Project: Capstone Project 1: Data Wrangling Report (6 - 10 Hours)

The data for my analysis was retrieved using native API from the provider. Initially the plan was to use API method to get data from QANDL similar to the examples on the exercises. However, since the data wasn't readily available for the stocks I was interested in (SRAX, ESV), I used Yahoo Finance instead. Before the data could be retrieved using Yahoo Finance, I had to install software called "fix_yahoo_finance" in Python. One I installed it, data retrieval was seamless. The link to Jupyter Notebook that contains all the executed notes for data wrangling can be found following this link (click here). It would've been quite beneficial to be able to use the Python methods to extract data using API, or some other downloaded CSV file. Nevertheless, there will be more opportunities during the course to apply those skills. For this exercise, I was quite interested in exploring the data using Python, creating visualizations in Python to see if there are any clear outliers, how to remove outliers etc. if there were any outliers.

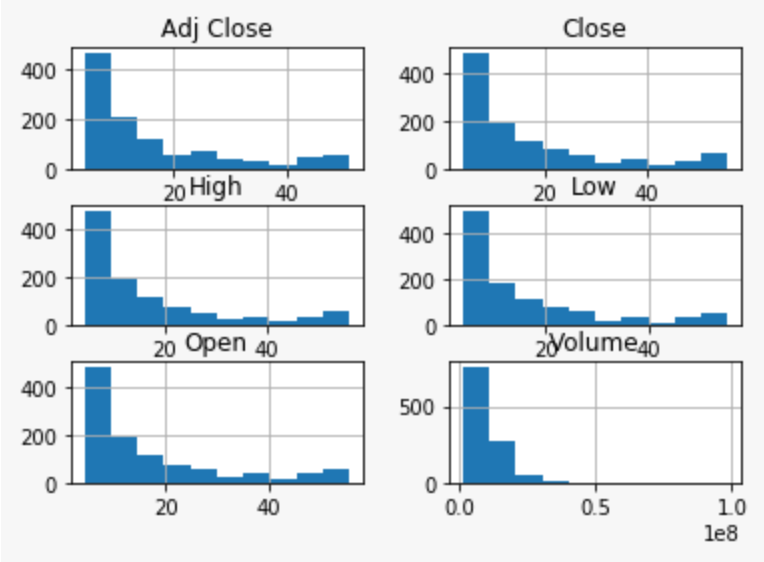
First thing I did was looked at first few rows using head() function to see how the data looked, and also visualized some metrics by plotting stock prices. I also looked at different stock prices for each day like Open, High, Low etc.

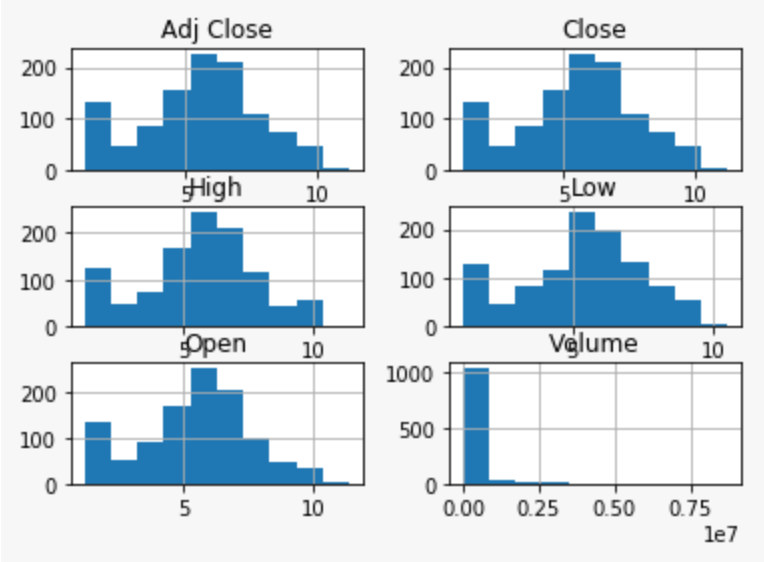
Date	Open	High	Low	Close	Adj Close	Volume
2014-05-01	50.270000	50.369999	49.750000	49.820000	45.184910	2726900
2014-05-02	49.930000	50.330002	49.820000	49.930000	45.284676	2226700
2014-05-05	49.869999	50.799999	49.750000	50.750000	46.028385	3078600
2014-05-06	50.770000	51.250000	50.360001	51.150002	46.391174	2093000
2014-05-07	51.259998	51.529999	50.689999	51.500000	46.708607	1794100

```
# Download Option1: Yahoo Finance
# Need to install fix_yahoo_finance using command <pip install fix_yahoo_finance> on terminal
>>> y_esv = yf.download(stock1,start_dt, end_dt)
>>> y_srax = yf.download(stock2,start_dt, end_dt)
>>> y_srax.Close.plot()
>>> y_esv.Close.plot()
>>> plt.show()
```

Also, looked at some histograms to see the distribution of different stock prices.

```
>>> y_esv.hist()
>>> y_srax.hist()
>>> plt.show()
```





Looked at some histograms to see the distribution of different stock prices.

```
>>> y_esv.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1089 entries, 2014-05-01 to 2018-08-24
Data columns (total 6 columns):
Open          1089 non-null float64
High          1089 non-null float64
Low           1089 non-null float64
Close         1089 non-null float64
Adj Close     1089 non-null float64
Volume        1089 non-null int64
dtypes: float64(5), int64(1)
memory usage: 59.6 KB
```

```
>>> y_srax.info()
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1089 entries, 2014-05-01 to 2018-08-24
Data columns (total 6 columns):
Open         1089 non-null float64
High         1089 non-null float64
Low          1089 non-null float64
Close        1089 non-null float64
Adj Close    1089 non-null float64
Volume       1089 non-null int64
dtypes: float64(5), int64(1)
memory usage: 59.6 KB
```

Stock prices were available for everyday all the way through to 2014, hence there were no missing data. Also, the data was quite clean not requiring any additional data manipulation to start analysis. I looked at other metrics like max, min, mean, median etc. for all available prices.

```
# Get min open, high, low and close for the selected period
>>> mio=round(y_esv.Open.min(),2)
>>> mih=round(min(y_esv.High),2)
>>> mil=round(min(y_esv.Low),2)
>>> mic=round(min(y_esv.Close),2)
```

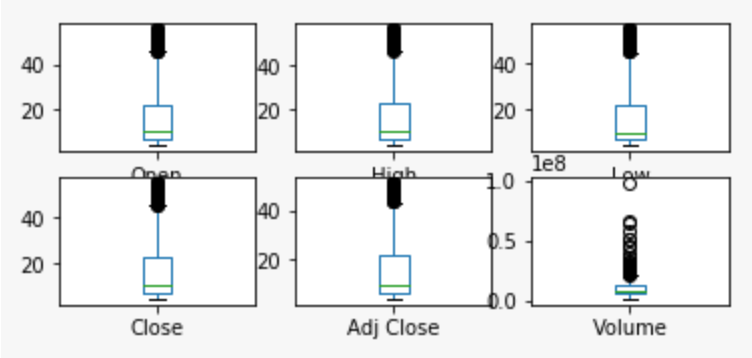
There were outliers, and they were removed using a function that I found online that discussed exploratory data analysis. The function basically takes the median of the input data, finds the distance of all data points from the median, finds the median of the distance and finally divides all the distances by the median distance. This results in numbers 0 through 2. The function then passes two parameters, initially the data and then the median distance of 2. The dataset that has outliers will end up having median distance of over 2, so they get excluded from the dataset.

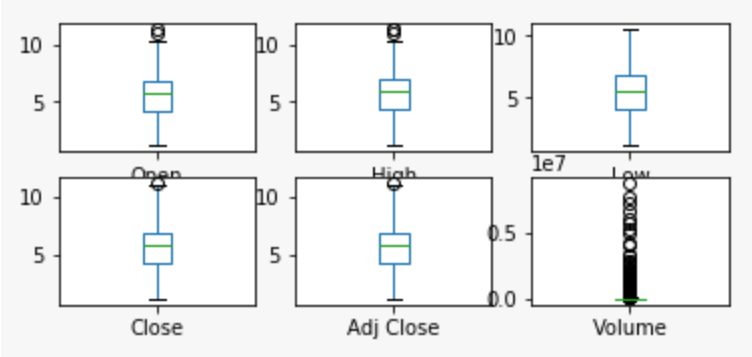
```
def reject_outliers_2(data, m = 2.):
    d = np.abs(data - np.median(data))
    mdev = np.median(d)
    s = d/(mdev if mdev else 1.)
    return data[s<m]

>>> esv_no=reject_outliers_2(y_esv)
>>> srax_no=reject_outliers_2(y_srax)
```

Outliers were checked using before and after box plots.

Before removing outliers:

```
>>> y_esv.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
>>> y_srax.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
>>> plt.show()
```





After removing outliers:

```
#let's look at boxplot again to see if outliers are removed
>>> esv_no.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
>>> srax_no.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
>>> plt.show()
```