DATA ANALYTICS

(CS40003)

Term Project

Auto-regression analysis with
time-series data for future event prediction

SUBMITTED BY:

KARTIK THAKKER (17IM10035)
PRERIT JAIN (16IM30033)

SUPERVISED BY:
PROF. DEBASIS SAMNTHA

CONTENT

PROBLEM STATMENT

We are supposed to apply auto-regression analysis with time-series data for predicting the stock value in the month of October 2017. We are also supposed to build the covariance matrix for different values of p, where p is the order of the auto-regression mode

THEORY

INTRODUCTION TO TIME SERIES:

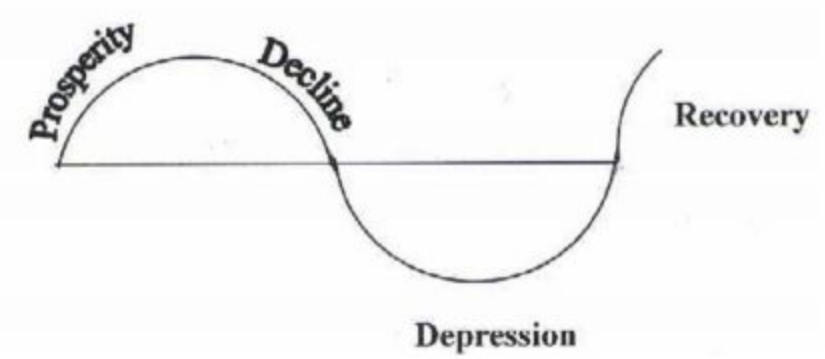
The general structure of a time-series is a collection of data points over an interval of time (usually at equal intervals). Examples of time series data include changing demand of a product over time, measurement of heart rates over time, etc.

It is mathematically defined as a set of vectors $x(t), t = 0,1,2,...$ where t represents the time elapsed. The variable $x(t$ is treated as a random variable. The measurements taken during an event in a time series are arranged in proper chronological order.

A time series containing records of a single variable is termed as univariate. But if records of more than one variable are considered, it is termed as multivariate. A time series can be continuous or discrete. In a continuous time-series, observations are measured at every instance of time, whereas a discrete-time series contains observations measured at discrete points of time.

The variations in the measured characteristic over time may be attributed to one of the following reasons :

1. Random variations: Irregular or random variations in a time series are caused by unpredictable influences, which are not regular and also do not repeat in a particular pattern. These variations are caused by incidences such as war, strike, earthquake, flood, revolution, etc. There is no defined statistical technique for measuring random fluctuations in a time series.
2. Trend: The general tendency of a time series to increase, decrease or stagnate over a long period of time is termed as Secular Trend or simply Trend. Thus, it can be said that trend is a long term movement in a time series. For example, series relating to population growth, the number of houses in a city, etc. show an upward trend, whereas the downward trend can be observed in a series relating to mortality rates, epidemics, etc.
3. Seasonality: Seasonal variations in a time series are fluctuations within a year during the season. The important factors causing seasonal variations are climate and weather conditions, customs, traditional habits, etc. For example sales of ice-cream increase in summer, sales of woolen cloths increase in winter. Seasonal variation is an important factor for businessmen, shopkeepers, and producers for making proper future plans.
4. Cyclic variations: The cyclical variation in a time series describes the medium-term changes in the series, caused by circumstances, which repeat in cycles. The duration of a cycle extends over a longer period of time, usually two or more years. Most of the economic and financial time series show some kind of cyclical variation.



Considering the effects of these four components, two different types of models are generally used for a time series viz. Multiplicative and Additive models.

STATIONARITY OF TIME SERIES

The concept of stationarity of a stochastic process can be visualized as a form of statistical equilibrium. The statistical properties such as mean and variance of a stationary process do not depend upon time. It is a necessary condition for building a time series model that is useful for future forecasting. Further, the mathematical complexity of the fitted model reduces with this assumption. Some mathematical tests like the one given by Dickey and Fuller are generally used to detect stationarity in a time series data.

Dickey-Fuller test:

The Dickey-Fuller test is testing if $\phi=0$ in this model of the data:

$$y_t = \alpha + \beta t + \phi y_{t-1} + e_t$$

which is written as

$$\Delta y_t = y_t - y_{t-1} = \alpha + \beta t + \gamma y_{t-1} + e_t$$

where y_t is your data. It is written this way so we can do a linear regression of Δy_t against t and y_{t-1} and test if γ is different from 0. If $\gamma=0$ then we have a random walk process. If not and $-1<1+\gamma<1$ then we have a stationary process.

AUTO-COVARIANCE MATRIX

If the {Xn} process is weakly stationary, the covariance of Xn and Xn+k depends only on the lag k. This leads to the following definition of the “autocovariance” of the process:

$$\gamma(k) = \text{cov}(X_{n+k}, X_n)$$

The autocorrelation function, ρ(k), is defined by

$$\rho(k) = \gamma(k)\ \gamma(0)$$

This is simply the correlation between Xn and Xn+k.

There are additional constraints on γ(k) beyond symmetry and maximum value at zero lag. To illustrate, assume Xn and Xn+1 are perfectly correlated, Xn+1 and Xn+2 perfectly correlated. It is clearly nonsensical to set the correlation between Xn and Xn+2 to zero. To obtain the additional constraints define the p × 1 vector

$$X' = [X_{n+1}, X_{n+2}, \ldots, X_{n+p}]$$

and a set of constants a

$$a' = [a_1, a_2, \ldots a_p]$$

Consider now the following linear combination of the elements of X:

$$Z = \sum a_i X_i = a'X$$

The variance of Z is a' ΣXX a where ΣXX is the p × p covariance matrix of the random vector X. If {Xn} is a stationary random process ΣXX has the following special form:

$$\Sigma_{XX} = \sigma^2 \begin{bmatrix} 1 & \rho(1) & \rho(2) & \ldots & \rho(p-1) \\ \rho(1) & 1 & \rho(1) & \ldots & \rho(p-2) \\ \rho(2) & \rho(1) & 1 & \ddots & \\ \vdots & & & \ddots & \rho(1) \\ \rho(p-1) & & & \rho(1) & 1 \end{bmatrix}$$

This is known as the Auto-covariance matrix for the data set.

Auto-regressive models:

• **Intuition** – Autoregressive models are based on the idea that current value of the series, Xt , can be explained as a linear combination of p past values, Xt−1, Xt−2, . . . , Xt−p, together with a random error in the same series.

• **Definition** – An autoregressive model of order p, abbreviated AR(p), is of the form

$$Xt = \varphi 1 Xt-1 + \varphi 2 Xt-2 + \cdots + \varphi p Xt-p + wt = X\ p\ i=1\ \varphi i Xt-i + wt$$

where Xt is stationary, wt ~ wn(0, σ2 w), and φ1, φ2, . . . , φp (φp 6= 0) are model parameters. The hyperparameter p represents the length of the "direct look back" in the series.

• **Backshift Operator** – The backshift operator is defined as

$$BXt = Xt-1.$$

It can be extended,

$$B2Xt = B(BXt) = B(Xt-1) = Xt-2, \text{ and so on.}$$

Thus, $$B\,kXt = Xt-k$$

• We can also define an inverse operator (f**orward-shift operator**) by enforcing

$$B\,-1B = 1, \text{ such that}$$
$$Xt = B\,-1BXt = B\,-1Xt-1.$$

By using the backshift operator we can rewrite it as

$$Xt - \varphi 1Xt-1 - \varphi 2Xt-2 - \cdots - \varphi pXt-p = wt\ (1 - \varphi 1B - \varphi 2B\ 2 - \cdots - \varphi pB\ p\ )Xt = wt$$

• The autoregressive operator is defined as:

$$\varphi(B) = 1 - \varphi 1B - \varphi 2B\ 2 - \cdots - \varphi pB\ p = 1 - X\ p\ j=1\ \varphi jB\ j,$$

then the AR(p) can be rewritten more concisely as:

$$\phi(B)X_t = w_t$$

AR(1) can be given by $Xt = \varphi 1Xt-1 + wt$.

• Mean: $E[Xt\ ] = 0$
• Variance: $Var(Xt) = \sigma\ 2\ w\ (1 - \varphi\ 2\ 1\ )$

IMPLEMENTATION

STEP 1: DATASET EXPLORATION

The analysis starts with checking the dataset size and the column descriptions, to check the type of models which can be used to predict future values of stock prices.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12074 entries, 0 to 12073
Data columns (total 7 columns):
Date       12074 non-null object
Open       12074 non-null float64
High       12074 non-null float64
Low        12074 non-null float64
Close      12074 non-null float64
Volume     12074 non-null int64
OpenInt    12074 non-null int64
dtypes: float64(4), int64(2), object(1)
memory usage: 660.4+ KB
```

We observe that there are no null values, hence no treatment is required. Also, all information is of float/Int data type, hence type conversions are not required.

```
data.describe()
```

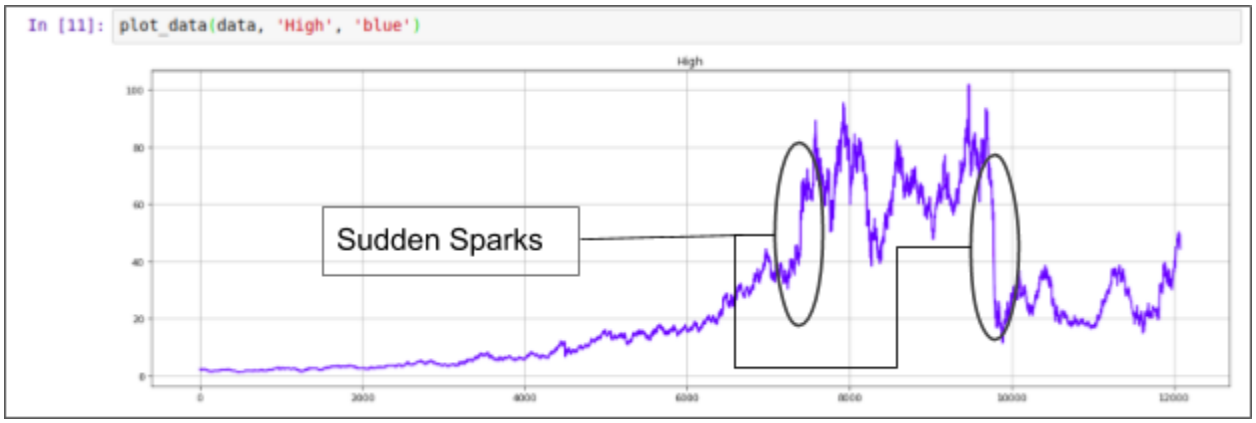
	Open	High	Low	Close	Volume	OpenInt
count	12074.000000	12074.000000	12074.000000	12074.000000	1.207400e+04	12074.0
mean	24.419762	24.751516	24.056557	24.406203	3.337757e+06	0.0
std	23.647989	23.990163	23.260402	23.627596	5.203407e+06	0.0
min	1.252800	1.273000	1.232800	1.252800	0.000000e+00	0.0
25%	4.536150	4.593400	4.452900	4.536150	5.334490e+05	0.0
50%	16.453000	16.662000	16.243000	16.474000	1.395958e+06	0.0
75%	35.095750	35.507500	34.575000	35.001500	3.202536e+06	0.0
max	99.930000	101.820000	97.212000	98.864000	1.159616e+08	0.0

We observe that the variable ‘OpenInt’ has constant value throughout the data set, hence we can drop the variable as it has no predictive power. Also, we have four different time series corresponding to the opening value, closing value, highest value and the lowest value of the stock price at a particular day. Following is a snippet showing some of the example rows in our dataset.

```
data.tail()
```

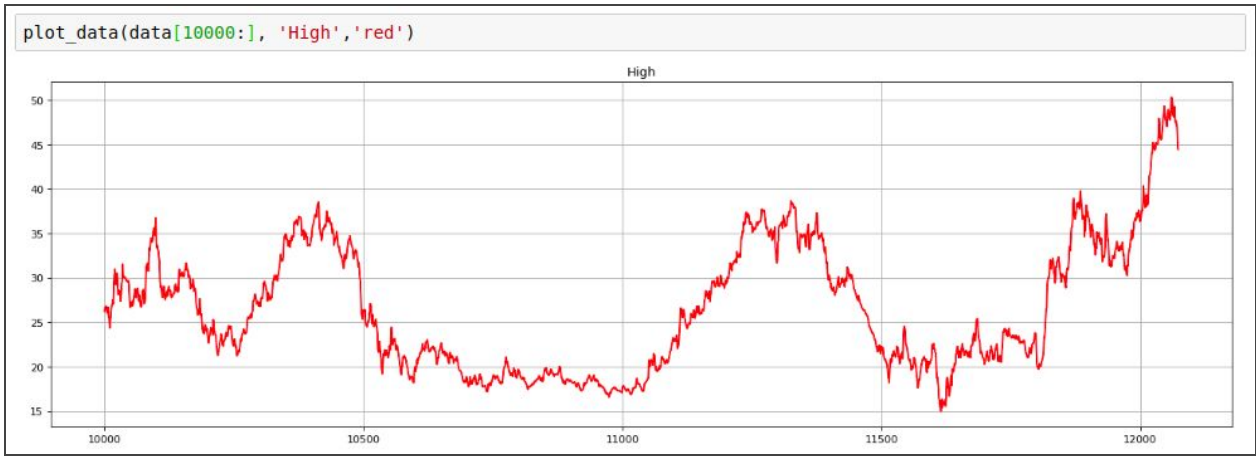
	Date	Open	High	Low	Close	Volume	OpenInt
12069	2017-11-06	47.20	47.6800	46.53	47.29	1725638	0
12070	2017-11-07	47.11	47.2042	46.54	46.97	2168351	0
12071	2017-11-08	46.73	46.7700	45.37	45.89	3347930	0
12072	2017-11-09	44.71	44.7100	43.11	43.33	7671810	0
12073	2017-11-10	42.93	44.4600	42.75	43.01	4463839	0

STEP 2: DATA VISUALIZATION



Above is the graph of the high stock value v/s time, we can see that the graph has sudden sparks at two points and assignable causes can be assigned to these sparks, and after that the time series is stable.

Hence, we consider the data after 2008 to predict the values of the near future. Belo is the graph after 2008. The below dataset is used for building the prediction model.



The above is the dataset whose future values need to be predicted. So, we will apply the Auto-regression analysis for building a predictive model to predict the future values of the above time series.

STEP 3: ENSURING STATIONARITY

As we all know that Auto-regression models are only applicable to stationary datasets, so checking stationarity is one of the major steps. We apply an augmented dickey fuller test to find out whether the above-shown data is stationary or not.

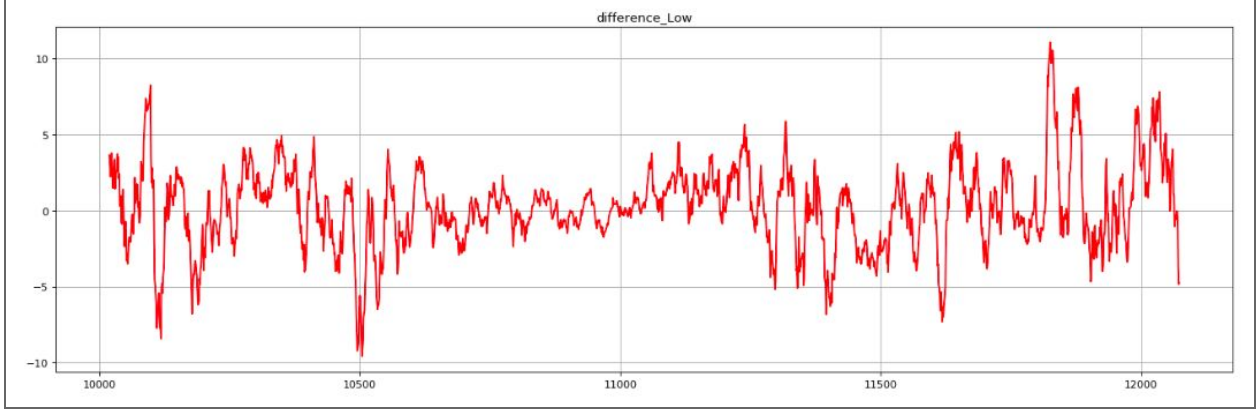
```
print("AUGMENTED DICKY FULLER TEST FOR STATIONARITY \n")

print("RESULTS FOR HIGH VALUES \n")
adf_stationarity_test(data[10000:],'High')

print("RESULTS FOR LOW VALUES \n")
adf_stationarity_test(data[10000:],'Low')

print("RESULTS FOR OPEN VALUES \n")
adf_stationarity_test(data[10000:],'Open')

print("RESULTS FOR CLOSE VALUES \n")
adf_stationarity_test(data[10000:],'Close')
```

```
AUGMENTED DICKY FULLER TEST FOR STATIONARITY

RESULTS FOR HIGH VALUES

ADF Statistic: -1.197394
p-value: 0.674716
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568


RESULTS FOR LOW VALUES

ADF Statistic: -1.333855
p-value: 0.613545
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568


RESULTS FOR OPEN VALUES

ADF Statistic: -1.247650
p-value: 0.652770
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568


RESULTS FOR CLOSE VALUES

ADF Statistic: -1.259750
p-value: 0.647380
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

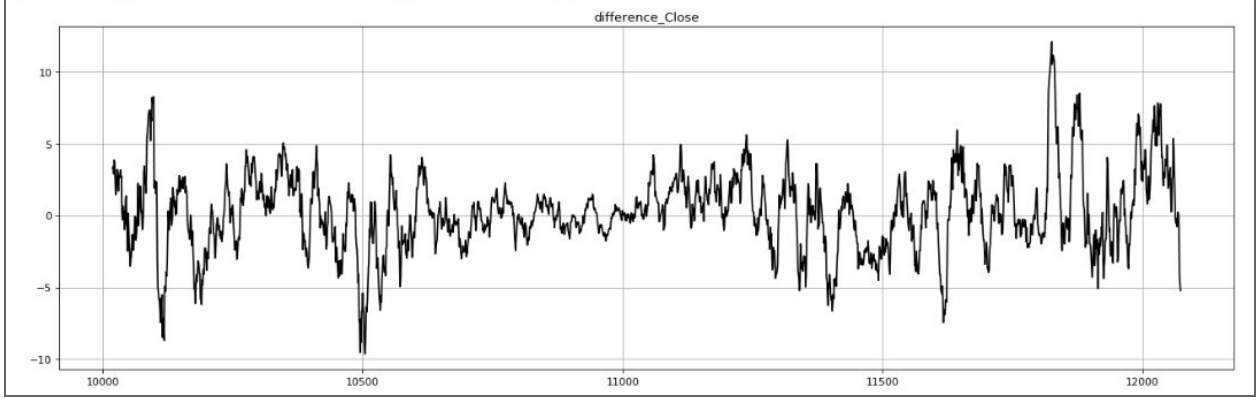
It is clear from the above result, that the above data is not stationary, even under a 90% confidence interval. So, As the mean of the data is varying, differencing can be tried to check whether the dataset becomes stationary or not.

So, the following are the four time series after performing differencing of order 1.

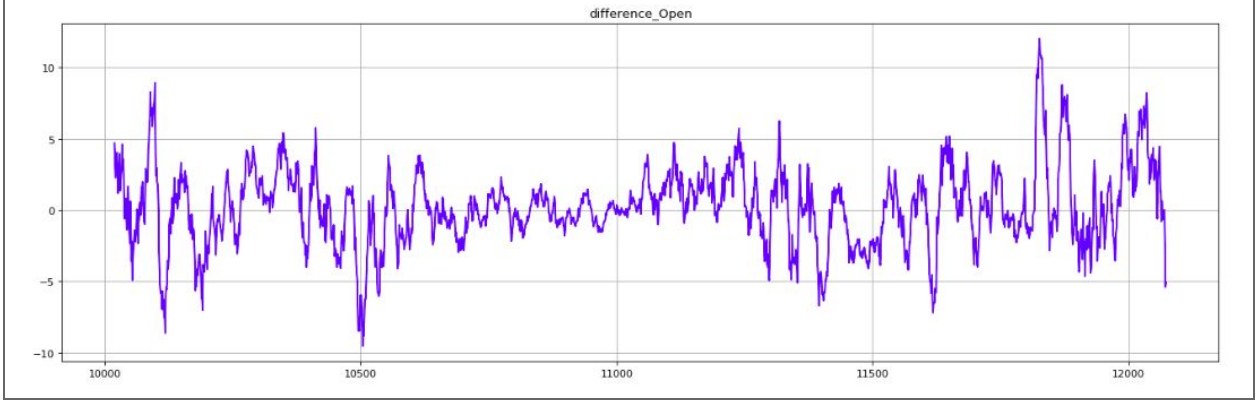
```
plot_data(data[10000:], 'difference_Low','red')
```



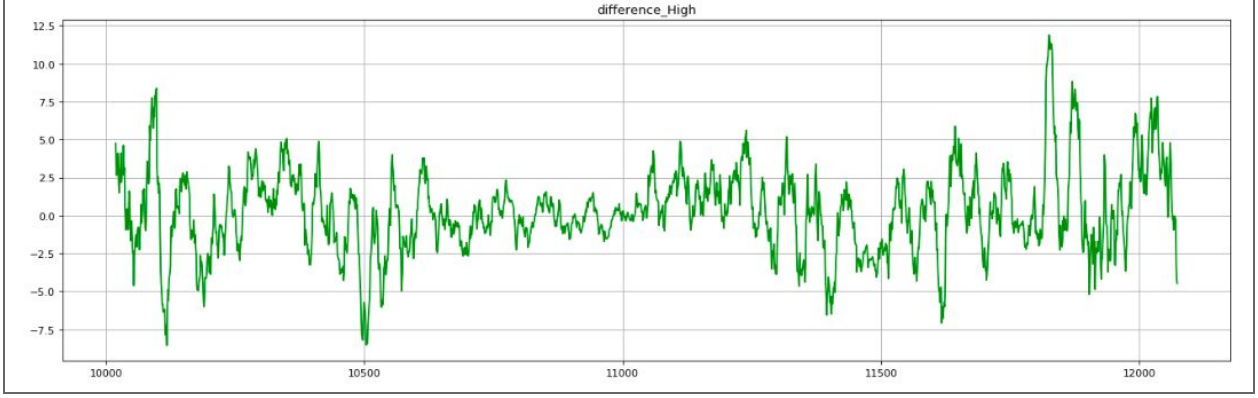

```
plot_data(data[10000:], 'difference_Close','black')
```




```
plot_data(data[10000:], 'difference_Open','blue')
```




```
plot_data(data[10000:], 'difference_High','green')
```



On applying augmented dickey-fuller test on the above datasets, each of them turned out to be stationary.

```
AUGMENTED DICKY FULLER TEST FOR STATIONARITY

RESULTS FOR difference_High VALUES

ADF Statistic: -40.480416
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568


RESULTS FOR difference_LOW VALUES

ADF Statistic: -24.739959
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568


RESULTS FOR difference_OPEN VALUES

ADF Statistic: -45.040802
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568


RESULTS FOR difference_CLOSE VALUES

ADF Statistic: -45.195568
p-value: 0.000000
Critical Values:
        1%: -3.434
        5%: -2.863
        10%: -2.568
```

It is clear that p-value for each of the test strongly suggests that each of the above considered time series are stationary in nature.

STEP 4: AUTOCOVARIANCE MATRIX

Building the autocovariance matrix, using the Pearson coefficient, between the lagged values of the same time series. After building the autocovariance matrix, it is used to find out the exact solution for the least squared loss function, and beta values are then used for predicting the future values in the test dataset.

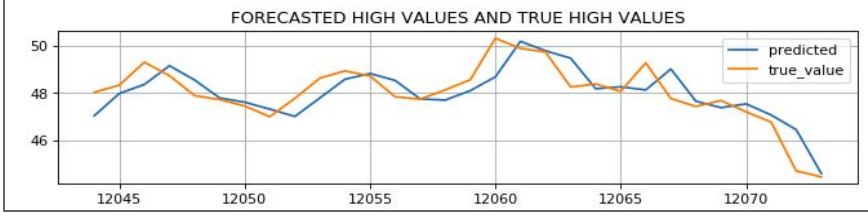
```
data_High = data_prep(data, 'difference_High')
MAX_LAG = 10
LENGTH_GENOME = max(data_High.index)
RES = []
for i in range(MAX_LAG+1):
    p = np.round(get_corr(i, data_High),3)
    RES.append(p)
lags = []
cors = []
for i in range(0, len(RES)):
    lags.append(RES[i][0])
    cors.append(RES[i][1])
        #print(cors)

cor_mat = ACF_mat(cors)
cor_mat = np.array(cor_mat)
print(cor_mat)
```

```
[[ 1.     0.118 -0.003 -0.005  0.008  0.007  0.016 -0.01  -0.021  0.035]
 [ 0.118  1.     0.118 -0.003 -0.005  0.008  0.007  0.016 -0.01  -0.021]
 [-0.003  0.118  1.     0.118 -0.003 -0.005  0.008  0.007  0.016 -0.01 ]
 [-0.005 -0.003  0.118  1.     0.118 -0.003 -0.005  0.008  0.007  0.016]
 [ 0.008 -0.005 -0.003  0.118  1.     0.118 -0.003 -0.005  0.008  0.007]
 [ 0.007  0.008 -0.005 -0.003  0.118  1.     0.118 -0.003 -0.005  0.008]
 [ 0.016  0.007  0.008 -0.005 -0.003  0.118  1.     0.118 -0.003 -0.005]
 [-0.01   0.016  0.007  0.008 -0.005 -0.003  0.118  1.     0.118 -0.003]
 [-0.021 -0.01   0.016  0.007  0.008 -0.005 -0.003  0.118  1.     0.118]
 [ 0.035 -0.021 -0.01   0.016  0.007  0.008 -0.005 -0.003  0.118  1.   ]]
```

STEP 5: MODEL PARAMETERS DETERMINATION AND RMSE CALCULATION

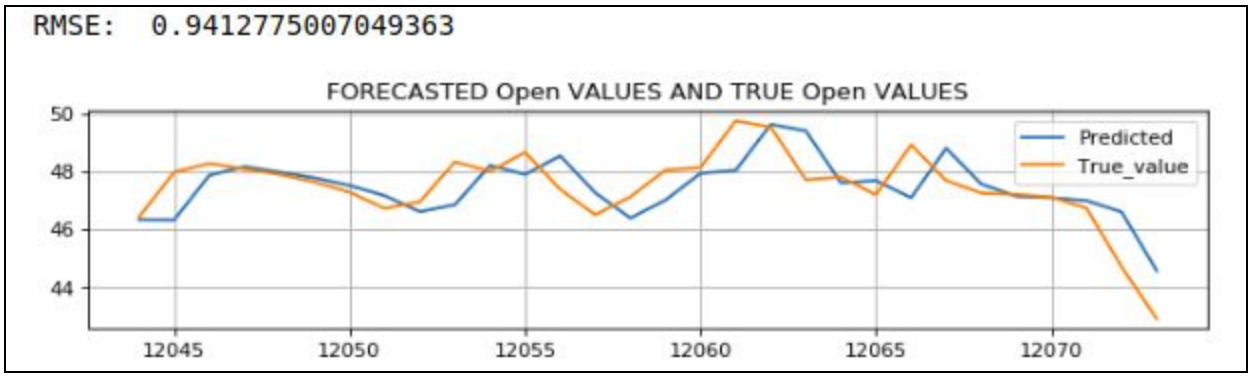
```
#### TRAINING MODEL FOR A PARTICULAR LAG VALUE AND DATASET
data_High = data_prep(data, 'difference_High')
MAX_LAG = 1
LENGTH_GENOME = max(data_High.index)
beta_High = AR_Model(MAX_LAG, data_High)
print(beta_High)

## PREDICTING DIFFERENCED VALUES USING BETA
true_values = data.copy()[(len(data)-30)-MAX_LAG:]['difference_High'].values
predictions = predict(beta_High, true_values)

## PREDICTING ACTUAL VALUES
true_high_values = list(data.copy()[(len(data)-30)-MAX_LAG:].High)
predict_HIGH = []
for i in range(0,len(predictions)):
    predicted_value = round(true_high_values[i+MAX_LAG-1] + predictions[i],2)
    predict_HIGH.append(predicted_value)
# print((predict_HIGH))
# print(true_high_values)print("RMSE: ", RMSE(predict_High, true_High_values[MAX_LAG:]))
print("RMSE: ", RMSE(predict_HIGH, true_high_values[MAX_LAG:]))

## PLOTTING VALUES PREDICTED V/S ACUTAL
import matplotlib.pyplot as plt
plt.figure(num=None, figsize=(10, 2), dpi=80, facecolor='w', edgecolor='k')
plt.plot(data.index[len(data)-30:],predict_HIGH,data.index[len(data)-30:],true_high_values[MAX_LAG:])
plt.title('FORECASTED HIGH VALUES AND TRUE HIGH VALUES')
plt.legend(['predicted','true_value'])
plt.grid(True)
plt.show()
```

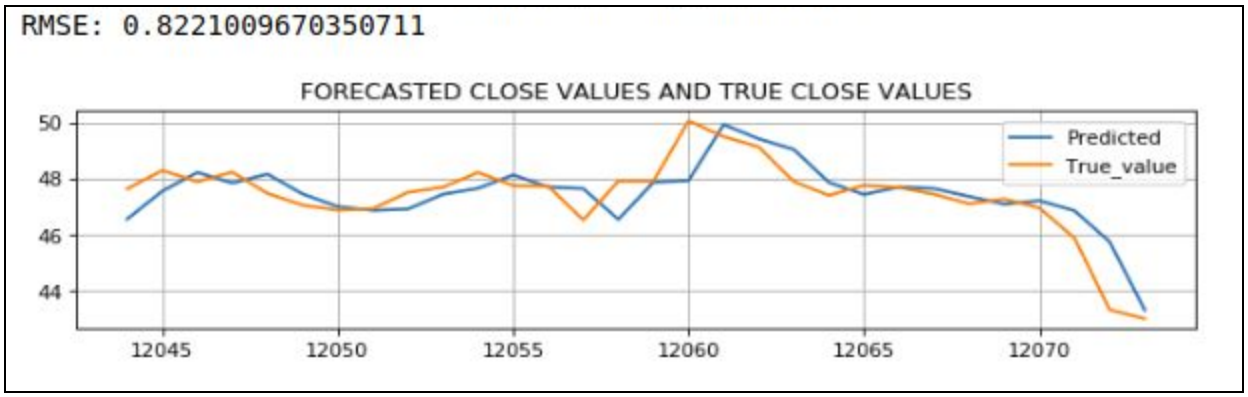
```
[0.11779269]
RMSE:  0.7203388684038461
```



Model parameters are determined using the autocovariance matrix, and autocorrelation values between lagged values of the same time series. The model parameters are then used to predict the values, and then RMSE is calculated using the standard formula.

The above procedure is followed for each of the four-time series and different models are built for them. The following are screenshots of the graphs of predicted v/s true values.





RESULTS AND CONCLUSION

COVARIANCE MATRIX FOR p = 10

For differenced_High Values:

```
[[ 1.     0.118 -0.003 -0.005  0.008  0.007  0.016 -0.01  -0.021  0.035]
 [ 0.118  1.     0.118 -0.003 -0.005  0.008  0.007  0.016 -0.01  -0.021 ]
 [-0.003  0.118  1.     0.118 -0.003 -0.005  0.008  0.007  0.016 -0.01  ]
 [-0.005 -0.003  0.118  1.     0.118 -0.003 -0.005  0.008  0.007  0.016]
 [ 0.008 -0.005 -0.003  0.118  1.     0.118 -0.003 -0.005  0.008  0.007]
 [ 0.007  0.008 -0.005 -0.003  0.118  1.     0.118 -0.003 -0.005  0.008]
 [ 0.016  0.007  0.008 -0.005 -0.003  0.118  1.     0.118 -0.003 -0.005]
 [-0.01   0.016  0.007  0.008 -0.005 -0.003  0.118  1.     0.118 -0.003 ]
 [-0.021 -0.01   0.016  0.007  0.008 -0.005 -0.003  0.118  1.     0.118 ]
 [ 0.035 -0.021 -0.01   0.016  0.007  0.008 -0.005 -0.003  0.118  1.   ]]
```

For differenced_Low Values:

```
[[ 1.     0.14  -0.029  0.03  -0.006  0.015  0.006 -0.016 -0.036 -0.003]
 [ 0.14   1.     0.14  -0.029  0.03  -0.006  0.015  0.006 -0.016 -0.036  ]
 [-0.029  0.14   1.     0.14  -0.029  0.03  -0.006  0.015  0.006 -0.016  ]
 [ 0.03  -0.029  0.14   1.     0.14  -0.029  0.03  -0.006  0.015  0.006  ]
 [-0.006  0.03  -0.029  0.14   1.     0.14  -0.029  0.03  -0.006  0.015  ]
 [ 0.015 -0.006  0.03  -0.029  0.14   1.     0.14  -0.029  0.03  -0.006  ]
 [ 0.006  0.015 -0.006  0.03  -0.029  0.14   1.     0.14  -0.029  0.03   ]
 [-0.016  0.006  0.015 -0.006  0.03  -0.029  0.14   1.     0.14  -0.029  ]
 [-0.036 -0.016  0.006  0.015 -0.006  0.03  -0.029  0.14   1.     0.14   ]
 [-0.003 -0.036 -0.016  0.006  0.015 -0.006  0.03  -0.029  0.14   1.   ]]
```

For differenced_Open Values:

```
[[ 1.     0.008  0.007 -0.008 -0.004  0.013  0.025 -0.036  0.     0.017  ]
 [ 0.008  1.     0.008  0.007 -0.008 -0.004  0.013  0.025 -0.036  0.     ]
 [ 0.007  0.008  1.     0.008  0.007 -0.008 -0.004  0.013  0.025 -0.036]
 [-0.008  0.007  0.008  1.     0.008  0.007 -0.008 -0.004  0.013  0.025]
 [-0.004 -0.008  0.007  0.008  1.     0.008  0.007 -0.008 -0.004  0.013]
 [ 0.013 -0.004 -0.008  0.007  0.008  1.     0.008  0.007 -0.008 -0.004]
 [ 0.025  0.013 -0.004 -0.008  0.007  0.008  1.     0.008  0.007 -0.008]
 [-0.036  0.025  0.013 -0.004 -0.008  0.007  0.008  1.     0.008  0.007]
 [ 0.    -0.036  0.025  0.013 -0.004 -0.008  0.007  0.008  1.     0.008  ]
 [ 0.017  0.    -0.036  0.025  0.013 -0.004 -0.008  0.007  0.008  1.     ]
```

For difference_Close Values:

```
[ 1.     0.004  0.003  0.012 -0.013  0.015  0.018 -0.021 -0.014 -0.011]
 [ 0.004  1.     0.004  0.003  0.012 -0.013  0.015  0.018 -0.021 -0.014]
 [ 0.003  0.004  1.     0.004  0.003  0.012 -0.013  0.015  0.018 -0.021]
 [ 0.012  0.003  0.004  1.     0.004  0.003  0.012 -0.013  0.015  0.018]
 [-0.013  0.012  0.003  0.004  1.     0.004  0.003  0.012 -0.013  0.015]
 [ 0.015 -0.013  0.012  0.003  0.004  1.     0.004  0.003  0.012 -0.013]
 [ 0.018  0.015 -0.013  0.012  0.003  0.004  1.     0.004  0.003  0.012]
 [-0.021  0.018  0.015 -0.013  0.012  0.003  0.004  1.     0.004  0.003]
 [-0.014 -0.021  0.018  0.015 -0.013  0.012  0.003  0.004  1.     0.004]
 [-0.011 -0.014 -0.021  0.018  0.015 -0.013  0.012  0.003  0.004  1.   ]
```

Hence we conclude based on the covariance matrix obtained that an autoregressive model with lag (1) would be apt for predicting the required variables. Also, from the graphs of the various variables vs time, we conclude that they follow close to identical variations. The data was detected to be non-stationary initially from the Dickey-Fuller test, which was corrected and the stationary data was used for the purpose of analysis and predictions.

PREDICTIONS FOR THE MONTH OF OCT-NOV, 2017

For differenced High_Values:

[47.05, 48.01, 48.45, 49.26, 48.65, 47.85, 47.73, 47.36, 46.93, 47.79, 48.75, 49.19, 48.67, 47.67, 47.68, 48.03, 48.55, 50.47, 49.92, 49.88, 48.25, 48.37, 47.96, 49.11, 47.79, 47.46, 47.6, 47.26, 46.79, 44.72]

For differenced Low_Values:

[45.87, 46.45, 47.38, 47.63, 47.5, 47.1, 46.48, 46.43, 46.19, 46.81, 47.48, 47.37, 47.19, 46.34, 46.06, 47.07, 47.76, 48.02, 48.41, 48.21, 46.96, 47.13, 46.43, 47.3, 46.92, 46.58, 46.43, 46.27, 44.95, 42.96]

For differenced Open_Values:

[46.34, 46.33, 47.86, 48.16, 47.97, 47.77, 47.51, 47.16, 46.61, 46.85, 48.2, 47.89, 48.53, 47.26, 46.39, 47.01, 47.93, 48.03, 49.61, 49.39, 47.6, 47.68, 47.09, 48.79, 47.56, 47.14, 47.09, 46.99, 46.6, 44.58]

For differenced Close_Values:

[46.57, 47.57, 48.24, 47.86, 48.18, 47.47, 47.02, 46.88, 46.93, 47.46, 47.67, 48.15, 47.72, 47.66, 46.56, 47.89, 47.95, 49.95, 49.44, 49.06, 47.88, 47.45, 47.72, 47.66, 47.38, 47.11, 47.23, 46.87, 45.76, 43.32]

References:

[1] www.sciencedirect.com
[2] Faculty.washington.edu
[3] www.stat.berkeley.edu
[4] Nwfsc-timeseries.github.io
[5] Machinelearningmastery.com
[6] en.wikipedia.org
[6] https://cse.iitkgp.ac.in/~dsamanta/courses/da/index.html
[7] www.statisticssolutions.com
[8] Medium.com
[9] towardsdatascience.com
[10] pandas.pydata.org