# Import libraries

```
import pandas as pd
import numpy as np
import math
import matplotlib.pyplot as plt1
import matplotlib.pyplot as plt2
import matplotlib.pyplot as plt3
import matplotlib.pyplot as plt4
import matplotlib.pyplot as plt5
import matplotlib.pyplot as plt9
```

# Utility Functions

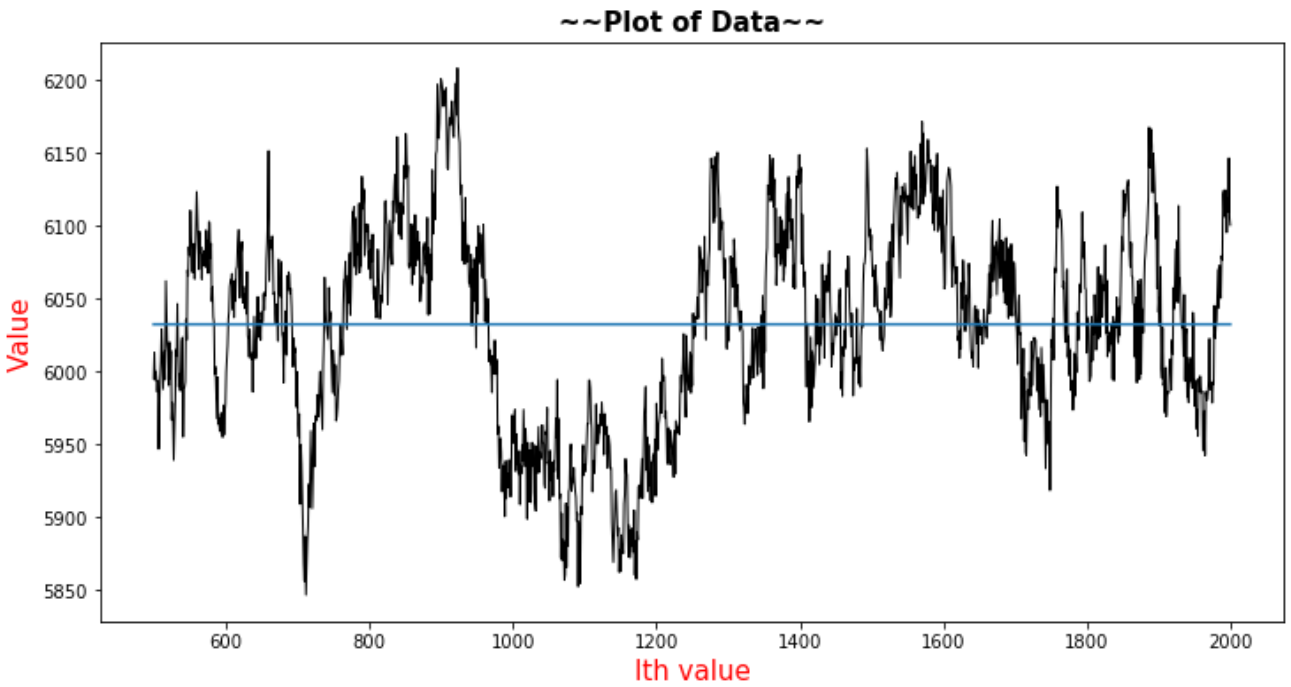
```
def convert(X):
    temp_list = []
    for a in X:
        temp_list.append(float(a))
    return temp_list

def exponential_moving_average(diff,alpha):
    predictedList = []
    predictedList.append(diff[0][0])
    for x in range(1,1499):
        predictedList.append(alpha*diff[x-1][0]+(1-alpha)*predictedList[x-1])
    return predictedList
def exponent_move_avg(diff,alpha):
    predictedList = []
    predictedList.append(diff[0][0])
    for x in range(1,500):
        predictedList.append(alpha*diff[x-1][0]+(1-alpha)*predictedList[x-1])
    return predictedList
```

# TASK 1: SIMPLE MOVING AVERAGE MODEL

# Calculating RMSE values for different values of M

```
rmseList = []
data = pd.read_csv("C:/Users/princ/Anaconda3/data.csv",names=['X'],header=None)
data = convert(data.X[1:].tolist())
finalData = pd.DataFrame(np.column_stack([data]))
finalData.columns = ['data']
finalDataList = finalData['data'][500:2000].values.tolist()
diff = finalData[150:1650].diff(periods=1)
diff = diff.dropna()
mean = np.sum(finalDataList)/len(finalDataList)
listofmean = [mean] * len(data[500:2000])
f, ax0 = plt9.subplots(figsize=(12,6))
ax0.plot(range(500,2000),data[500:2000],color='black',linewidth=1.0)
ax0.plot(range(500,2000),listofmean)
ax0.set_title('~~Plot of Data~~',fontweight='bold',fontsize=15)
ax0.set_xlabel('Ith value',fontsize=15,color='red')
ax0.set_ylabel('Value',fontsize=15,color='red')
plt9.show()
```

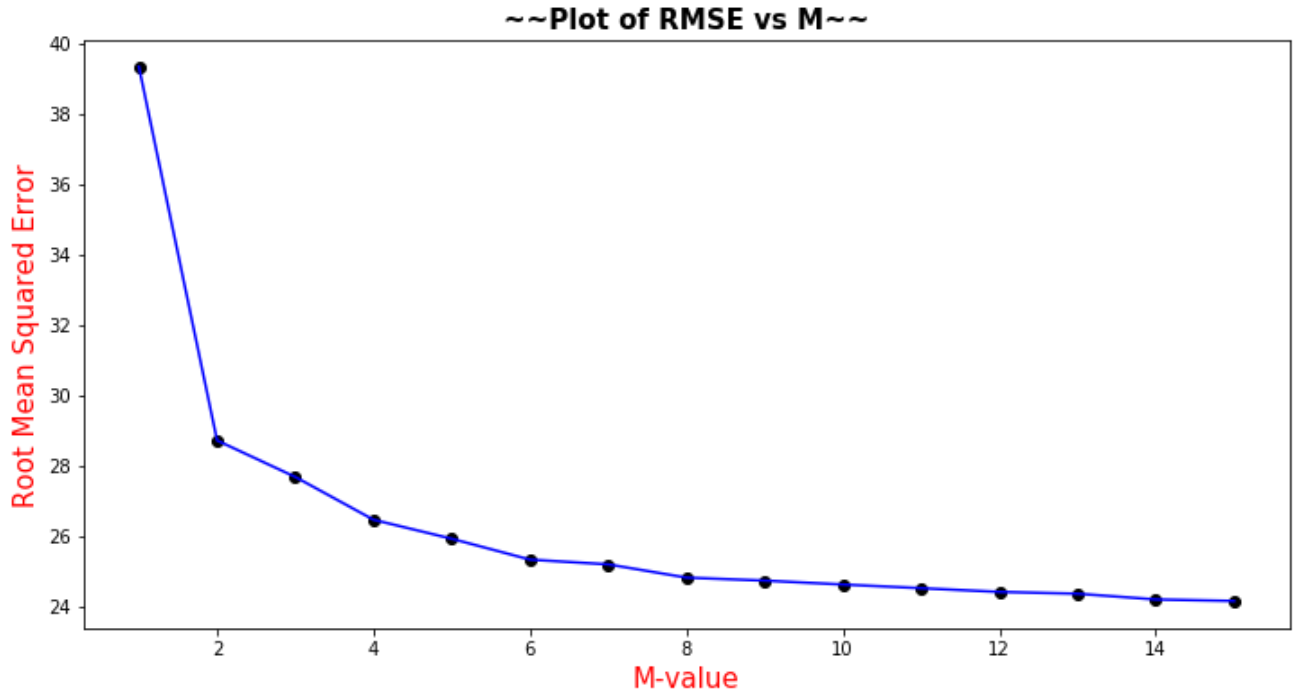



There is trend in the data which can be seen in the plot just shown above. The trend in the data causes the error to increase as the trend progresses further for different values of m for moving average model. To remove the trend, we have to apply differencing on data. Differencing removes the trend in data which will be used to decide the value of M

```
for x in range(0,15):
    error = []
    predicted = diff.rolling(window=x+1).mean()
    predicted.columns = ['pred']
    predicted=predicted.pred.shift(1)
    diffList = diff[x+1:1500].values.tolist()
    predictedList = predicted[x+1:1500].values.tolist()
    result = pd.DataFrame(np.column_stack([diffList,predictedList]))
    #print(result)
    sum = 0
    for y in range(0,1500-x-2):
        sum = sum + (diffList[y][0]-predictedList[y])**2
        error.append(diffList[y][0]-predictedList[y])
    rmse = math.sqrt(sum/len(predictedList))
    print(rmse)
    rmseList.append(rmse)

f, ax1 = plt1.subplots(figsize=(12,6))
ax1.scatter(range(1,16),rmseList,  color='black')
ax1.plot(range(1,16),rmseList, color='blue')
ax1.set_title('~~Plot of RMSE vs M~~',fontweight='bold',fontsize=15)
ax1.set_xlabel('M-value',fontsize=15,color='red')
ax1.set_ylabel('Root Mean Squared Error',fontsize=15,color='red')
plt1.show()
```

```
39.34368682287498
28.712495073878312
27.67501492446164
26.458247778914902
25.918024091318124
25.324707717956528
25.190291976392974
24.813414468185247
24.728206492084073
24.61744325288255
24.512721042992055
24.407362322512977
24.355894334832588
24.1926249599146
24.14965775693632
```

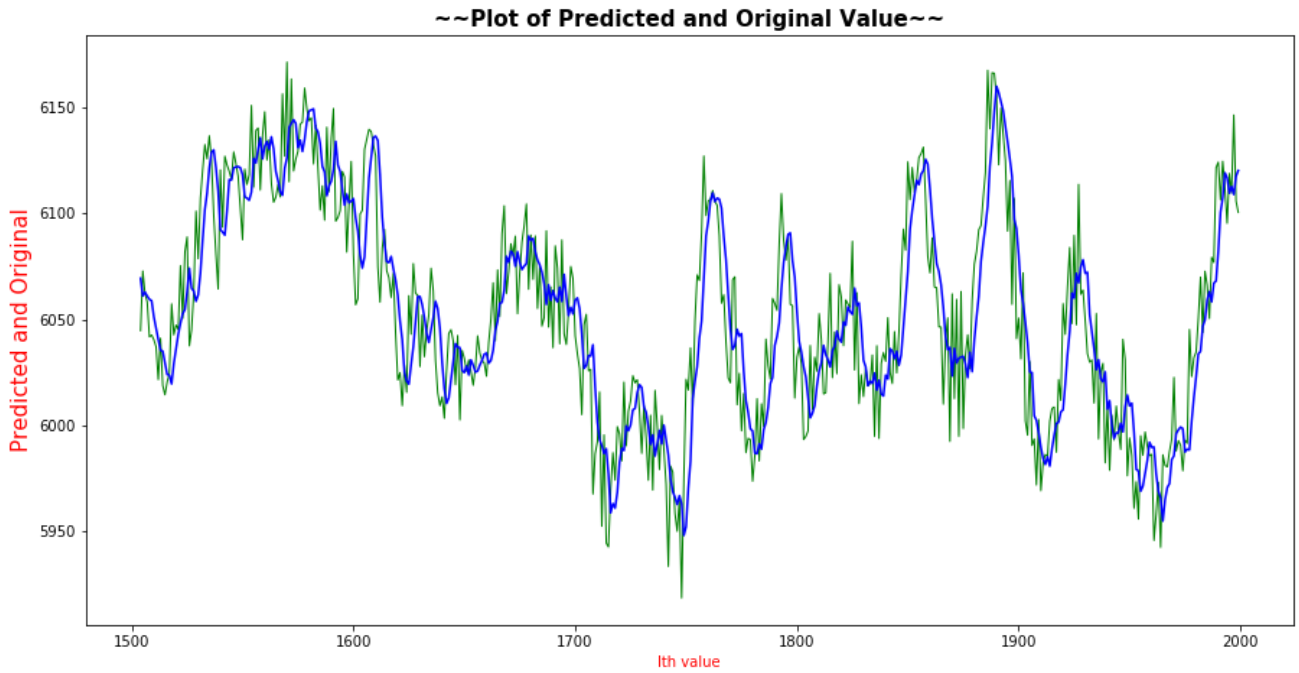



# Selection of 'm' value based on rmse value

From the plot bewteen RMSE values and M, we can select M=4. The reason for choosing this value is that after that value there is less change in RMSE values. We dont get much gain in reducing the error after M value equal to 4. So, for predicting the values for test data, M value equal to 4 is used.

# Plotting graph between predicted and original values for best value of M=4

```
diffBest = finalData['data'][1500:2000]
predictedBest = diffBest.rolling(window=4).mean()
predictedBest = predictedBest.shift(1)
resultBest = pd.DataFrame(np.column_stack([diffBest[4:500],predictedBest[4:500]]),columns=
['diffBest','predicted'])
f, ax2 = plt4.subplots(figsize=(15,7.5))
ax2.plot(range(1504,2000),diffBest[4:500],color='green',linewidth=1.0)
ax2.plot(range(1504,2000),predictedBest[4:500],color='blue')
ax2.set_title('~~Plot of Predicted and Original Value~~',fontweight='bold',fontsize=15)
ax2.set_xlabel('Ith value',color='red')
ax2.set_ylabel('Predicted and Original',fontsize=15,color='red')
plt4.show()
diffBestList = resultBest['diffBest'].values.tolist();
predictedBestList = resultBest['predicted'].values.tolist()
sum=0
for y in range(0,496):
    sum = sum + (diffBestList[y]-predictedBestList[y])**2
rmseMA = math.sqrt(sum/len(predictedBestList))
```



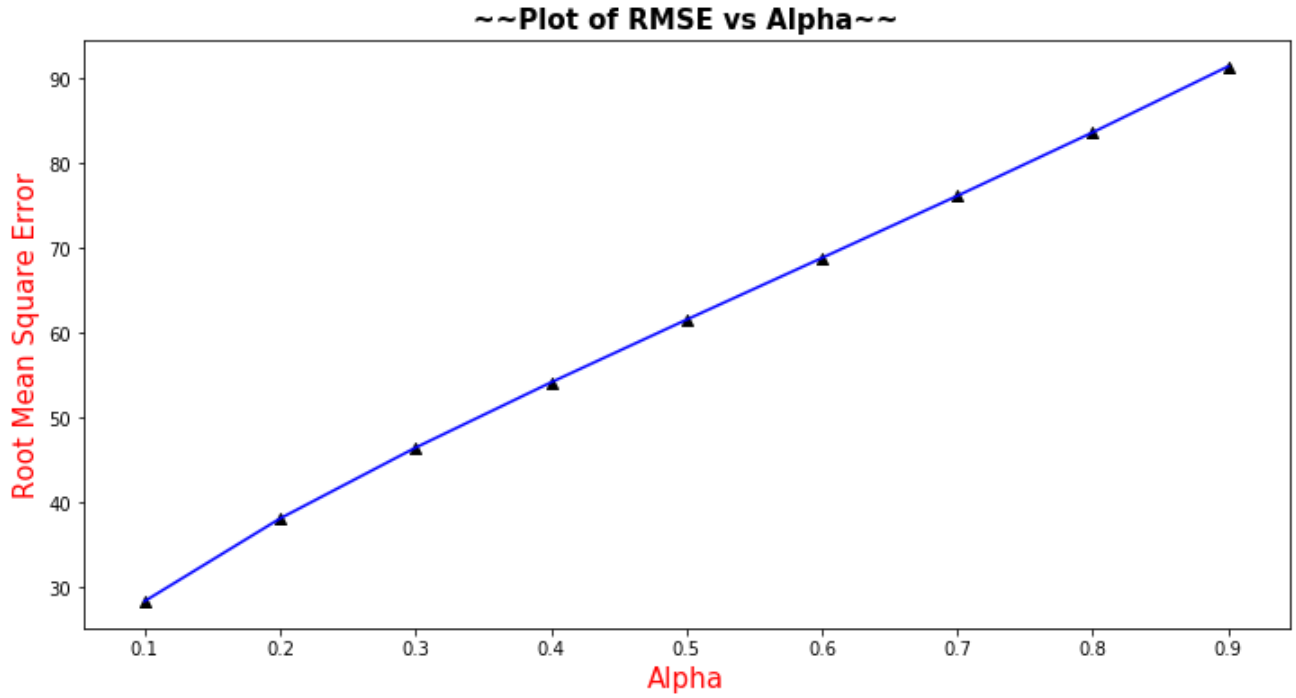

# Inferences from the plot

From the above plot, we can say that the moving average model follows the original data for window equal to 4. The predicted data follows the trend of the original data but the error is produced when successive data in the original values falls and grows very rapidly which will produce error in the predicted values. The reason for the error is the model cannot accurately follow that kind of falls and growth in original data because the average of values will lie between the maximum and minimum values which will be in between two values.

# TASK2 : EXPONENTIAL SMOOTHING MODEL

```
rmseList = []
arr = np.arange(0.1, 1.0, 0.1)
diff = finalData[0:1500].diff(periods=1)
diff = diff.dropna()
diffExp = diff.values.tolist()
for x in arr:
    error = []
    predicted = exponential_moving_average(diffExp,x)
    for y in range(0,1499):
        sum = sum + (diffExp[y][0]-predicted[y])**2
        error.append(diffExp[y][0]-predicted[y])
    resultExp = pd.DataFrame(np.column_stack([diffExp,predicted,error]),columns=
['Original','Predicted','error'])
    rmse = math.sqrt(sum/len(diffExp[0:1499]))
    print('RMSE for Alpha equal to ',(x),': ',rmse)
    rmseList.append(rmse)
f, ax4 = plt3.subplots(figsize=(12,6))
ax4.scatter(arr,rmseList,  color='black', marker='^')
ax4.plot(arr,rmseList, color='blue')
ax4.set_title('~~Plot of RMSE vs Alpha~~', fontweight='bold',fontsize=15)
ax4.set_xlabel('Alpha',fontsize=15,color='red')
ax4.set_ylabel('Root Mean Square Error',fontsize=15,color='red')
plt3.show()
```

```
RMSE for Alpha equal to  0.1 :  28.340281168520054
RMSE for Alpha equal to  0.2 :  38.093610087010134
RMSE for Alpha equal to  0.3 :  46.44464285818556
RMSE for Alpha equal to  0.4 :  54.144901481103446
RMSE for Alpha equal to  0.5 :  61.535139298629055
RMSE for Alpha equal to  0.6 :  68.81910455939416
RMSE for Alpha equal to  0.7 :  76.14459318064014
RMSE for Alpha equal to  0.8 :  83.63693088520972
RMSE for Alpha equal to  0.9 :  91.41700042145851
```

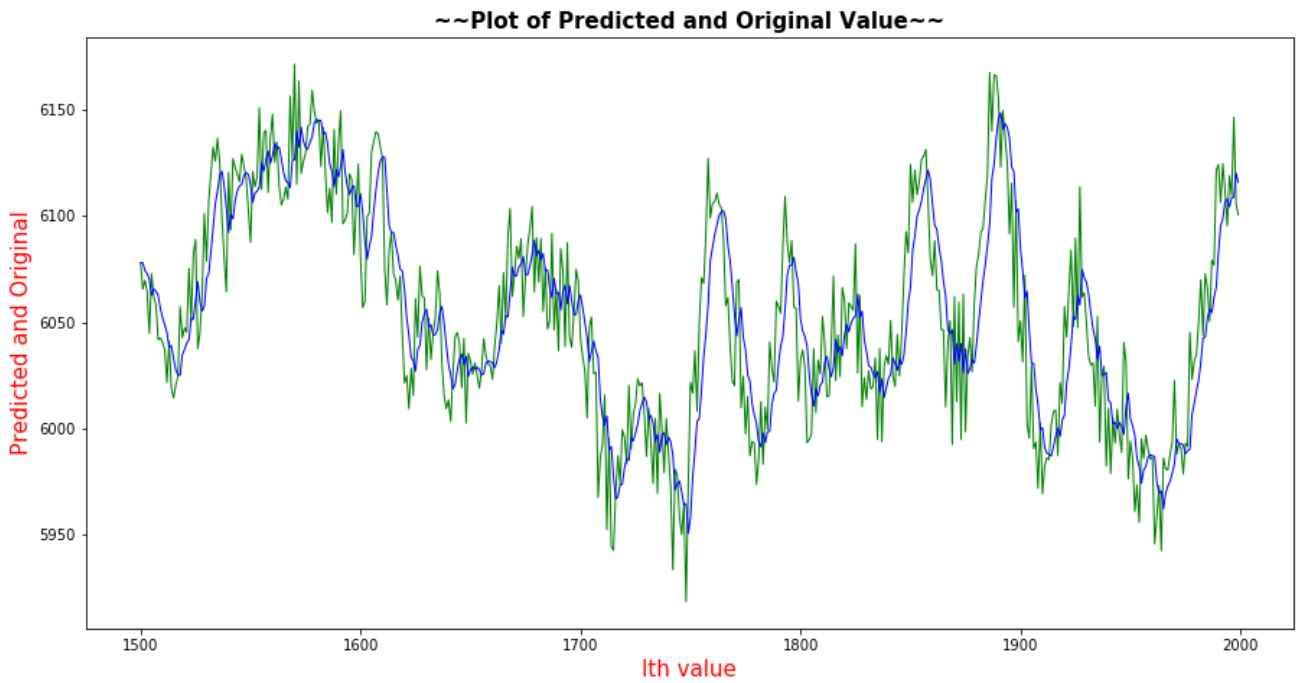



# Selection of Alpha value based on RMSE value

From the graph plotted between different values of alpha and corresponding rmse values, we can see that the lowest value of alpha for which Root Mean Squared Error is low is 0.3. The Exponential moving average model function made by me uses first value of prediction as the original value. It then progresses the prediction by using the formula predicted(t) = alpha*original(t-1) + (1-alpha)*predicted(t-1). The predicted values produces RMSE value lowest for 0.1 but I choose 0.3 because if we choose 0.1 then it will lay more error because the prediction of the value at instance t will be more dependent on prediction made for instance (t-1).

# Plotting graph between predicted and original value for best value of alpha=0.3

```
finalList = finalData[1500:2000].values.tolist()
predictedExpBest = exponent_move_avg(finalList,0.3)
resultExpBest = pd.DataFrame(np.column_stack([finalList,predictedExpBest]),columns=
['Original','Predicted'])
f, ax5 = plt5.subplots(figsize=(15,7.5))
ax5.plot(range(1500,2000),resultExpBest['Original'],color='green',linewidth=1.0)
ax5.plot(range(1500,2000),resultExpBest['Predicted'],color='blue',linewidth=1.0)
ax5.set_title('~~Plot of Predicted and Original Value~~',fontweight='bold',fontsize=15)
ax5.set_xlabel('Ith value',color='red',fontsize=15)
ax5.set_ylabel('Predicted and Original',fontsize=15,color='red')
plt5.show()
sum=0
originalList = resultExpBest['Original'].values.tolist();
predictedList = resultExpBest['Predicted'].values.tolist();
for y in range(0,500):
    sum = sum + (originalList[y]-predictedList[y])**2
rmseExp = math.sqrt(sum/len(predictedList))
```

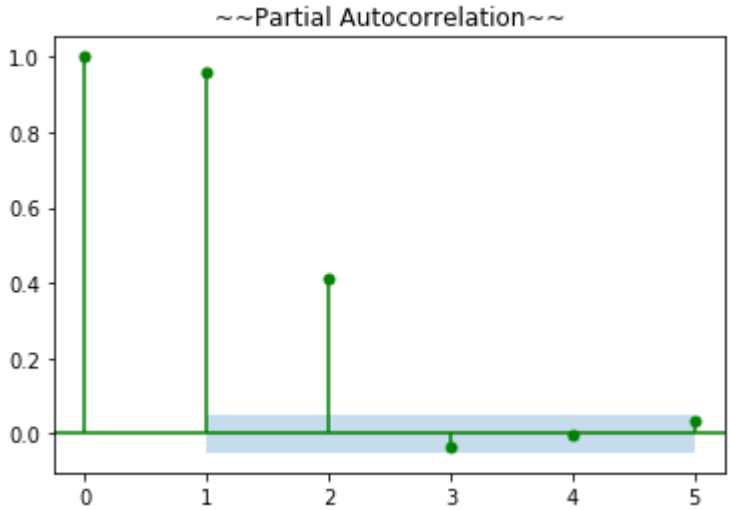



# Inferences from plot

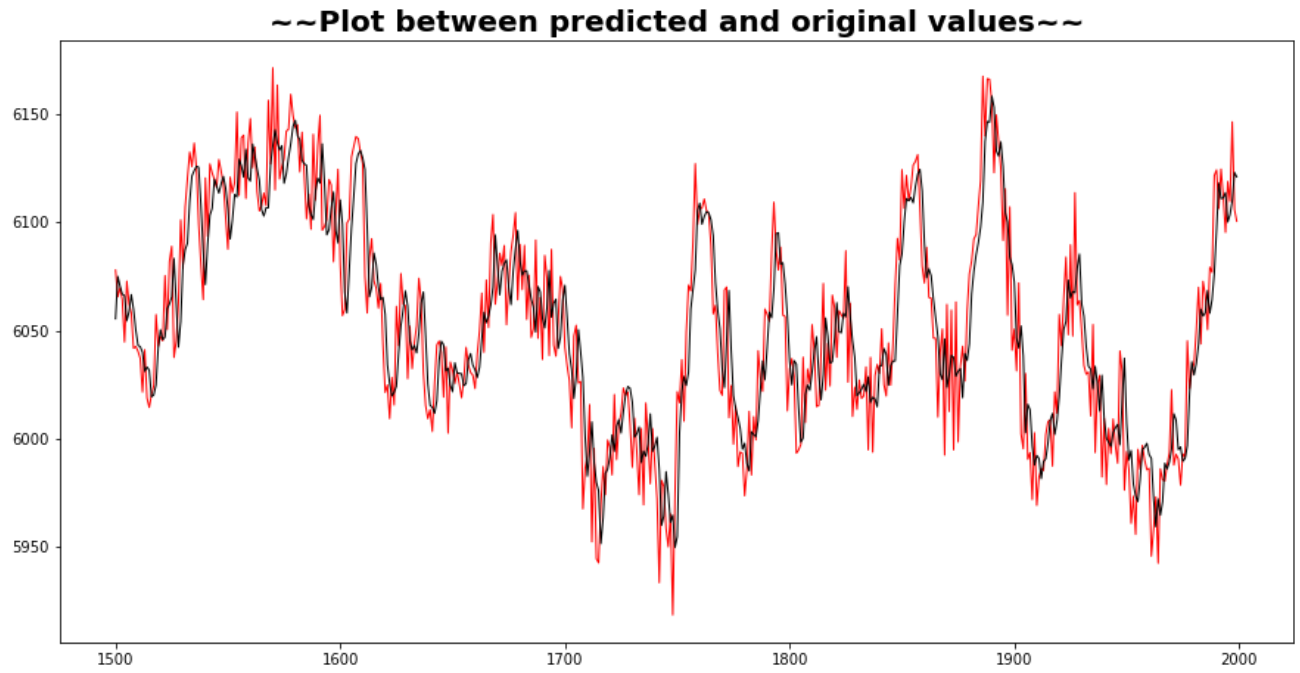
From the plot above between Predicted and original values, we can see that the error imposed is more. The plot shows that some of the values on spikes is not predicted accurately because the model doesnt follow the trend of the original values. The model doesn't predict values accurately near the spikes in the original values. But compared to Mean Average Model, the spike values are predicted accurately which can be seen from the plot.

```
import matplotlib.pyplot as plt7
import matplotlib.pyplot as plt8
from statsmodels.tsa.arima_model import ARIMA
from statsmodels.graphics.tsaplots import plot_pacf
final = finalData[1500:2000].values.tolist();
plot_pacf(finalData[150:1650],lags=5,method='ols',color='green',title='~~Partial
Autocorrelation~~')
```

```
C:\Users\princ\Anaconda3\lib\site-packages\statsmodels\compat\pandas.py:56: FutureWarning: The
pandas.core.datetools module is deprecated and will be removed in a future version. Please use
the pandas.tseries module instead.
  from pandas.core import datetools
```




```
def AR(p):
    arma_mod20 = ARIMA(final, order=(p,0,0)).fit();
    f, ax = plt7.subplots(figsize=(15,7.5))
    ax.plot(range(1500,2000),final,color='red',linewidth=1)
    ax.plot(range(1500,2000),arma_mod20.fittedvalues,color='black',linewidth=1)
    ax.set_title('~~Plot between predicted and original
values~~',fontweight='bold',fontsize='20')
    plt7.show()
    sum = 0
    predictedList = arma_mod20.fittedvalues
    for y in range(0,500):
        sum = sum + (final[y]-predictedList[y])**2
    rmseAR = math.sqrt(sum/len(predictedList))
    return rmseAR
rmseAR = AR(2)
```




# Inferences from the plot

From the plot, we can see that the predicted values follow the trend of the original data near the spikes and other positions also. Also, compared to the previous models, the spike values are predicted more accurately which is reducing the error of the overall model.

# Conclusion:

```
from tabulate import tabulate
print (tabulate([['Moving Average Model', rmseMA], ['Exponential Moving Average', rmseExp],
['Auto Regressive Model', rmseAR]], headers=['Model', 'Error'], tablefmt='orgtbl'))
```

```
| Model                      |   Error |
|----------------------------+---------|
| Moving Average Model       | 24.488  |
| Exponential Moving Average | 25.0442 |
| Auto Regressive Model      | 22.2451 |
```

From the above table, we can choose the Auto Regressive Model as the RMSE value is lower for that model

```
```