AI60003 : MODULE 2
ASSIGNMENT-2 REPORT
DEBANJAN SAHA (19CS30014)

>> Problem Statement: Building a time series forecasting model to predict the stock prices of yahoo.

>> External Reference: This assignment was implemented by me in the following notebook: **Google Colab Notebook** (Please wait a few seconds after opening the notebook for the images to load)

>> Dataset:

➔ **Link:** https://www.kaggle.com/arashnic/time-series-forecasting-with-yahoo-stock-price

➔ **Size of Dataset:** The dataset consists of **1825 rows and 7 columns** covering the stock price historical data of YAHOO from **23rd Nov 2015 to 20th Nov 2020**

➔ **Description of Dataset:** The dataset gives us information about 6 different things for each particular date. These are as follows:

- **High =>** Highest Price of the stock for that particular date
- **Low =>** Lowest Price of the stock for that particular date
- **Open =>** Opening Price of the stock
- **Close =>** Closing Price of the stock
- **Volume =>** Total amount of Trading Activity
- **AdjClose =>** Adjusted values factor in corporate actions such as dividends, stock splits, and new share issuance

>> Implementation:

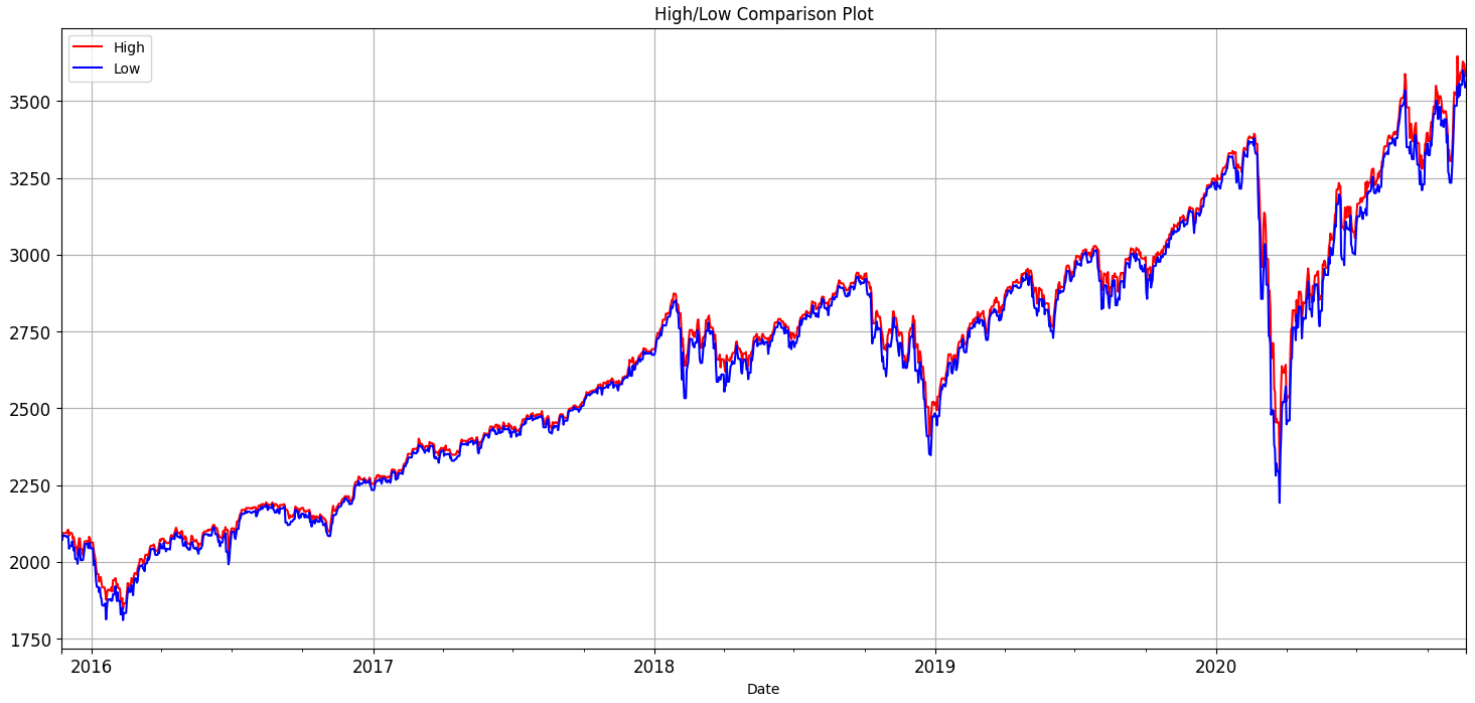
➔ **Data Analysis:**

- Using several **pandas dataframe analysis library functions**, the dataset was found to have **no null entries** and so **no data imputation was required** in this case. Also, the number of unique data points were exactly the same as the size of the dataset = 1825.

	Date	High	Low	Open	Close	Volume	Adj Close
0	2015-11-23	2095.610107	2081.389893	2089.409912	2086.590088	3.587980e+09	2086.590088
1	2015-11-24	2094.120117	2070.290039	2084.419922	2089.139893	3.884930e+09	2089.139893
2	2015-11-25	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
3	2015-11-26	2093.000000	2086.300049	2089.300049	2088.870117	2.852940e+09	2088.870117
4	2015-11-27	2093.290039	2084.129883	2088.820068	2090.110107	1.466840e+09	2090.110107
...	...	...	...	...	...	...	...
1820	2020-11-16	3628.510010	3600.159912	3600.159912	3626.909912	5.281980e+09	3626.909912
1821	2020-11-17	3623.110107	3588.679932	3610.310059	3609.530029	4.799570e+09	3609.530029
1822	2020-11-18	3619.090088	3567.330078	3612.090088	3567.790039	5.274450e+09	3567.790039
1823	2020-11-19	3585.219971	3543.840088	3559.409912	3581.870117	4.347200e+09	3581.870117
1824	2020-11-20	3581.229980	3556.850098	3579.310059	3557.540039	2.236662e+09	3557.540039

1825 rows × 7 columns

- I plotted the graphs of YAHOO stock **'Open' and 'Close'** Prices together and then plotted the **'High' and 'Low'** prices together. The results are as follows:

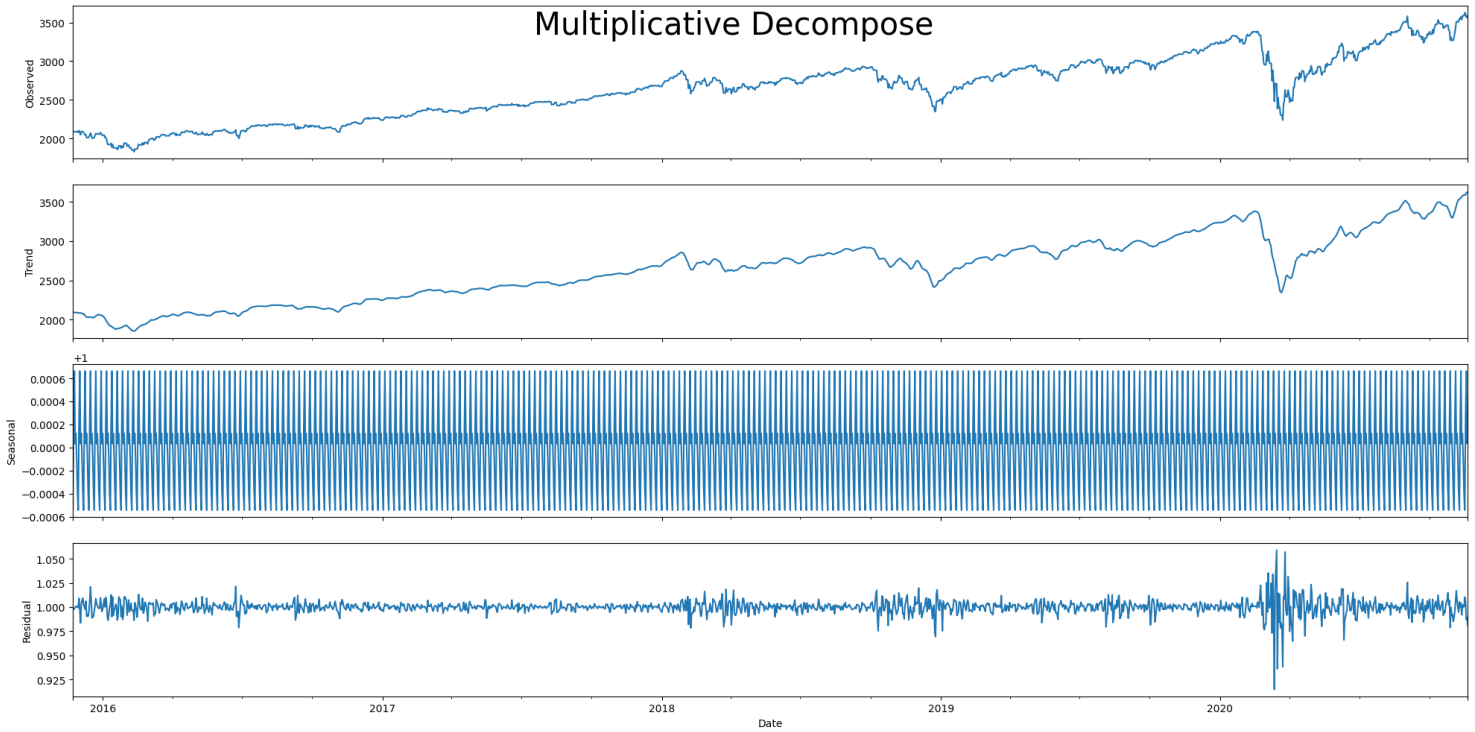


- The four parameters don't vary much compared with each other. However, one important observation here is that around 2020 March-April, due to the pandemic, the stock prices crashed rapidly and this observation will be helpful to choose our train and test split carefully.

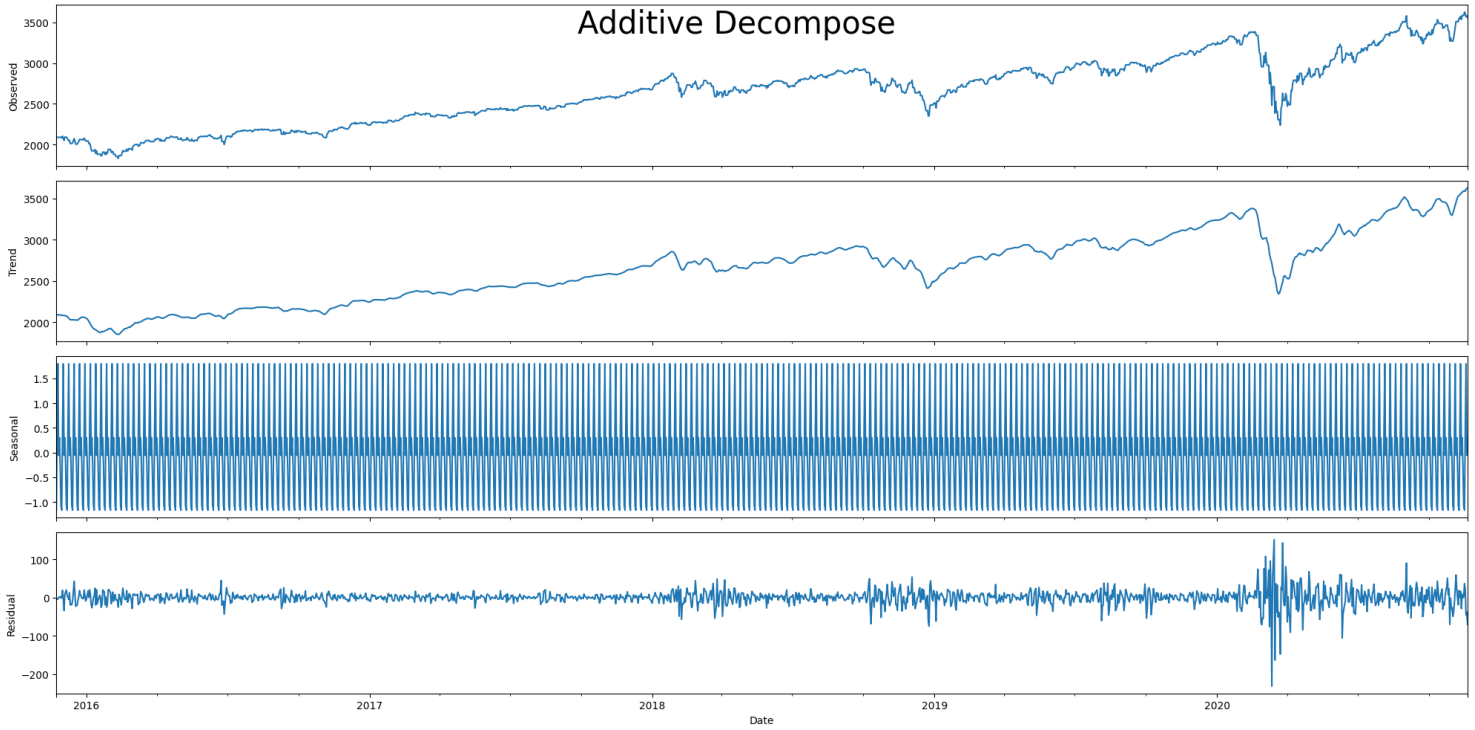
➔ <u>Decomposition of Time Series Data:</u>

- The main components of a time series such as **Trend, Seasonal, Residual** were plotted with the help of standard library functions using the **stock closing prices** of yahoo

- In the multiplicative model, the original time series **is expressed as the product of trend, seasonal and irregular components**. Under this model, the trend has the same units as the original series, but the seasonal and irregular components are unitless factors, distributed around 1.



- In an additive time series, the components add together to make the time series. If you have an increasing trend, you still see roughly the same size peaks and troughs throughout the time series. This is often seen in indexed time series where the absolute value is growing but changes stay relative



➢ **Inference :**

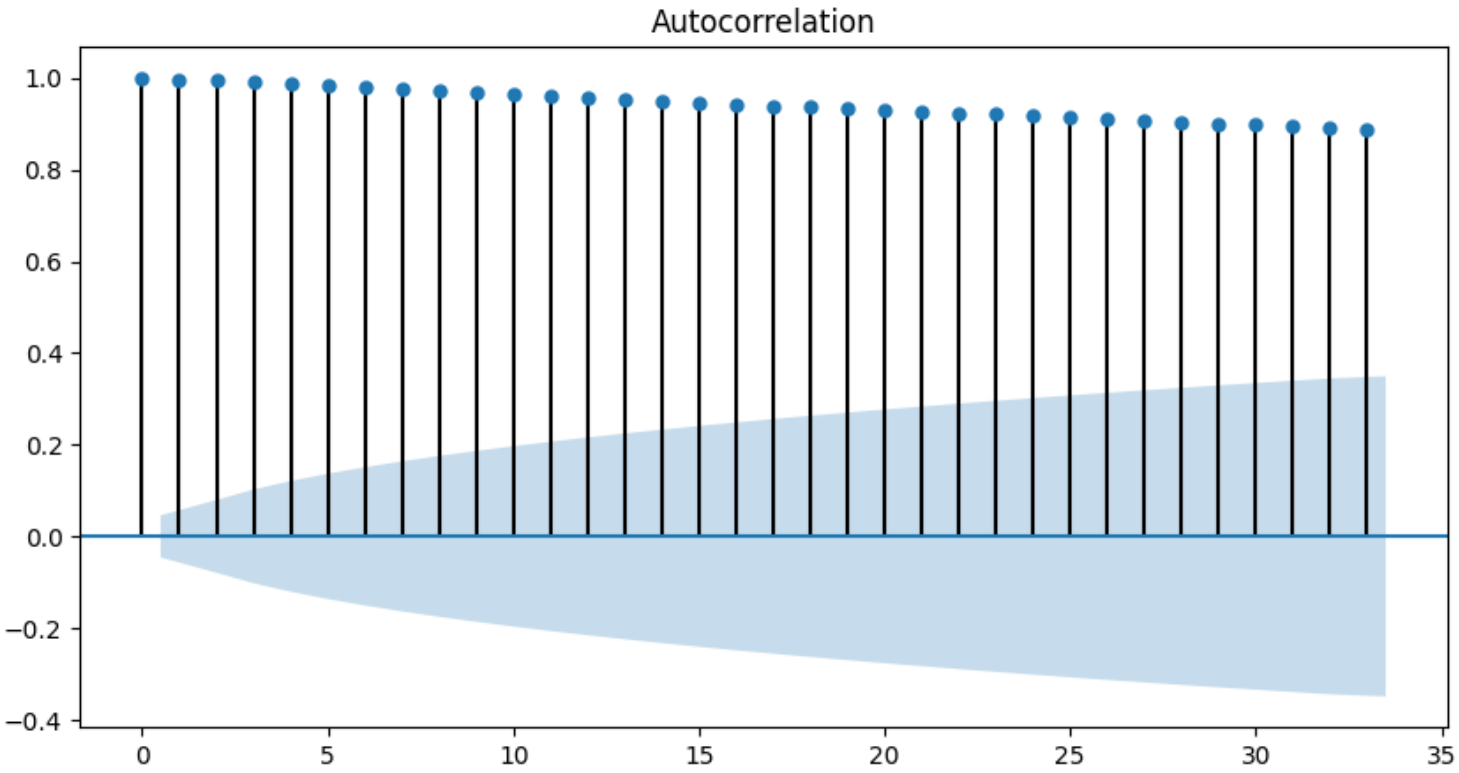
Date	Seasonal	Trend	Residual	Actual_Values
2015-11-23	-1.164921	2093.707970	-5.952962	2086.590088
2015-11-24	0.458303	2092.533927	-3.852337	2089.139893
2015-11-25	1.797703	2091.359883	-4.287469	2088.870117
2015-11-26	-0.055060	2089.114362	-0.189185	2088.870117
2015-11-27	0.301771	2088.231480	1.576856	2090.110107
...	...	...	...	...
2020-11-16	-1.164921	3591.649972	36.424861	3626.909912
2020-11-17	0.458303	3587.705706	21.366021	3609.530029
2020-11-18	1.797703	3611.870212	-45.877876	3567.790039
2020-11-19	-0.055060	3619.821080	-37.895903	3581.870117
2020-11-20	0.301771	3627.771948	-70.533680	3557.540039

1825 rows × 4 columns

- **The additive model is useful when the seasonal variation is relatively constant over time. The multiplicative model is useful when the seasonal variation increases over time.**
- The residuals plotted should not be following any kind of pattern, it should be spread randomly. It is evident from the plot as well.
- Trend and Seasonality information obtained from the series does seem genuine. The residuals show periods of high variability during the rapid ups and downs in the time series (**especially during the first outbreak of pandemic**)
- The trend can be clearly observed in the plots above. We can also see that the additive model represents the series as a sum of seasonality, trend and residual.

➔ **Autocorrelation:**

- The plot of autocorrelation for the **stock closing prices** of the dataset is as follows:



➢ **Inference:**

- Now,we have an ACF plot. In simple terms, it describes how well the present value of the series is related with its past values. A time series can have components like trend, seasonality, cyclic and residual. ACF considers all these components while finding correlations hence it's a 'complete auto-correlation plot'.
- **The autocorrelation plot starts with a very high autocorrelation at lag 1 but slowly declines until it becomes negative and starts showing an increasing negative autocorrelation. This type of pattern indicates a strong autocorrelation, which can be helpful in predicting future trends.**

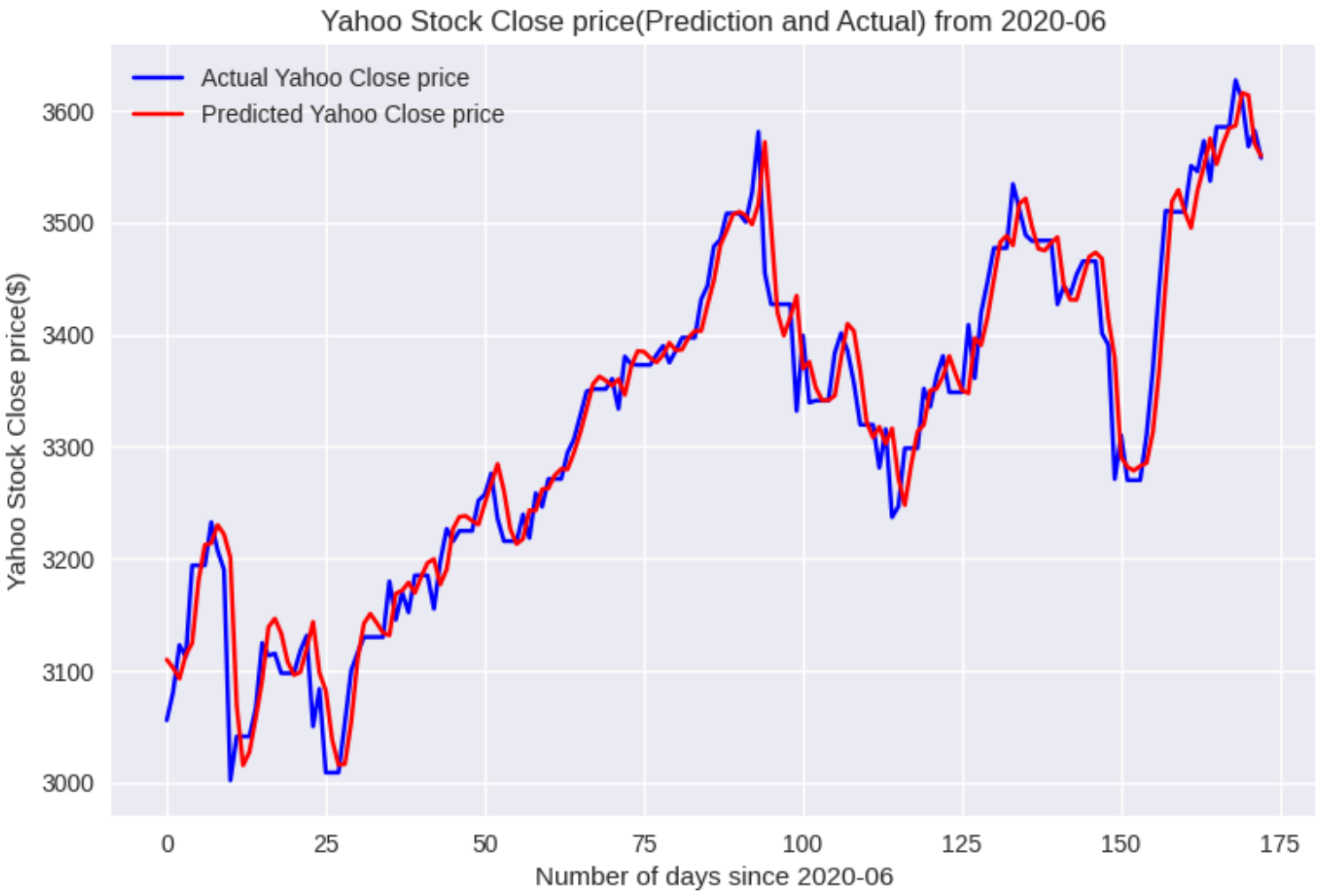
➔ **Building the Model:**

- A **prediction window** of **60** days has been taken for predicting future values and based on that the training set will be prepared. Keeping that as one of the training input dimensions, I have built the model as follows.
- The model is implemented as a **Sequential** model from **Keras** library and consists of **3 hidden LSTM** layers each of size **50**
- **Adam Optimizer** has been used for optimising the loss function
- **MSE** (mean squared error) has been used as the loss function.
- A separate **callback method** has been implemented to save the model with the minimum loss function possible to avoid the model when the loss function overshoots the optimum value using **the Keras EarlyStopping** callback

➔ **Training and Testing**:

- Splitting the dataset: I made an approximately **95% - 5%** train-test split of the given dataset. The dataset was **not split on the exact given percentage** but on the **basis of a time period**.

- I have filtered the dataset **before June-2020** and used it as a training set and the remaining dataset as the test set. The reason behind such a split is so that the pandemic stock price crash event can be accounted for while the prices have crashed and just rising up around June.
- Based on that future predictions can be made and the model can be thoroughly tested for the rest of the 2020 data from June. Otherwise, if the model is trained up to 2019, it might predict an absurd increase in the stock price in 2020 based on past years' data which would be inaccurate.
- The training set has been generated using data points from the column **'Close'** (**The closing price of the stocks**) from all the rows of the dataset and with a **prediction window of 60 days.**
- With these implementation details, the model was trained and the state with minimum loss was saved.
- Then testing was done on the data after **2020-06(June-2020)** by generating the set of predictions with the test set as input to the optimized model.
- The plot of the test set prediction was found as given below.
- The **average error** on the test set prediction was found to be **5.204%**



➔ **Future value Prediction :**

- Using the trained model, a prediction for the next day after this dataset ends has been made using the last window of 60 days.
- Result: For **21st November 2020**, **$ 3547.903** is the **predicted closing price** for the yahoo stocks.
- After checking the plot and the last few rows of the dataset and keeping in mind that the average error was 5%(appx) for the test set, the above prediction can be considered a reasonable and decent prediction.

	Date	High	Low	Open	Close	Volume	Adj Close
1820	2020-11-16	3628.510010	3600.159912	3600.159912	3626.909912	5.281980e+09	3626.909912
1821	2020-11-17	3623.110107	3588.679932	3610.310059	3609.530029	4.799570e+09	3609.530029
1822	2020-11-18	3619.090088	3567.330078	3612.090088	3567.790039	5.274450e+09	3567.790039
1823	2020-11-19	3585.219971	3543.840088	3559.409912	3581.870117	4.347200e+09	3581.870117
1824	2020-11-20	3581.229980	3556.850098	3579.310059	3557.540039	2.236662e+09	3557.540039

1825 rows × 7 columns

- Due to some limitations on the yahoo finance API and unavailability of open-source yahoo stock prices around November-December 2020 and before, further prediction errors could not be calculated. However, if we see the last few days' data, we can see this prediction is pretty close to them and is very likely to have an insignificant error.