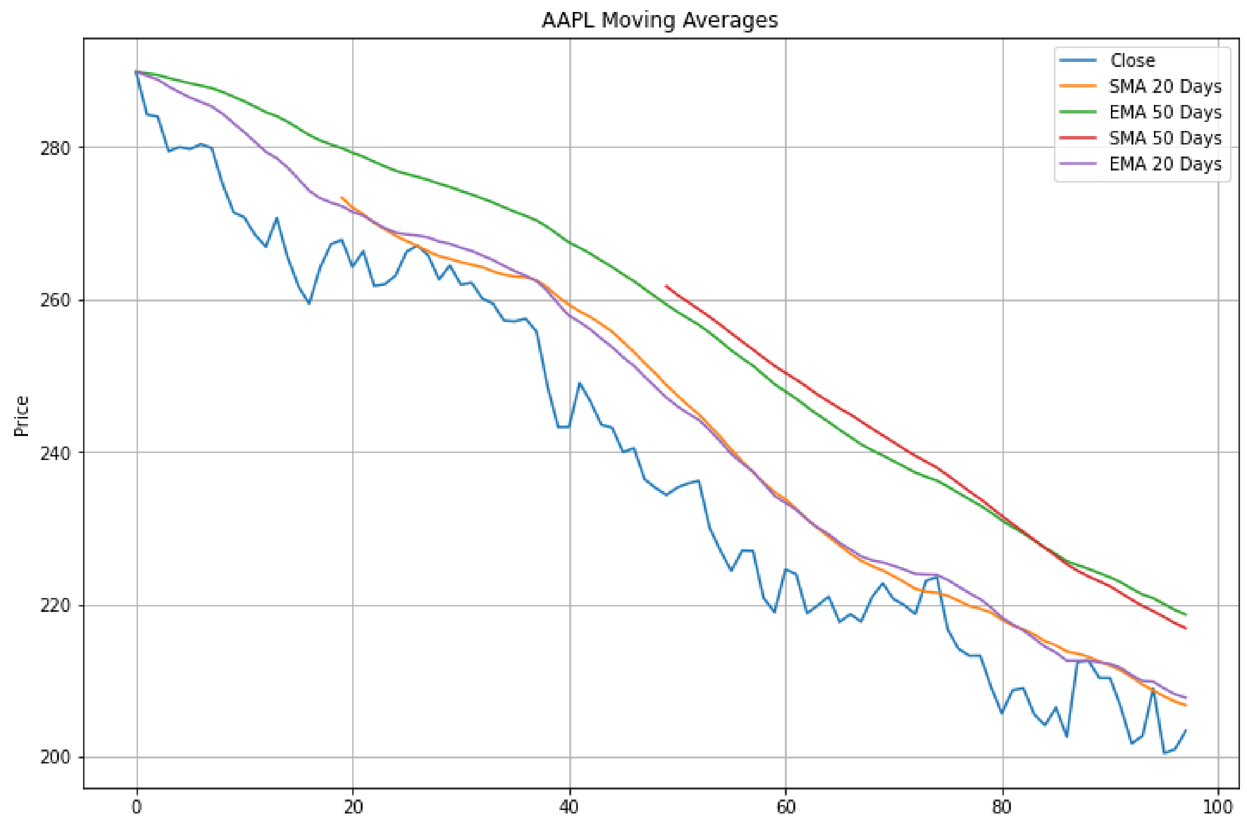
We all know that the virus did originate in Wuhan in December 2019. It rapidly grew to other countries and eventually has become a global pandemic. Because of this outbreak, there are also effects on the share market. Hence, we have made a comparison of the share market before and after the outbreak.

The after-effect observation includes calculation of SMA, EMA from 2019-12-27 till the present date. Whereas the before effect includes data of 100 days before the 27th of December 2019.

After Corona



Before Corona



Comparison of the stock price of AAPl

Open is the price of the stock at the beginning of the trading day (it need not be the closing price of the previous trading day), high is the highest price of the stock on that trading day, low the lowest price of the stock on that trading day, and close the price of the stock at closing time. Volume indicates how many stocks were traded. Adjusted close is the closing price of the stock that adjusts the price of the stock for corporate actions.

Before Corona

If we compare the closing price of AAPL stock between Dec 06, 2019, and Dec 26, 2019, we can see the increment of 6.62% in trading price.

Close price on Dec 06,2019 = 270.71

Close price on Dec 26,2019 = 289.91

Calculation:

((289.91 – 270.71)/289.91) * 100% = 6.62%

After Corona

If we compare the closing price of AAPL stock between April 23, 2020, and July 06, 2020, we can see the increment of 26.46% in trading price.

Close price on July 06,2020 = 374.01

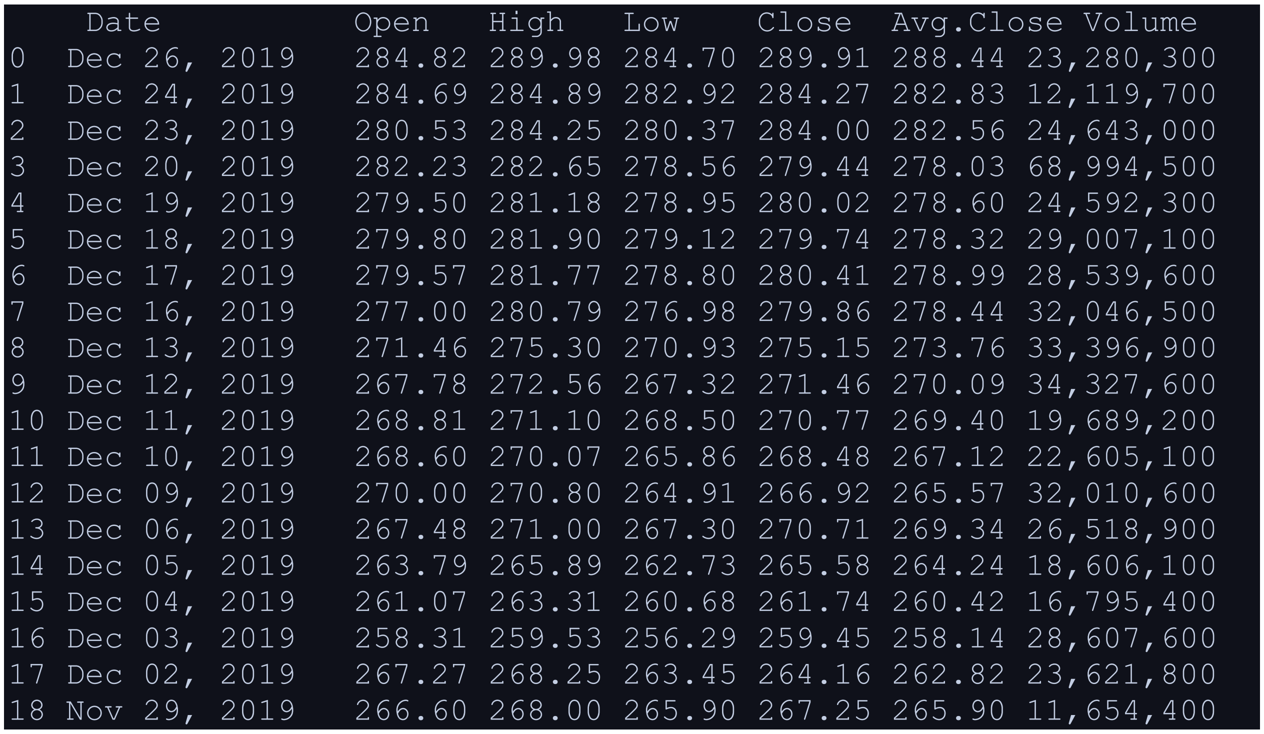
Close price on April 23,2020 = 275.03

Calculation:

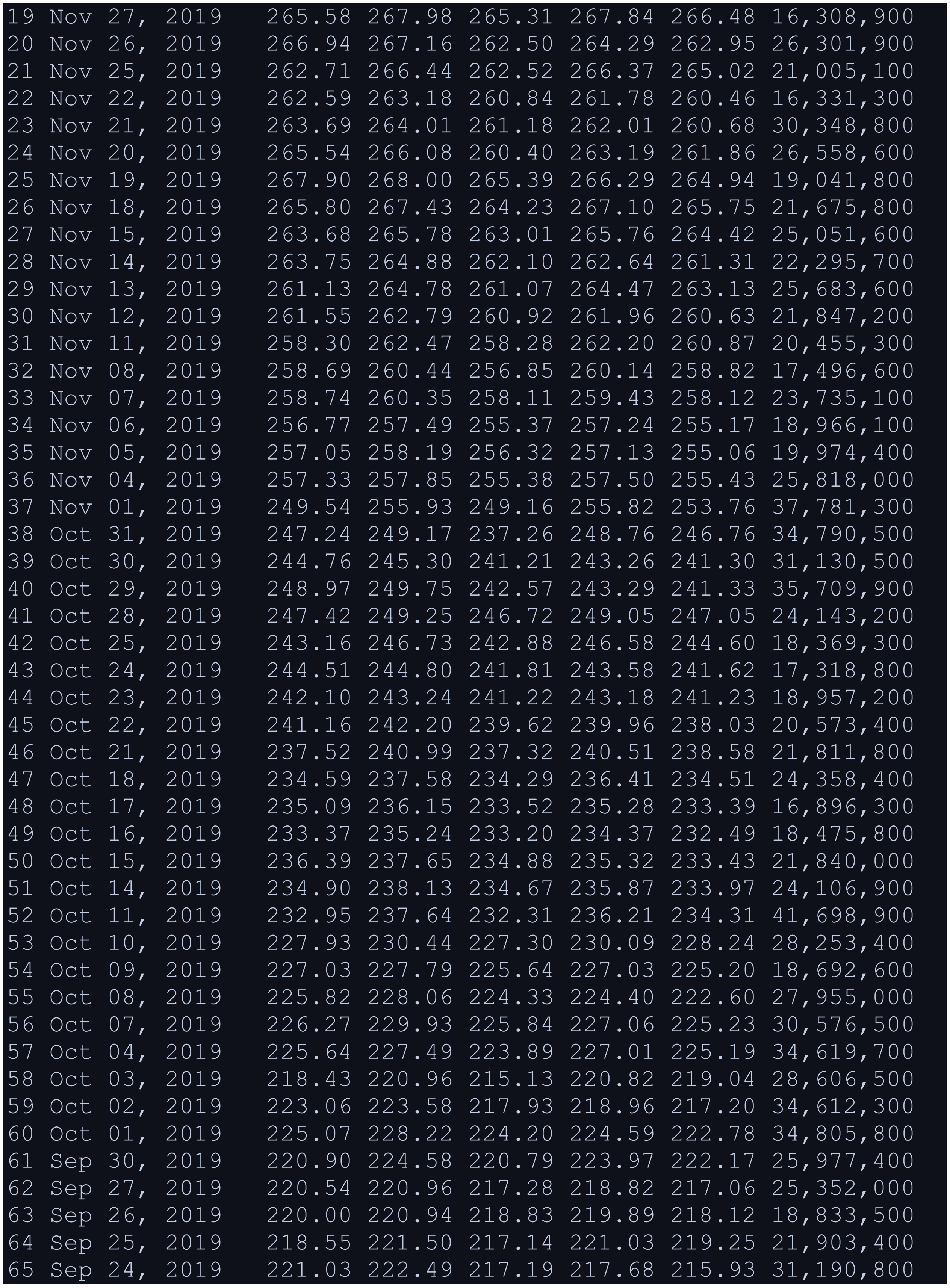
((374.01 – 275.03) / 374.01) * 100% = 26.46%

Hence, we can say, there is not much negative effect of the coronavirus outbreak on the AAPL stock market. Rather, the price seems to be increasing even after the corona outbreak.

Before Corona data



```
     Date              Open   High   Low    Close  Avg.Close Volume
0   Dec 26, 2019   284.82 289.98 284.70 289.91 288.44 23,280,300
1   Dec 24, 2019   284.69 284.89 282.92 284.27 282.83 12,119,700
2   Dec 23, 2019   280.53 284.25 280.37 284.00 282.56 24,643,000
3   Dec 20, 2019   282.23 282.65 278.56 279.44 278.03 68,994,500
4   Dec 19, 2019   279.50 281.18 278.95 280.02 278.60 24,592,300
5   Dec 18, 2019   279.80 281.90 279.12 279.74 278.32 29,007,100
6   Dec 17, 2019   279.57 281.77 278.80 280.41 278.99 28,539,600
7   Dec 16, 2019   277.00 280.79 276.98 279.86 278.44 32,046,500
8   Dec 13, 2019   271.46 275.30 270.93 275.15 273.76 33,396,900
9   Dec 12, 2019   267.78 272.56 267.32 271.46 270.09 34,327,600
10 Dec 11, 2019   268.81 271.10 268.50 270.77 269.40 19,689,200
11 Dec 10, 2019   268.60 270.07 265.86 268.48 267.12 22,605,100
12 Dec 09, 2019   270.00 270.80 264.91 266.92 265.57 32,010,600
13 Dec 06, 2019   267.48 271.00 267.30 270.71 269.34 26,518,900
14 Dec 05, 2019   263.79 265.89 262.73 265.58 264.24 18,606,100
15 Dec 04, 2019   261.07 263.31 260.68 261.74 260.42 16,795,400
16 Dec 03, 2019   258.31 259.53 256.29 259.45 258.14 28,607,600
17 Dec 02, 2019   267.27 268.25 263.45 264.16 262.82 23,621,800
18 Nov 29, 2019   266.60 268.00 265.90 267.25 265.90 11,654,400
```

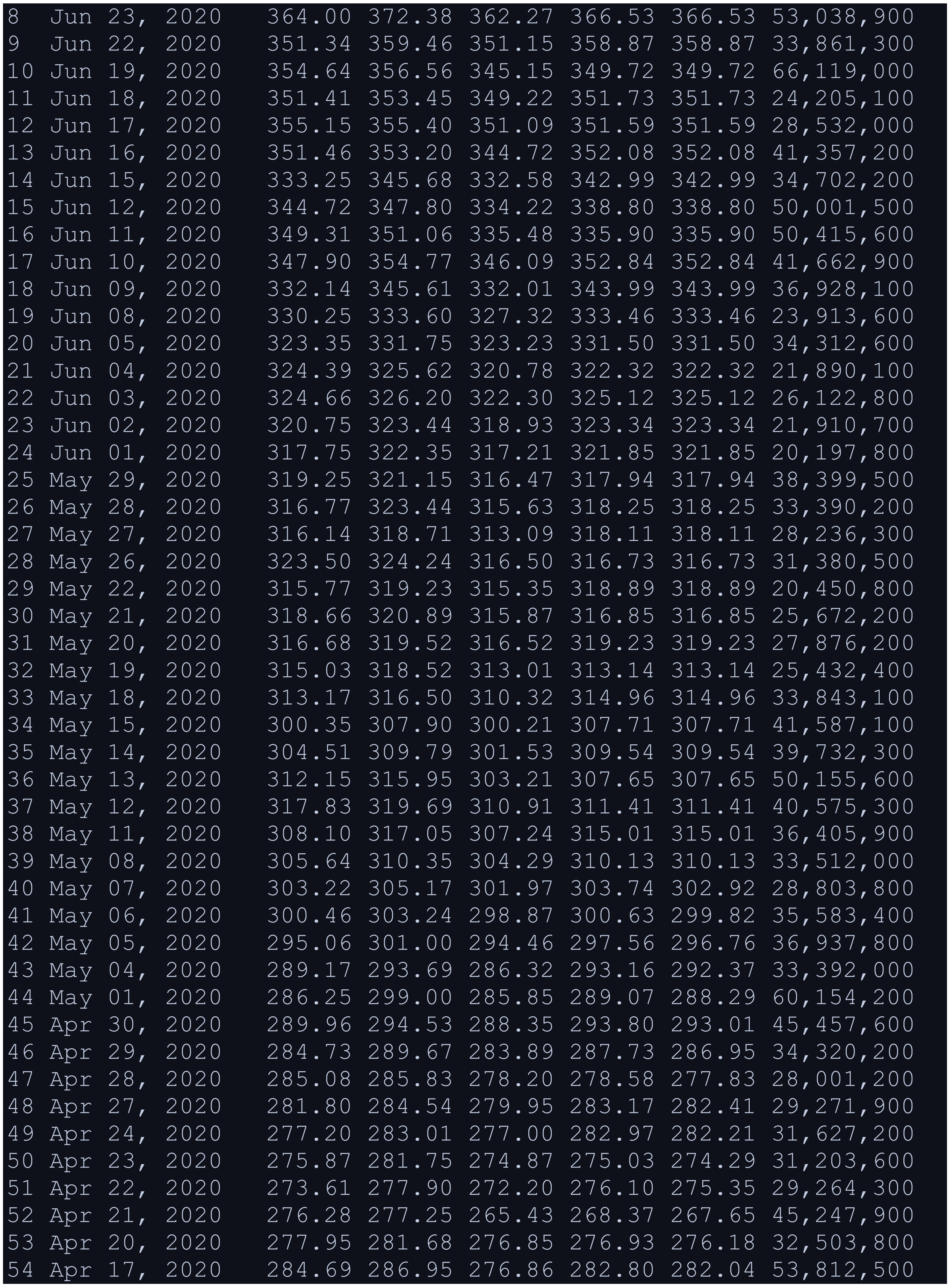


```
19 Nov 27, 2019   265.58 267.98 265.31 267.84 266.48 16,308,900
20 Nov 26, 2019   266.94 267.16 262.50 264.29 262.95 26,301,900
21 Nov 25, 2019   262.71 266.44 262.52 266.37 265.02 21,005,100
22 Nov 22, 2019   262.59 263.18 260.84 261.78 260.46 16,331,300
23 Nov 21, 2019   263.69 264.01 261.18 262.01 260.68 30,348,800
24 Nov 20, 2019   265.54 266.08 260.40 263.19 261.86 26,558,600
25 Nov 19, 2019   267.90 268.00 265.39 266.29 264.94 19,041,800
26 Nov 18, 2019   265.80 267.43 264.23 267.10 265.75 21,675,800
27 Nov 15, 2019   263.68 265.78 263.01 265.76 264.42 25,051,600
28 Nov 14, 2019   263.75 264.88 262.10 262.64 261.31 22,295,700
29 Nov 13, 2019   261.13 264.78 261.07 264.47 263.13 25,683,600
30 Nov 12, 2019   261.55 262.79 260.92 261.96 260.63 21,847,200
31 Nov 11, 2019   258.30 262.47 258.28 262.20 260.87 20,455,300
32 Nov 08, 2019   258.69 260.44 256.85 260.14 258.82 17,496,600
33 Nov 07, 2019   258.74 260.35 258.11 259.43 258.12 23,735,100
34 Nov 06, 2019   256.77 257.49 255.37 257.24 255.17 18,966,100
35 Nov 05, 2019   257.05 258.19 256.32 257.13 255.06 19,974,400
36 Nov 04, 2019   257.33 257.85 255.38 257.50 255.43 25,818,000
37 Nov 01, 2019   249.54 255.93 249.16 255.82 253.76 37,781,300
38 Oct 31, 2019   247.24 249.17 237.26 248.76 246.76 34,790,500
39 Oct 30, 2019   244.76 245.30 241.21 243.26 241.30 31,130,500
40 Oct 29, 2019   248.97 249.75 242.57 243.29 241.33 35,709,900
41 Oct 28, 2019   247.42 249.25 246.72 249.05 247.05 24,143,200
42 Oct 25, 2019   243.16 246.73 242.88 246.58 244.60 18,369,300
43 Oct 24, 2019   244.51 244.80 241.81 243.58 241.62 17,318,800
44 Oct 23, 2019   242.10 243.24 241.22 243.18 241.23 18,957,200
45 Oct 22, 2019   241.16 242.20 239.62 239.96 238.03 20,573,400
46 Oct 21, 2019   237.52 240.99 237.32 240.51 238.58 21,811,800
47 Oct 18, 2019   234.59 237.58 234.29 236.41 234.51 24,358,400
48 Oct 17, 2019   235.09 236.15 233.52 235.28 233.39 16,896,300
49 Oct 16, 2019   233.37 235.24 233.20 234.37 232.49 18,475,800
50 Oct 15, 2019   236.39 237.65 234.88 235.32 233.43 21,840,000
51 Oct 14, 2019   234.90 238.13 234.67 235.87 233.97 24,106,900
52 Oct 11, 2019   232.95 237.64 232.31 236.21 234.31 41,698,900
53 Oct 10, 2019   227.93 230.44 227.30 230.09 228.24 28,253,400
54 Oct 09, 2019   227.03 227.79 225.64 227.03 225.20 18,692,600
55 Oct 08, 2019   225.82 228.06 224.33 224.40 222.60 27,955,000
56 Oct 07, 2019   226.27 229.93 225.84 227.06 225.23 30,576,500
57 Oct 04, 2019   225.64 227.49 223.89 227.01 225.19 34,619,700
58 Oct 03, 2019   218.43 220.96 215.13 220.82 219.04 28,606,500
59 Oct 02, 2019   223.06 223.58 217.93 218.96 217.20 34,612,300
60 Oct 01, 2019   225.07 228.22 224.20 224.59 222.78 34,805,800
61 Sep 30, 2019   220.90 224.58 220.79 223.97 222.17 25,977,400
62 Sep 27, 2019   220.54 220.96 217.28 218.82 217.06 25,352,000
63 Sep 26, 2019   220.00 220.94 218.83 219.89 218.12 18,833,500
64 Sep 25, 2019   218.55 221.50 217.14 221.03 219.25 21,903,400
65 Sep 24, 2019   221.03 222.49 217.19 217.68 215.93 31,190,800
```

```
66 Sep 23, 2019   218.95 219.84 217.65 218.72 216.96 19,165,500
67 Sep 20, 2019   221.38 222.56 217.47 217.73 215.98 55,413,100
68 Sep 19, 2019   222.01 223.76 220.37 220.96 219.18 22,060,600
69 Sep 18, 2019   221.06 222.85 219.44 222.77 220.98 25,340,000
70 Sep 17, 2019   219.96 220.82 219.12 220.70 218.93 18,318,700
71 Sep 16, 2019   217.73 220.13 217.56 219.90 218.13 21,158,100
72 Sep 13, 2019   220.00 220.79 217.02 218.75 216.99 39,763,300
73 Sep 12, 2019   224.80 226.42 222.86 223.09 221.30 32,226,700
74 Sep 11, 2019   218.07 223.71 217.73 223.59 221.79 44,289,600
75 Sep 10, 2019   213.86 216.78 211.71 216.70 214.96 31,777,900
76 Sep 09, 2019   214.84 216.44 211.07 214.17 212.45 27,309,400
77 Sep 06, 2019   214.05 214.42 212.51 213.26 211.55 19,362,300
78 Sep 05, 2019   212.00 213.97 211.51 213.28 211.57 23,913,700
79 Sep 04, 2019   208.39 209.48 207.32 209.19 207.51 19,188,100
80 Sep 03, 2019   206.43 206.98 204.22 205.70 204.05 20,023,000
81 Aug 30, 2019   210.16 210.45 207.20 208.74 207.06 21,143,400
82 Aug 29, 2019   208.50 209.32 206.66 209.01 207.33 20,990,500
83 Aug 28, 2019   204.10 205.72 203.32 205.53 203.88 15,938,800
84 Aug 27, 2019   207.86 208.55 203.53 204.16 202.52 25,873,300
85 Aug 26, 2019   205.86 207.19 205.06 206.49 204.83 26,043,600
86 Aug 23, 2019   209.43 212.05 201.00 202.64 201.01 46,818,000
87 Aug 22, 2019   213.19 214.44 210.75 212.46 210.75 22,253,700
88 Aug 21, 2019   212.99 213.65 211.60 212.64 210.93 21,535,400
89 Aug 20, 2019   210.88 213.35 210.32 210.36 208.67 26,884,300
90 Aug 19, 2019   210.62 212.73 210.03 210.35 208.66 24,413,600
91 Aug 16, 2019   204.28 207.16 203.84 206.50 204.84 27,620,400
92 Aug 15, 2019   203.46 205.14 199.67 201.74 200.12 27,227,400
93 Aug 14, 2019   203.16 206.44 202.59 202.75 201.12 36,547,400
94 Aug 13, 2019   201.02 212.14 200.48 208.97 207.29 47,218,500
95 Aug 12, 2019   199.62 202.05 199.15 200.48 198.87 22,474,900
96 Aug 09, 2019   201.30 202.76 199.29 200.99 199.37 24,619,700
97 Aug 08, 2019   200.20 203.53 199.39 203.43 201.03 27,009,500
```

After Corona data

```
     Date             Open   High   Low    Close  Avg.Close Volume
0   Jul 06, 2020   370.00 375.77 369.87 374.01 374.01 19,267,916
1   Jul 02, 2020   367.85 370.47 363.64 364.11 364.11 28,510,400
2   Jul 01, 2020   365.12 367.36 363.91 364.11 364.11 27,684,300
3   Jun 30, 2020   360.08 365.98 360.00 364.80 364.80 35,055,800
4   Jun 29, 2020   353.25 362.17 351.28 361.78 361.78 32,661,500
5   Jun 26, 2020   364.41 365.32 353.02 353.63 353.63 51,314,200
6   Jun 25, 2020   360.70 365.00 357.57 364.84 364.84 34,380,600
7   Jun 24, 2020   365.00 368.79 358.52 360.06 360.06 48,155,800
```



```
8  Jun 23, 2020   364.00 372.38 362.27 366.53 366.53 53,038,900
9  Jun 22, 2020   351.34 359.46 351.15 358.87 358.87 33,861,300
10 Jun 19, 2020   354.64 356.56 345.15 349.72 349.72 66,119,000
11 Jun 18, 2020   351.41 353.45 349.22 351.73 351.73 24,205,100
12 Jun 17, 2020   355.15 355.40 351.09 351.59 351.59 28,532,000
13 Jun 16, 2020   351.46 353.20 344.72 352.08 352.08 41,357,200
14 Jun 15, 2020   333.25 345.68 332.58 342.99 342.99 34,702,200
15 Jun 12, 2020   344.72 347.80 334.22 338.80 338.80 50,001,500
16 Jun 11, 2020   349.31 351.06 335.48 335.90 335.90 50,415,600
17 Jun 10, 2020   347.90 354.77 346.09 352.84 352.84 41,662,900
18 Jun 09, 2020   332.14 345.61 332.01 343.99 343.99 36,928,100
19 Jun 08, 2020   330.25 333.60 327.32 333.46 333.46 23,913,600
20 Jun 05, 2020   323.35 331.75 323.23 331.50 331.50 34,312,600
21 Jun 04, 2020   324.39 325.62 320.78 322.32 322.32 21,890,100
22 Jun 03, 2020   324.66 326.20 322.30 325.12 325.12 26,122,800
23 Jun 02, 2020   320.75 323.44 318.93 323.34 323.34 21,910,700
24 Jun 01, 2020   317.75 322.35 317.21 321.85 321.85 20,197,800
25 May 29, 2020   319.25 321.15 316.47 317.94 317.94 38,399,500
26 May 28, 2020   316.77 323.44 315.63 318.25 318.25 33,390,200
27 May 27, 2020   316.14 318.71 313.09 318.11 318.11 28,236,300
28 May 26, 2020   323.50 324.24 316.50 316.73 316.73 31,380,500
29 May 22, 2020   315.77 319.23 315.35 318.89 318.89 20,450,800
30 May 21, 2020   318.66 320.89 315.87 316.85 316.85 25,672,200
31 May 20, 2020   316.68 319.52 316.52 319.23 319.23 27,876,200
32 May 19, 2020   315.03 318.52 313.01 313.14 313.14 25,432,400
33 May 18, 2020   313.17 316.50 310.32 314.96 314.96 33,843,100
34 May 15, 2020   300.35 307.90 300.21 307.71 307.71 41,587,100
35 May 14, 2020   304.51 309.79 301.53 309.54 309.54 39,732,300
36 May 13, 2020   312.15 315.95 303.21 307.65 307.65 50,155,600
37 May 12, 2020   317.83 319.69 310.91 311.41 311.41 40,575,300
38 May 11, 2020   308.10 317.05 307.24 315.01 315.01 36,405,900
39 May 08, 2020   305.64 310.35 304.29 310.13 310.13 33,512,000
40 May 07, 2020   303.22 305.17 301.97 303.74 302.92 28,803,800
41 May 06, 2020   300.46 303.24 298.87 300.63 299.82 35,583,400
42 May 05, 2020   295.06 301.00 294.46 297.56 296.76 36,937,800
43 May 04, 2020   289.17 293.69 286.32 293.16 292.37 33,392,000
44 May 01, 2020   286.25 299.00 285.85 289.07 288.29 60,154,200
45 Apr 30, 2020   289.96 294.53 288.35 293.80 293.01 45,457,600
46 Apr 29, 2020   284.73 289.67 283.89 287.73 286.95 34,320,200
47 Apr 28, 2020   285.08 285.83 278.20 278.58 277.83 28,001,200
48 Apr 27, 2020   281.80 284.54 279.95 283.17 282.41 29,271,900
49 Apr 24, 2020   277.20 283.01 277.00 282.97 282.21 31,627,200
50 Apr 23, 2020   275.87 281.75 274.87 275.03 274.29 31,203,600
51 Apr 22, 2020   273.61 277.90 272.20 276.10 275.35 29,264,300
52 Apr 21, 2020   276.28 277.25 265.43 268.37 267.65 45,247,900
53 Apr 20, 2020   277.95 281.68 276.85 276.93 276.18 32,503,800
54 Apr 17, 2020   284.69 286.95 276.86 282.80 282.04 53,812,500
```

```
55 Apr 16, 2020   287.38 288.20 282.35 286.69 285.92 39,281,300
56 Apr 15, 2020   282.40 286.33 280.63 284.43 283.66 32,788,600
57 Apr 14, 2020   280.00 288.25 278.05 287.05 286.28 48,748,700
58 Apr 13, 2020   268.31 273.70 265.83 273.25 272.51 32,755,700
59 Apr 09, 2020   268.70 270.07 264.70 267.99 267.27 40,529,100
60 Apr 08, 2020   262.74 267.37 261.23 266.07 265.35 42,223,800
61 Apr 07, 2020   270.80 271.70 259.00 259.43 258.73 50,721,800
62 Apr 06, 2020   250.90 263.11 249.38 262.47 261.76 50,455,100
63 Apr 03, 2020   242.80 245.70 238.97 241.41 240.76 32,470,000
64 Apr 02, 2020   240.34 245.15 236.90 244.93 244.27 41,483,500
65 Apr 01, 2020   246.50 248.72 239.13 240.91 240.26 44,054,600
66 Mar 31, 2020   255.60 262.49 252.00 254.29 253.60 49,250,500
67 Mar 30, 2020   250.74 255.52 249.40 254.81 254.12 41,994,100
68 Mar 27, 2020   252.75 255.87 247.05 247.74 247.07 51,054,200
69 Mar 26, 2020   246.52 258.68 246.36 258.44 257.74 63,021,800
70 Mar 25, 2020   250.75 258.25 244.30 245.52 244.86 75,900,500
71 Mar 24, 2020   236.36 247.69 234.30 246.88 246.21 71,882,800
72 Mar 23, 2020   228.08 228.50 212.61 224.37 223.76 84,188,200
73 Mar 20, 2020   247.18 251.83 228.00 229.24 228.62 100,423,300
74 Mar 19, 2020   247.39 252.84 242.61 244.78 244.12 67,964,300
75 Mar 18, 2020   239.77 250.00 237.12 246.67 246.00 75,058,400
76 Mar 17, 2020   247.51 257.61 238.40 252.86 252.18 81,014,000
77 Mar 16, 2020   241.95 259.08 240.00 242.21 241.56 80,605,900
78 Mar 13, 2020   264.89 279.92 252.95 277.97 277.22 92,683,000
79 Mar 12, 2020   255.94 270.00 248.00 248.23 247.56 104,618,500
80 Mar 11, 2020   277.39 281.22 271.86 275.43 274.69 63,899,700
81 Mar 10, 2020   277.14 286.44 269.37 285.34 284.57 71,322,500
82 Mar 09, 2020   263.75 278.09 263.00 266.17 265.45 71,686,200
83 Mar 06, 2020   282.00 290.82 281.23 289.03 288.25 56,544,200
84 Mar 05, 2020   295.52 299.55 291.41 292.92 292.13 46,893,200
85 Mar 04, 2020   296.44 303.40 293.13 302.74 301.92 54,794,600
86 Mar 03, 2020   303.67 304.00 285.80 289.32 288.54 79,868,900
87 Mar 02, 2020   282.28 301.44 277.72 298.81 298.00 85,349,300
88 Feb 28, 2020   257.26 278.41 256.37 273.36 272.62 106,721,200
89 Feb 27, 2020   281.10 286.00 272.96 273.52 272.78 79,834,500
90 Feb 26, 2020   286.53 297.88 286.50 292.65 291.86 49,513,700
91 Feb 25, 2020   300.95 302.53 286.13 288.08 287.30 57,668,400
92 Feb 24, 2020   297.26 304.18 289.23 298.18 297.38 55,548,800
93 Feb 21, 2020   318.62 320.45 310.50 313.05 312.20 32,388,500
94 Feb 20, 2020   322.63 324.65 318.21 320.30 319.44 25,141,500
95 Feb 19, 2020   320.00 324.57 320.00 323.62 322.75 23,496,000
96 Feb 18, 2020   315.36 319.75 314.61 319.00 318.14 38,132,800
97 Feb 14, 2020   324.74 325.98 322.85 324.95 324.07 20,028,400
98 Feb 13, 2020   324.19 326.22 323.35 324.87 323.99 23,686,900
```

Conclusion

Hence, web scraping is a very useful technique for businesses and organizations to extract, parse, analyze and visualize the data for research and other purposes. Stock market being source of very dynamic data, when it comes to stock market analysis, it is said that by looking at the history of a stock's trading activity, one will find all the relevant information needed. This is because price action repeats itself as a result of investors patterned behavior. Hence, getting accurate stock market analysis is extremely important for any financial actions.

Code

```
#!/usr/bin/env python3

# -*- coding: utf-8 -*-




import os

from bs4 import BeautifulSoup

import requests

import pandas as pd

from pandas import Series, DataFrame

import numpy as np

from datetime import datetime, timedelta

import time

from pandas.tseries.offsets import BDay

import matplotlib.pyplot as plt


# For Before Corona

#end = datetime(2019,12,27)

#start = end - BDay(100)


# For After Corona

start = datetime(2019,12,27)

end = datetime.today()


baseUrl = 'https://finance.yahoo.com/quote/'
```

```
tech_list = 'AAPL'

# Function to get stock data, plot graph and store data in csv
def getStockData(tick, endPeriod, startPeriod):
    end = endPeriod
    start = startPeriod
    print(start,end)

    #Convert Dates to UNIX Time for Params
    unixStart = int(time.mktime(start.timetuple()))
    unixEnd = int(time.mktime(end.timetuple()))

    # Url link to yahoo finance with search parameters
    url = (baseUrl + str(tick) + '/history?period1=' + str(unixStart) +
'&period2=' + str(
        unixEnd) + '&interval=1d&filter=history&frequency=1d')
    scrapWebsite(url)

# Function to scrap website
def scrapWebsite(url):

    # Send request to website
    result = requests.get(url)
    c = result.content

    # Set Beautiful Soup object
    soup = BeautifulSoup(c, "lxml")
    summary = soup.find('div', {'class': 'Pb(10px) Ovx(a) W(100%)'})

    # Fetch table with historical data
    tables = summary.find_all('table')
    # Array to store table rows
    data = []
    rows = tables[0].find_all('tr')
    for tr in rows:
        cols = tr.findAll('td')
        if len(cols) == 7:
            for td in cols:
                text = td.find(text=True)
                data.append(text)

    # Create Pandas DataFrame object
    dFrame = pd.DataFrame(np.array(data).reshape(int(len(data) / 7), 7))
    dFrame.columns = ['Date', 'Open', 'High', 'Low', 'Close', 'Aclose',
'Volume']
    dFrame.set_index('Date', inplace=False)

    # Call function to store data to csv file
    dump2csv(dFrame)

    # Call function to Calculate SMA and EMA and plot graph
    movingAverage(dFrame)

# Function to Calculate SMA and EMA and plot graph
def movingAverage(stockprices):
    # For printing close and type casting its values to int
    stockprices['Close'] = stockprices['Close'].astype(float)

    # Calculate SMA
    stockprices['SMA 20 Days'] = stockprices['Close'].rolling(20).mean()
    stockprices['SMA 50 Days'] = stockprices['Close'].rolling(50).mean()

    # Calculate EMA
    stockprices['EMA 20 Days'] = stockprices['Close'].ewm(span=20, 
adjust=False).mean()

    stockprices['EMA 50 Days'] = stockprices['Close'].ewm(span=50, 
adjust=False).mean()


    # PLot Graph of Price CLose, SMA and EMA Data

    stockprices[['Close', 'SMA 20 Days', 'EMA 50 Days', 'SMA 50 Days', 'EMA 
20 Days']].plot(figsize=(12, 8))

    plt.grid(True)

    plt.title('AAPL Moving Averages')

    plt.axis('tight')

    plt.ylabel('Price')


# Funciton to Store data to CSV

def dump2csv(dataFrame):

    filename = str(tech_list) + '.csv'


    # If file already exists, append data to end of file, otherwise create 
new file with Column Names

    if os.path.exists(filename):

        append_write = 'a'

        dataFrame.to_csv(filename, sep='\t', mode=append_write, header=False)

    else:

        append_write = 'w'

        dataFrame.to_csv(filename, sep='\t', mode=append_write)



# Call the main function to run the script

getStockData(tech_list, end, start)
```