

Time Series Analysis of Oil Prices

CMSC 6950: Computer Based Research Tools and Applications



Contributors:

Group 5

Aafia Jabeen - 201790648

Rabeya Akhter - 201793299

Sanusi Musa - 201581949

Shiplu Das - 201891201

Index

1. Introduction
2. Makefile
3. Data Downloading
 - a. Prerequisites for the project
 - b. Pipenv
 - c. Download the Data
 - d. Histogram
4. Clean the data
5. Analyzing and plotting
 - a. Understanding the data
 - b. Dot Plot
 - c. Box Plot
 - d. Swarm Plot
 - e. Box + Swarm Plot
 - f. Bar Chart
 - g. Histogram
 - h. Pie Chart
6. Test Suite Implementation
7. Conclusion
8. References

Introduction

In this project, we are working with the oil price dataset to analyze the prices of oil with respect to time. Firstly, we created a python function to download the dataset from web automatically. The downloaded dataset contains the Brent oil price data (xml) file and csv files containing the stock prices of different companies like Shell (RDSB), British Petroleum (BP), Cairn Energy (CNE), Premier Oil (PMO), etc. The oil price (xml) contains the following columns: ***Date*** and ***Brent Oil Price*** and the share price (csv) contains the following columns: ***Date, Open, High, Low, Close, Adj Close, Volume***.

Since machine learning has been widely used in Oil and Gas sector, and the economy of a country is greatly affected by its oil prices, therefore we have chosen this dataset to work with.

The address of our git lab repository is: <https://gitlab.com/cmsc6950/team5/time-series-analysis>

Some of the main libraries used in this project are:

1. **Numpy** and **Pandas** for mathematical functions and arrays
2. **Matplotlib**, **Seaborn** and **Bokeh** for Visualization and Plotting

Makefile

We have created a Makefile in our project to automate the various commands required to run the project. Makefile helps in organizing the code compilation and makes it easy to run the project. A single '\$make' command will clean the project directory of already generated png or html files, the downloaded and cleaned data (in case it is there) and cache files. It will then execute download.py to download the data from web to a new folder in the project directory. After the data is downloaded, it will execute cleanData.py to clean the

data and generate cleaned csv files. It will then execute different plot files (.py) to generate the univariate and multivariate plots and will also give the results of different test cases executed through test suite (test_all_cases.py).

To run the project through Makefile:

- To call the Makefile and execute all:

```
- $ make
```

- To remove the downloaded data, processed data and other compiled files:

```
- $ make clean
```

- To download the data and generate histogram of numerical attributes of a csv file (CNE.L):

```
- $ make download
```

- To clean the downloaded data and generate cleaned csv files:

```
- $ make processData
```

- To generate different plots:

```
- $ make plotting
```

- To generate the results of different test cases through test suite:

```
- $ make tests
```

Data Downloading

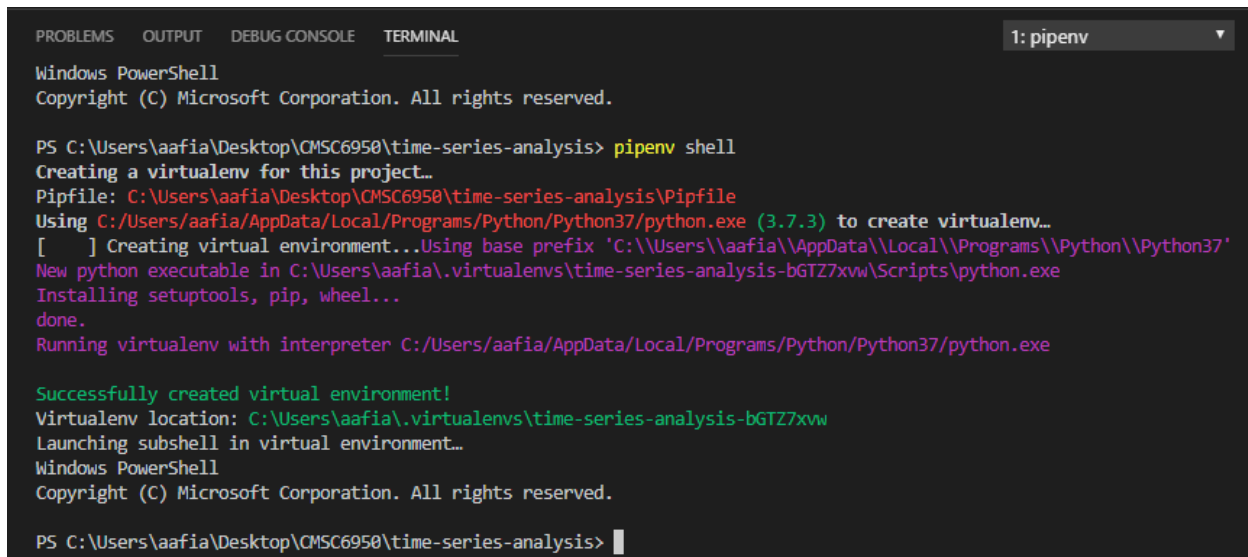
■ Prerequisites

- Make sure you have Python 3 and pipenv installed. To install pipenv, run:

```
- $pip install pipenv
```

- After installation, activate the shell

- `$pipenv shell`



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\aaafia\Desktop\CMSC6950\time-series-analysis> pipenv shell
Creating a virtualenv for this project...
Pipfile: C:\Users\aaafia\Desktop\CMSC6950\time-series-analysis\Pipfile
Using C:/Users/aaafia/AppData/Local/Programs/Python/Python37/python.exe (3.7.3) to create virtualenv...
[ ] Creating virtual environment...Using base prefix 'C:\\Users\\aaafia\\AppData\\Local\\Programs\\Python\\Python37'
New python executable in C:\Users\aaafia\virtualenvs\time-series-analysis-bGTZ7xvw\Scripts\python.exe
Installing setuptools, pip, wheel...
done.
Running virtualenv with interpreter C:/Users/aaafia/AppData/Local/Programs/Python/Python37/python.exe

Successfully created virtual environment!
Virtualenv location: C:\Users\aaafia\virtualenvs\time-series-analysis-bGTZ7xvw
Launching subshell in virtual environment..
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\aaafia\Desktop\CMSC6950\time-series-analysis>

```

Fig.1 Command line execution of `$pipenv shell`

Pipenv will generate two files: *pipfile* and *pipfile.lock*. The *pipfile* will list all the python packages installed for the project and the *pipfile.lock* will cover the dependencies and version numbers of those packages and libraries. It kind of changes the way the dependencies are stored on the system while working with python.

The file '*download.py*' can be used to download the data from web. Since, we are using a kaggle dataset, install kaggle:

- `$pipenv install Kaggle`

The function '*DownloadDataset*' uses python **requests** library through Kaggle API and **argparse** to download the zipped folder from kaggle website. The execution of this function will unzip the folder and will download the files in a new folder called '*downloaded_data*'.

This file will also print the header rows and columns of four csv files for the companies British Petroleum, Premier Oil, Shell, Cairn Energy: BP. L, PMO.L,

RDSB.L and CNE.L. It will generate a histogram as *'histogram_columns_cne.png'* using *matplotlib.pyplot* to analyze and to get a better understanding of the numerical attributes in the dataset. The histogram is further saved in plots folder.

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: pipenv +
PS C:\Users\aaafia\Desktop\CMSC6950\time-series-analysis> python src/download.py
  Date    Open    High    Low    Close    Adj Close    Volume
0 2000-01-03 622.5  622.5  622.5  291.545807  622.5  0.0
1 2000-01-04 596.0  596.0  596.0  279.134583  596.0  0.0
2 2000-01-05 596.0  596.0  596.0  279.134583  596.0  0.0
3 2000-01-06 602.0  602.0  602.0  281.944702  602.0  0.0
4 2000-01-07 597.0  597.0  597.0  279.603088  597.0  0.0
  Date    Open    High    Low    Close    Adj Close    Volume
0 2000-01-03 17.274  17.274  17.274  17.274  17.274  1490554
1 2000-01-04 16.605  16.605  16.605  16.605  16.605  0
2 2000-01-05 16.362  16.362  16.362  16.362  16.362  0
3 2000-01-06 21.229  21.229  21.229  21.229  21.229  2183438
4 2000-01-07 16.115  16.115  16.115  16.115  16.115  0
  Date    Open    High    Low    Close    Adj Close    Volume
0 2000-05-15 1973.550049 1997.910034 1953.530029 1193.597412 1980.130005 5477576.0
1 2000-05-16 1970.069946 2010.959961 1942.219971 1191.409302 1976.500000 6786513.0
2 2000-05-17 1955.270020 1956.140015 1919.599976 1164.911255 1932.540039 4175377.0
3 2000-05-18 1940.479980 2022.280029 1931.780029 1204.315063 1997.910034 5365214.0
4 2000-05-19 2004.869995 2038.810059 1911.770020 1215.334106 2016.189941 4296496.0
  Date    Open    High    Low    Close    Adj Close    Volume
0 2000-01-03 30.236000 30.236000 30.236000 29.328886 30.236000 0.0
1 2000-01-04 29.622999 29.622999 29.622999 28.734276 29.622999 0.0
2 2000-01-05 29.214001 29.214001 29.214001 28.337547 29.214001 0.0
3 2000-01-06 28.601999 28.601999 28.601999 27.743906 28.601999 0.0
4 2000-01-07 29.622999 29.622999 29.622999 28.734276 29.622999 0.0

src/download.py Downloading the dataset: time-series-analysis-of-oil-prices from Kaggle to the folder:./src/downloaded_data
Plotting histogram for each numerical attribute to understand the type of data

```

Fig. 2 Executing download.py

Histogram depicting the columns for Cairn Energy (CNE)

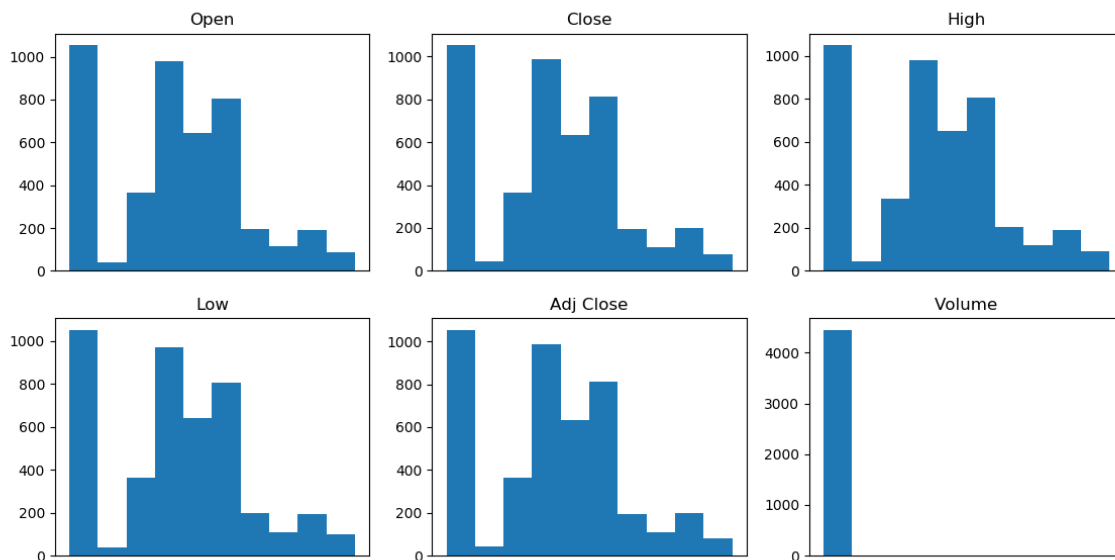


Fig. 3 Histogram plot of columns in CNE.L.csv file generated by download.py

Clean the data

To clean the data files, run

- **\$python src/cleanData.py**

We have used **pandas.DataFrame** to load the csv files:

- Here del command is used to remove it from all csv file.
 - **del df['Volume']**
- To convert string to datetime, we used
 - **pd.to.datetime()**
- To set the index of DataFrame using existing column 'Date', we have used
 - **df1 = df1.set_index('Date')**
- Got rid of the unnecessary rows in data files
 - **df1 = df1.dropna()**

In our project cleanData.py script takes care of raw data.

- This script imports the data from the data files (.csv)
- It generates new cleaned csv files with two additional column - Year and Month.
- Saves all cleaned csv files in new_data folder.

This command will create new csv files.

- **\$python cleanData.py**

Result:

- new_RDSB.L.csv
 - new_BP.L.csv
 - new_PMO.L.csv
 - new_CNE.L.csv
-
- CleanData.py also clean the xml and csv files and will store the data in a main data frame called all_data.
 - It generates new cleaned csv files with two additional fields - Year and Month.
 - plotting.py imports the module from cleanData.py file. This file can be used to generate clean dataframe for individual csv files. This file is optional since we have not used it in plotting.
 - We can also generate the shrink data by specifying the value for 'last' and 'year' (last 5 years data)
 - Arguments like 'all' or 'brate' can be passed with data function:
 - cleandata("all") : for getting the combined dataframe
 - cleandata("brate") : for getting the oil prices

Analyzing and Plotting

▪ Understanding the data:

Before jumping into the plotting, it's important to understand what message the excel file data is giving.

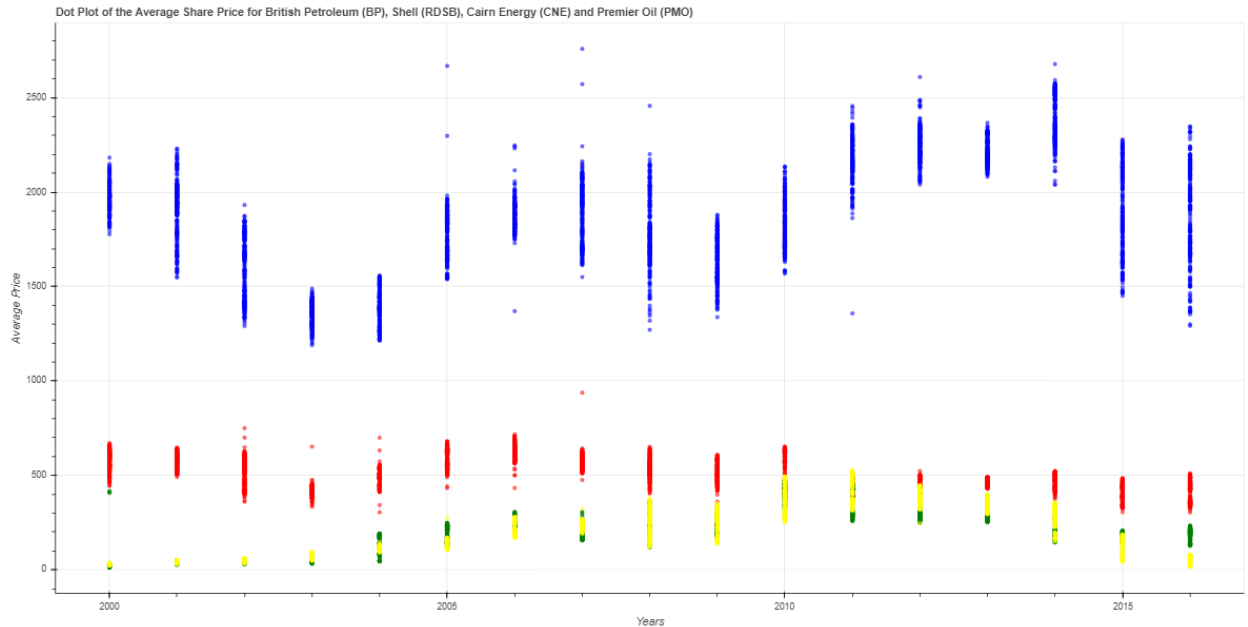
	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Adj Close	Volume
2	03/01/2000	622.5	622.5	622.5	291.545807	622.5	0
3	04/01/2000	596	596	596	279.134583	596	0
4	05/01/2000	596	596	596	279.134583	596	0
5	06/01/2000	602	602	602	281.944702	602	0
6	07/01/2000	597	597	597	279.603088	597	0
7	10/01/2000	579.5	579.5	579.5	271.407043	579.5	0
8	11/01/2000	569.5	569.5	569.5	266.723389	569.5	0
9	12/01/2000	570	570	570	266.95752	570	0
10	13/01/2000	570.5	570.5	570.5	267.191925	570.5	0
11	14/01/2000	553	553	553	258.995728	553	0
12	17/01/2000	550.5	550.5	550.5	257.824921	550.5	0
13	18/01/2000	556	556	556	260.400787	556	0
14	19/01/2000	562.5	562.5	562.5	263.445007	562.5	0
15	20/01/2000	543	543	543	254.31221	543	0
16	21/01/2000	562.5	562.5	562.5	263.445007	562.5	0
17	24/01/2000	576	576	576	269.767639	576	0
18	25/01/2000	561	561	561	262.742462	561	0
19	26/01/2000	555	555	555	259.932465	555	0
20	27/01/2000	556	556	556	260.400787	556	0
21	28/01/2000	555.5	555.5	555.5	260.166626	555.5	0

If we consider a real life scenario, suppose a share company starts its operation at 8 AM and ends at 4 PM. So, from the dataset, the “Open” value represents the starting operation of the company and “Close” value represent the end figure of the operation for a specific date.

However, the “High” and “Low” values represent the share price goes up and fall within a day.

▪ Dot plot:

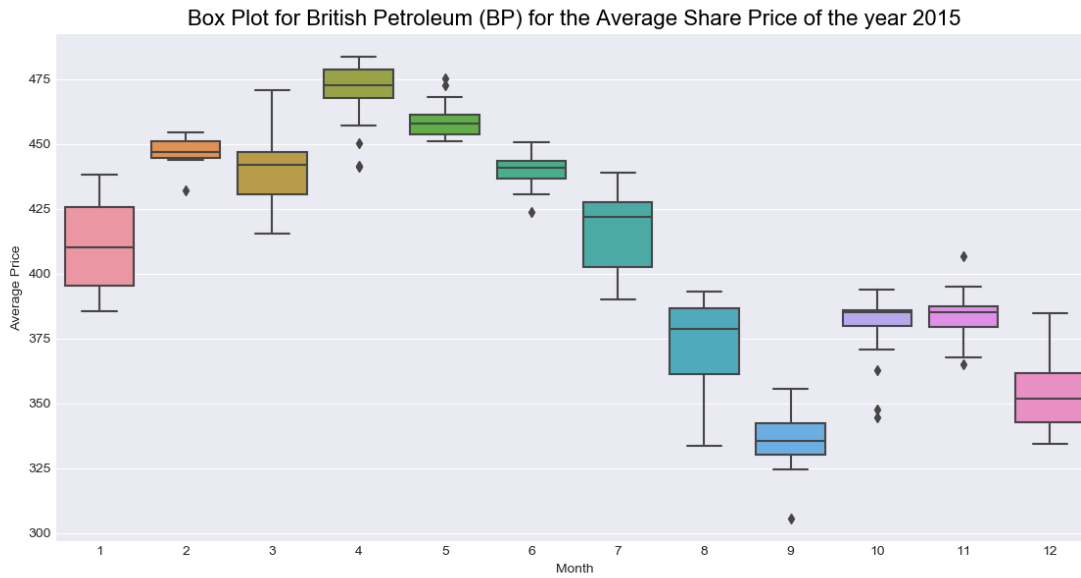
In the dot plotting section of the project, we tried to demonstrate the average price of all the four share companies over a period of 17 years from 2000 2017. While running the script, the user will be asked to enter start year and an end year. From the given input, the average of the “High” and “Low” value will be calculated for the given time period. The bokeh library has been used to create the plotting; therefore, the plotting will be opened in a browser window and a dot_plot.html file will be generated into the plots folder.



Using the data from the dataset we can conclude that the average value for the Shell (RDSB) company has always been higher compared to other companies.

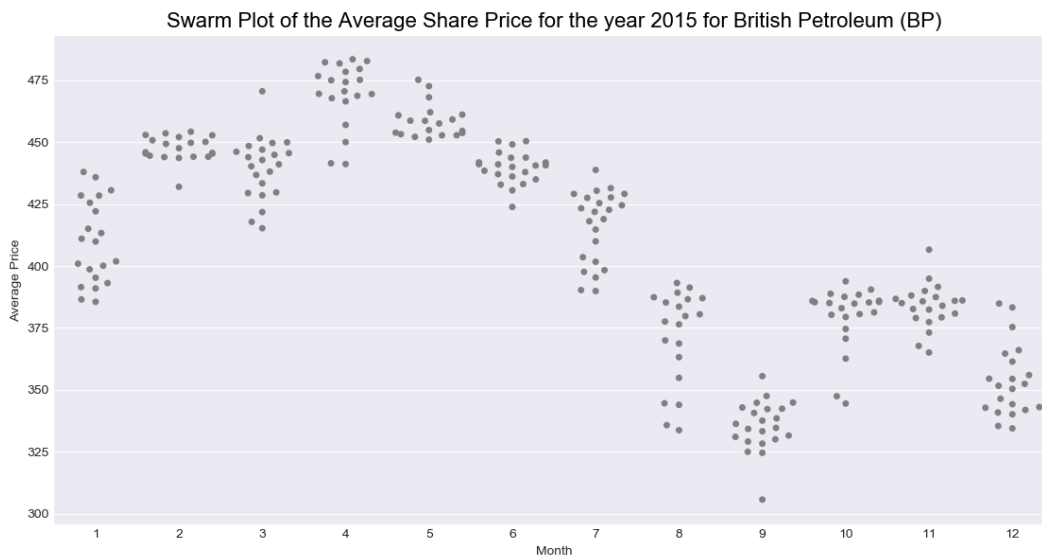
■ Box Plot:

The box plot diagram represents the average price of the British Petroleum (BP) for the year 2015. According to the box plot concept, the line in the middle of the box represents the mean value. However, 25% of the data should remain above the line and 25% data should be below the line. The top and edge line represent the maximum and minimum value the data can go. Beyond that line are the outliers.



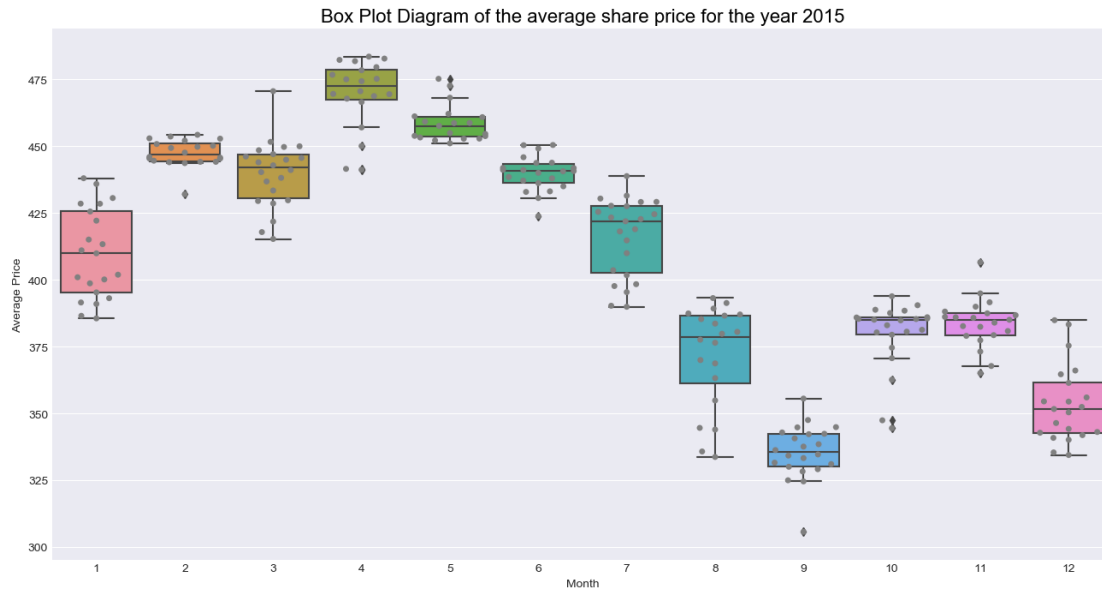
■ Swarm Plot:

The swarm plot is pretty much like dot plots. In the figure given below, the swarm plot shows the average share price for the year 2015 for British Petroleum (BP). On an average, there are 22 dots representing the working days of a month on x-axis and average price on y-axis.



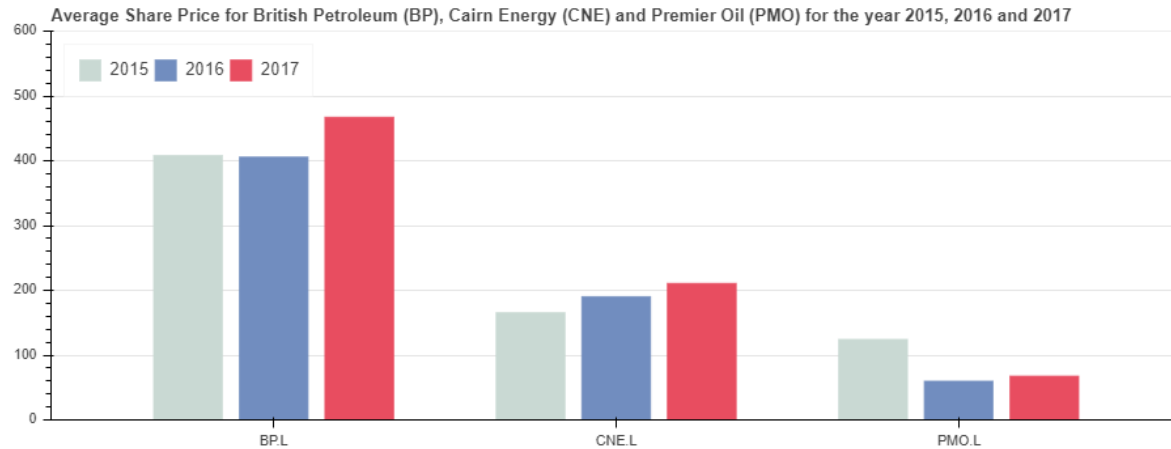
■ Box plot + Swarm Plot

The figure given below shows the combination of box plot and swarm plot diagram



■ Bar Chart

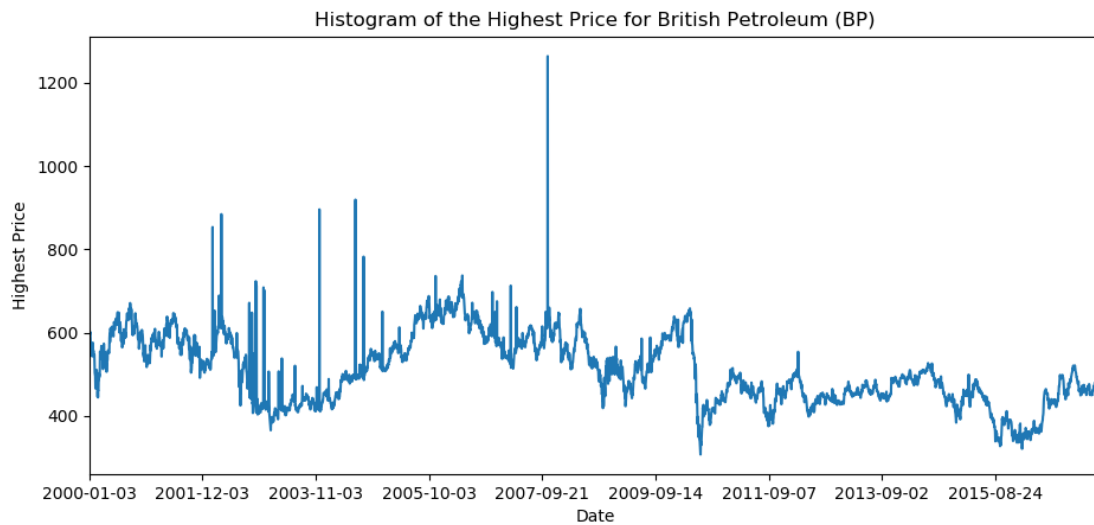
The bar chart plotting shows the average price of share for three different companies for the year 2015, 2016 and 2017.



Note: for the year 2017, the dataset contains the record for only 7 months.

■ Histogram:

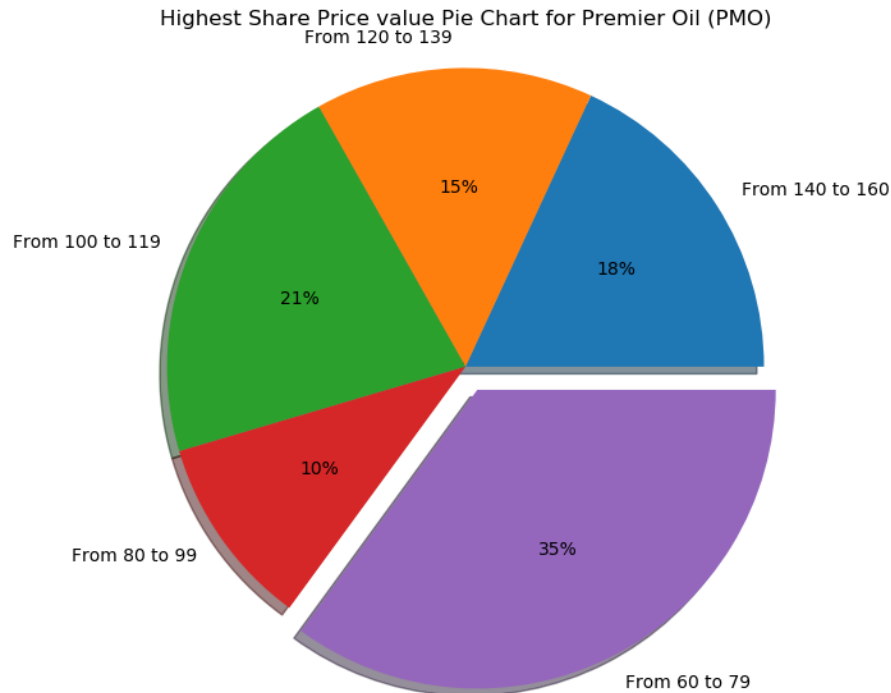
Histogram is the next plot of the project. This simple diagram shows the fluctuation of highest price of the share market over a period of 17 years.



■ Pie Chart

Lastly, the pie chart is plotted. As an example, we have chosen the highest price of British Petroleum (BP). We have implemented five conditions to test the variation in highest price from \$60 to \$160 range.

From the figure above it can be seen that 35% of the time the highest price of British Petroleum (BP) remains within \$60 to \$79 for the range of \$60 to \$160.



Note:

In certain cases, we have found inconsistency in the plotting. After examining the data from the dataset, we have noticed huge gap between companies which is shown below:

	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Adj Close	Month	Year
2	03/01/2000	622.5	622.5	622.5	291.545807	622.5	1	2000
3	04/01/2000	596	596	596	279.134583	596	1	2000
4	05/01/2000	596	596	596	279.134583	596	1	2000
5	06/01/2000	602	602	602	281.944702	602	1	2000
6	07/01/2000	597	597	597	279.603088	597	1	2000
7	10/01/2000	579.5	579.5	579.5	271.407043	579.5		2000
8	11/01/2000	569.5	569.5	569.5	266.723389	569.5	BP.L	2000
9	12/01/2000	570	570	570	266.95752	570	1	2000
10	13/01/2000	570.5	570.5	570.5	267.191925	570.5	1	2000
11	14/01/2000	553	553	553	258.995728	553	1	2000
12	17/01/2000	550.5	550.5	550.5	257.824921	550.5	1	2000

	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Adj Close	Month	Year
2	03/01/2000	17.274	17.274	17.274	17.274	17.274	1	2000
3	04/01/2000	16.605	16.605	16.605	16.605	16.605	1	2000
4	05/01/2000	16.362	16.362	16.362	16.362	16.362	1	2000
5	06/01/2000	21.229	21.229	21.229	21.229	21.229	1	2000
6	07/01/2000	16.115	16.115	16.115	16.115	16.115	1	2000
7	10/01/2000	16.115	16.115	16.115	16.115	16.115	1	2000
8	11/01/2000	27.472	27.472	27.472	27.472	27.472	1	2000
9	12/01/2000	27.056	27.056	27.056	27.056	27.056	1	2000
10	13/01/2000	17.83	17.83	17.83	17.83	17.83	1	2000

	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Adj Close	Month	Year
2	03/01/2000	30.236	30.236	30.236	29.328886	30.236	1	2000
3	04/01/2000	29.622999	29.622999	29.622999	28.734276	29.622999	1	2000
4	05/01/2000	29.214001	29.214001	29.214001	28.337547	29.214001	1	2000
5	06/01/2000	28.601999	28.601999	28.601999	27.743906	28.601999	1	2000
6	07/01/2000	29.622999	29.622999	29.622999	28.734276	29.622999	1	2000
7	10/01/2000	29.214001	29.214001	29.214001	28.337547	29.214001	1	2000
8	11/01/2000	30.236	30.236	30.236	29.328886	30.236	1	2000
9	12/01/2000	29.622999	29.622999	29.622999	28.734276	29.622999	1	2000
10	13/01/2000	29.214001	29.214001	29.214001	28.337547	29.214001	1	2000
11	14/01/2000	30.236	30.236	30.236	29.328886	30.236	1	2000
12	17/01/2000	31.052999	31.052999	31.052999	30.121372	31.052999	1	2000
13	18/01/2000	31.052999	31.052999	31.052999	30.121372	31.052999	1	2000

	A	B	C	D	E	F	G	H
1	Date	Open	High	Low	Close	Adj Close	Month	Year
2	15/05/2000	1973.550049	1997.910034	1953.530029	1193.597412	1980.130005	5	2000
3	16/05/2000	1970.069946	2010.959961	1942.219971	1191.409302	1976.5	5	2000
4	17/05/2000	1955.27002	1956.140015	1919.599976	1164.911255	1932.540039	5	2000
5	18/05/2000	1940.47998	2022.280029	1931.780029	1204.315063	1997.910034	5	2000
6	19/05/2000	2004.869995	2038.810059	1911.77002	1215.334106	2016.189941	5	2000
7	22/05/2000	2013.569946	2049.25	1997.040039	1216.383057	2017.930054	5	2000
8	23/05/2000	2032.719971	2032.719971	1983.98999	1200.643921	1991.819945	5	2000
9	24/05/2000	1992.689941	2013.569946	1919.599976	1163.404297	1930.040039	5	2000
10	25/05/2000	1938.73999	1954.400024	1860.430054	1127.737427	1870.869995	5	2000
11	26/05/2000	1879.569946	1891.75	1853.459961	1123.017456	1863.040039	5	2000

The reason for the huge difference in price can be related to the quality of crude oil. Some companies like Shell (RDSB) sell high quality crude oil which can be marked of higher price as compared to other companies.

Testing

Testing code help to catch mistakes or avoid getting them into production in the first place.

- We have created a test file named test_all_cases.py


```

src ▸ test_all_cases.py ▸ {} sys
19
20 #Function part to test new.CNE.L.csv file is in new_data folder
21 def checkcsv2():
22     try:
23         if os.path.isfile('./src/new_data/new.CNE.L.csv'):
24             return True
25     except Exception as e:
26         print(e)
27
28 #Function part to test new.PM0.L.csv file is in new_data folder
29 def checkcsv3():
30     try:
31         if os.path.isfile('./src/new_data/new.PM0.L.csv'):
32             return True
33     except Exception as e:
34         print(e)
35
36 #Function part to test new.RDSB.L.csv file is in new_data folder
37 def checkcsv4():
38     try:
39         if os.path.isfile('./src/new_data/new.RDSB.L.csv'):
40             return True
41     except Exception as e:
42         print(e)
43
44 #Function part to test download_data folder is not empty
45 def download_data():
46     try:
47         if any(os.scandir('./src/downloaded_data')):
48             return True
49     except Exception as e:
50         print(e)
51

```

- Test suite:

Pytest: A robust Python testing tool, pytest can be used for all types and levels of software testing.

We have used Python Assert statement to demonstrate the calculation works.

- Assertions are checks that return either True or False status.
- In pytest, if an assertion fails in a test method, then that method execution is stopped there. The remaining code in that test method is not executed, and pytest will continue with the next test method.

This script executes the test suite and report the test execution status.


```
$py.test ./src/test_all_cases.py
```

Result

```
py.test ./src/test_all_cases.py
===== test session starts =====
platform linux -- Python 3.7.3, pytest-4.3.1, py-1.8.0, pluggy-0.9.0
rootdir: /home/erfan/Documents/project cmcsc6950/v4/time-series-analysis, inifile:
plugins: remotedata-0.3.1, openfiles-0.3.2, doctestplus-0.3.0, arraydiff-0.3
collected 6 items

src/test_all_cases.py ..... [100%]

===== 6 passed in 0.05 seconds =====
```

Conclusion

We have generated plots and graphs of some of the popular companies from Oil and Gas sector. We can conclude to the fact that the prices of these companies differ according to their quality of crude oil. Also, the plots comparing the share prices of different companies can gives us a picture of the share market and can be helpful in case we want to invest in the shares of any of these companies. Since, oil prices is always a huge discussion and affects a country's economy greatly, it is beneficial and interesting to get an inside image of the share prices of the giants (Shell, British Petroleum, Cairn Energy, Premier Oil) of Oil and Gas sector.

References

1. Kaggle Dataset : <https://www.kaggle.com/javierbravo/oil-price-and-share-price-of-a-few-companies>
2. Oil price dataset downloaded from the U.S Energy Information administration
<https://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=PET&s=rbrte&f=D>

3. Royal Dutch Shell share price dataset:

<https://uk.finance.yahoo.com/quote/RDSB.L/history?period1=946684800&period2=1499122800&interval=1d&filter=history&frequency=1d>

4. Presentation repository link: <https://gitlab.com/rakhter/rakhter.gitlab.io.git>