Uber Dataset Analysis using Time series Forecasting

Bishakha Bikhya (180710007013)
Himashree Hazarika (180710007023)
Jhumpa Sarma(180710007026)

INTRODUCTION

A time series is defined as a set of random variables ordered with respect to time. Time series are studied both to interpret a phenomenon, identifying the components of a trend, cyclicity, seasonality and to predict its future values.

A normal machine learning dataset is a collection of observations.Time does play a role in normal machine learning datasets.Predictions are made for new data when the actual outcome may not be known until some future date.

VARIOUS TIME SERIES ANALYSIS TECHNIQUES-

1) Trend analysis to determine whether it is linear or not

2) Outliers detection to understand how to spot and handle them

3) Stationarity test to understand if we can assume that the time series is stationary or not

4) Seasonality analysis to determine what is the best seasonal parameter to use when modeling.

OBJECTIVE OF THE PROJECT

1.Since the number of Uber trips per day in NYC is still growing significantly.Our idea is to identify the nature of the phenomenon represented by the sequence of observations and perform a thorough analysis of the data .

2.The objective of our project is to perform a Time series forecasting using various methods and compare the best model out.

3.To find the accuracy of the models for better model selection

4.To compare RMSE values of the selected model.

VARIOUS MODELS IMPLEMENTED IN OUR PROJECT

1) Holt's Winter Seasonal

2) ARIMA (AutoRegressive Integrated Moving Average)

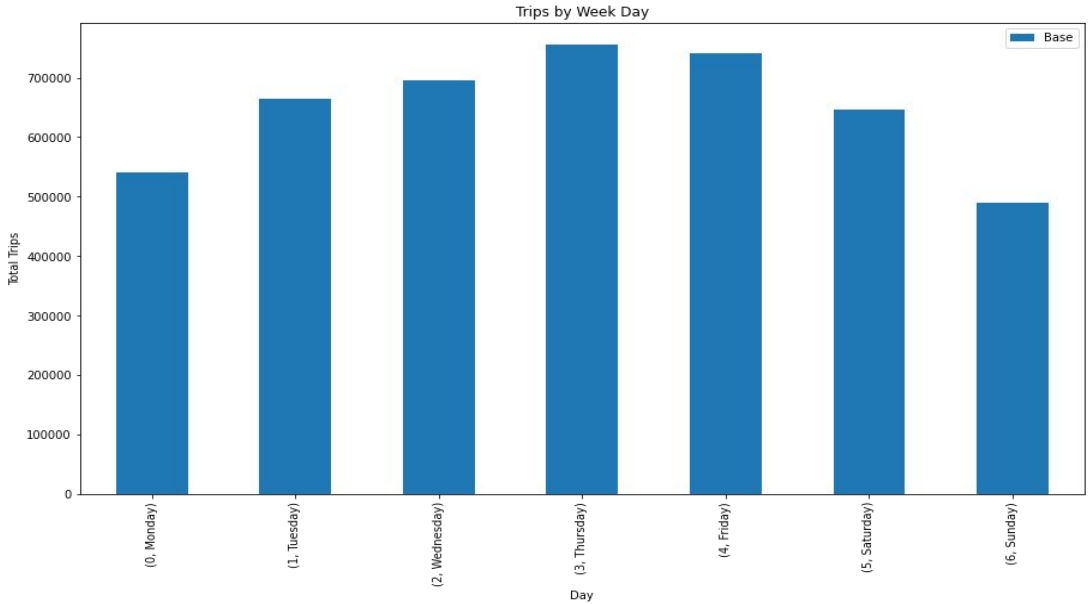
3) ANN (Artificial Neural Network)

EDA (Exploratory Data Analysis)

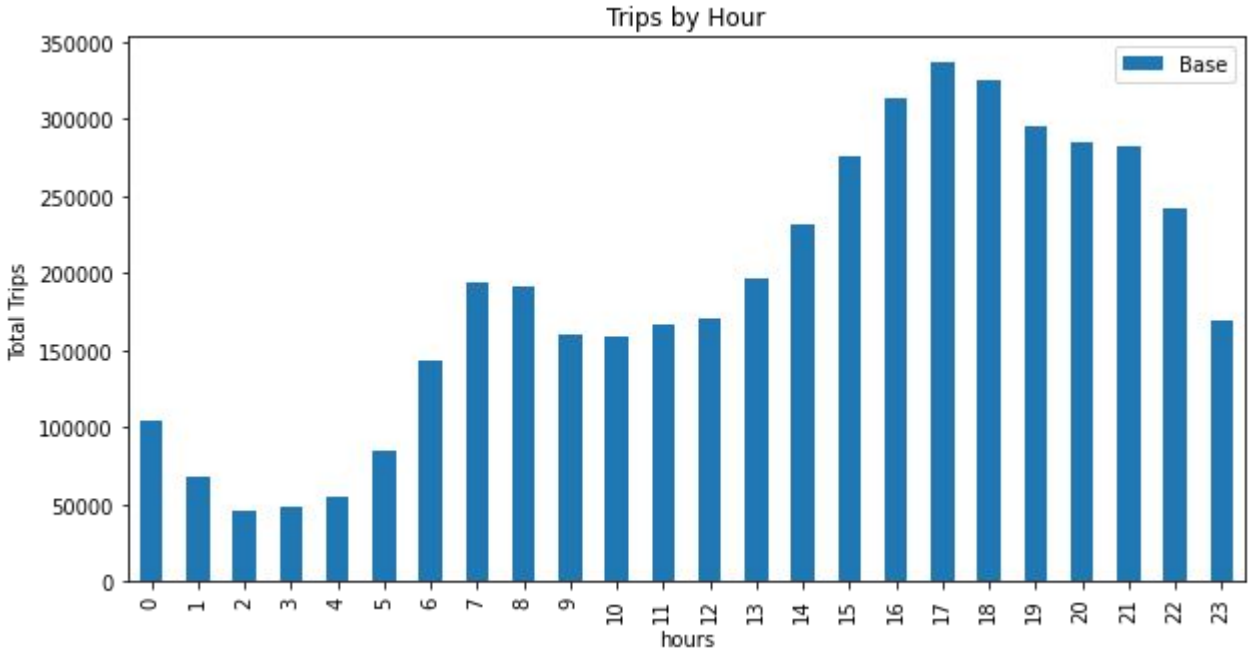
Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns,to spot anomalies,to test hypothesis and to check assumptions with the help of summary statistics and graphical representations.

OBSERVATIONS

Analyzing the peak days in terms of trips



Analysis of the peak hours



Time series forecast error/accuracy measures:

1.MSE(Mean Square Error)

2.RMSE(Root Mean Square Error)

3.MAD(Mean Absolute Deviation)

4.MAPE(Mean Absolute Percentage Error)

Mean Absolute Deviation

Mean absolute deviation is an error statistic that averages the distance between each pair of actual and fitted data points.

The formula for calculating the MAD:

$$\frac{\sum_{t=1}^{n} \left| Y_t - \hat{Y}_t \right|}{n}$$

where *Yt* is the actual value of a point for a given time period *t*, *n* is the total number of fitted points, and $\hat{Y}_t$ is the forecast value for the time period *t*.

Mean Square Error:

if $y_1, \ldots, y_n$ represents a time series, then $\hat{y}_i$ represents the i^{th} forecasted value, where $i \leq n$ then

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE:Mean Square Error
n=no of data points
Y_i **=actual values**
$\hat{Y}_i$ **=forecast value**

Root Mean Square Error

Root mean squared error is an absolute error measure that squares the deviations to keep the positive and negative deviations from canceling one another out. This measure also tends to exaggerate large errors, which can help when comparing methods.

The formula for calculating RMSE:

where *Yt* is the ac$\sqrt{\frac{\sum_{t=1}^{n} (Y_t - \hat{Y}_t)^2}{n}}$ oint for a given time period *t*, *n* is the total number of fitted points, and $\hat{Y}_t$ is the fitted forecast value for the time period *t*.

Mean Absolute Percentage Error

Mean absolute percentage error is a relative error measure that uses absolute values to keep the positive and negative errors from canceling one another out and uses relative errors to enable you to compare forecast accuracy between time-series models.

The formula for calculating t

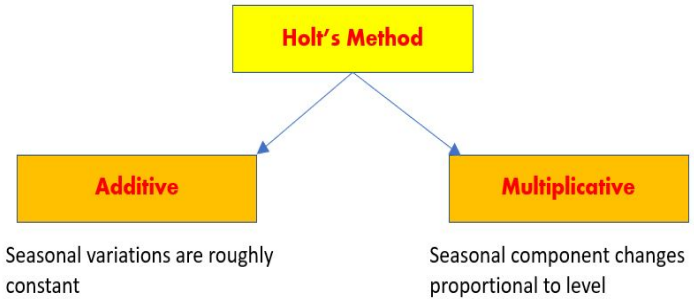
$$MAPE = \frac{\sum_{t=1}^{n} | \hat{Y}_t - Y_t |}{\sum_{t=1}^{n} (| \hat{Y}_t | + | Y_t |)/2}$$

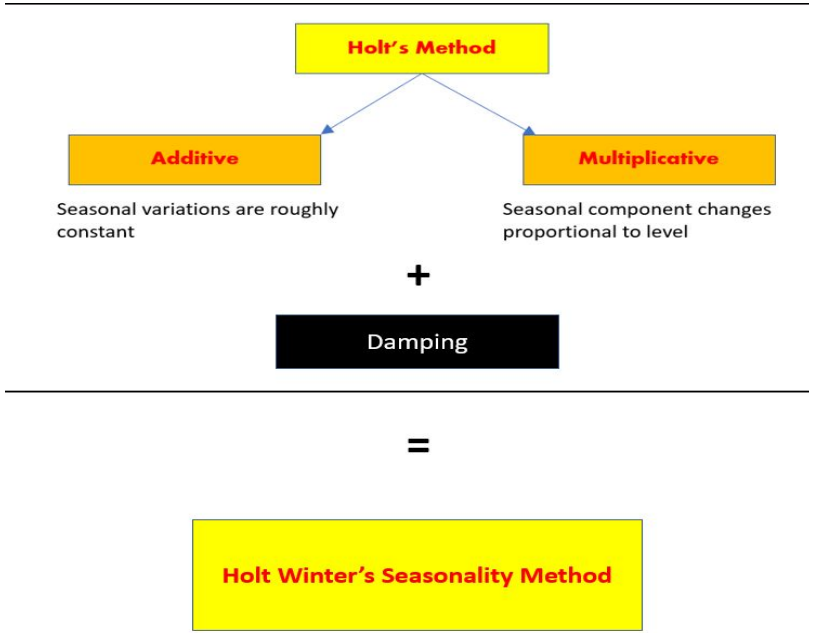
where *Yt* is the actual value of a point for a given time period *t*, *n* is the total number of fitted points, and $\hat{Y}_t$ is the forecast value for the time period *t*.

Holtz Winter Seasonal Method:

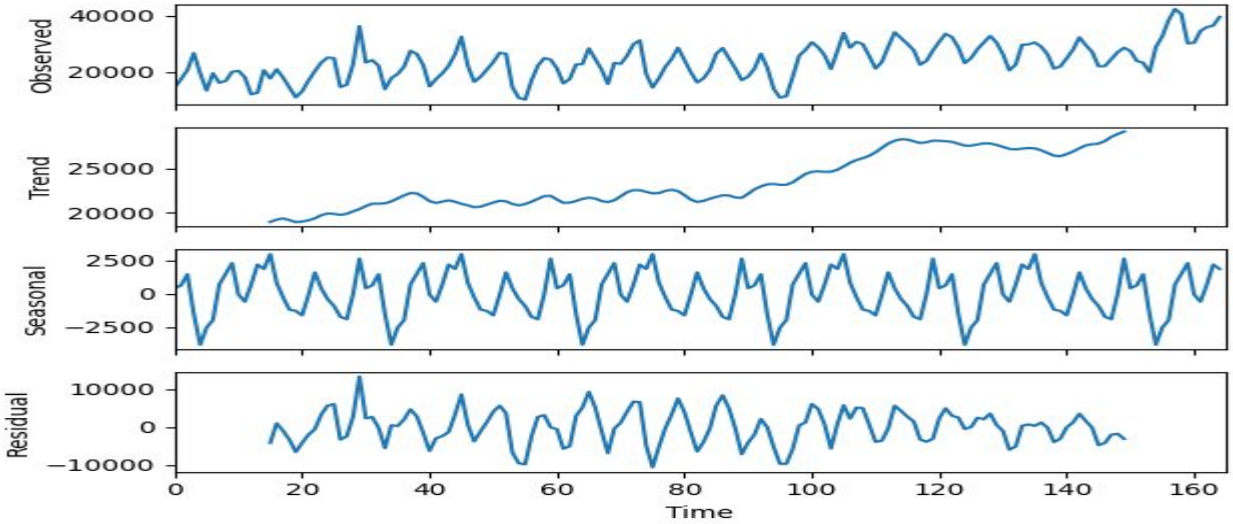
The method is suitable for univariate time series with trend and/or seasonal components.

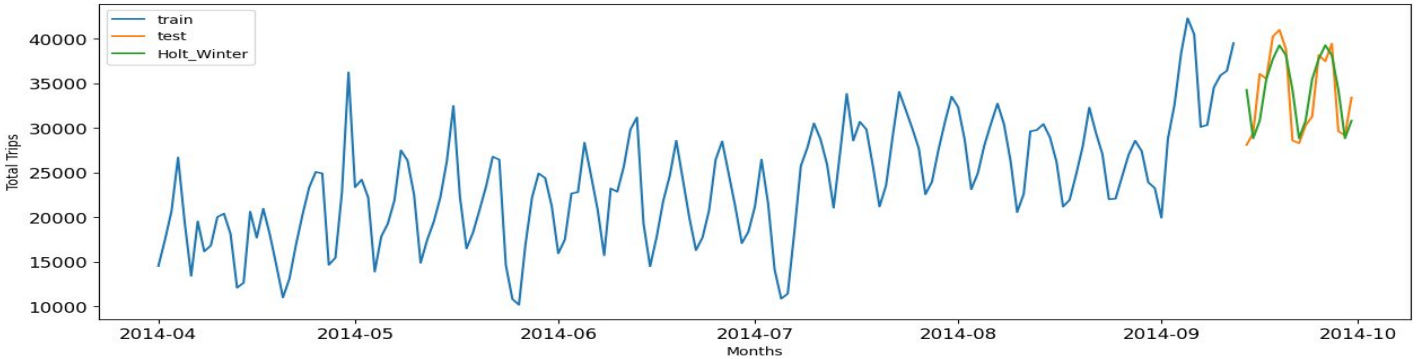
The main factors it takes into account are:level, trend and seasonality.





Holtz Winter Seasonal Method:





Holt's Model Evaluation

```
[ ] from sklearn.metrics import mean_squared_error
    rm=np.sqrt(mean_squared_error(test_ts['Date/Time'],hat_avg['Holt_Winter']))
    rm
```

```
3063.1351183920597
```

Implementation of ARIMA Model

What is the ARIMA Model all about ?

ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.

It is characterized by 3 terms :

p : Order of the AR term, q = order of the MA term, d = number of differencing required to make the time series stationary

Importance of Time Series Forecasting :

Forecasting a time series (like demand and sales) is often of tremendous commercial value.

In most manufacturing companies, it drives the fundamental business planning, procurement and production activities. Any errors in the forecasts will ripple down throughout the supply chain or any business context for that matter. So it's important to get the forecasts accurate in order to save on costs and is critical to success.

STEPS IN IMPLEMENTING THE ARIMA MODEL

Step 1 : Before building the ARIMA model, the first thing we need to do is to make the **time series stationary**. This is done because ARIMA model is a linear regression model and a linear regression works the best when the predictors are not correlated and are independent of each other. This can be done through the method of **differencing** and plotting the rolling mean and standard deviation to check the presence of trends or noise.

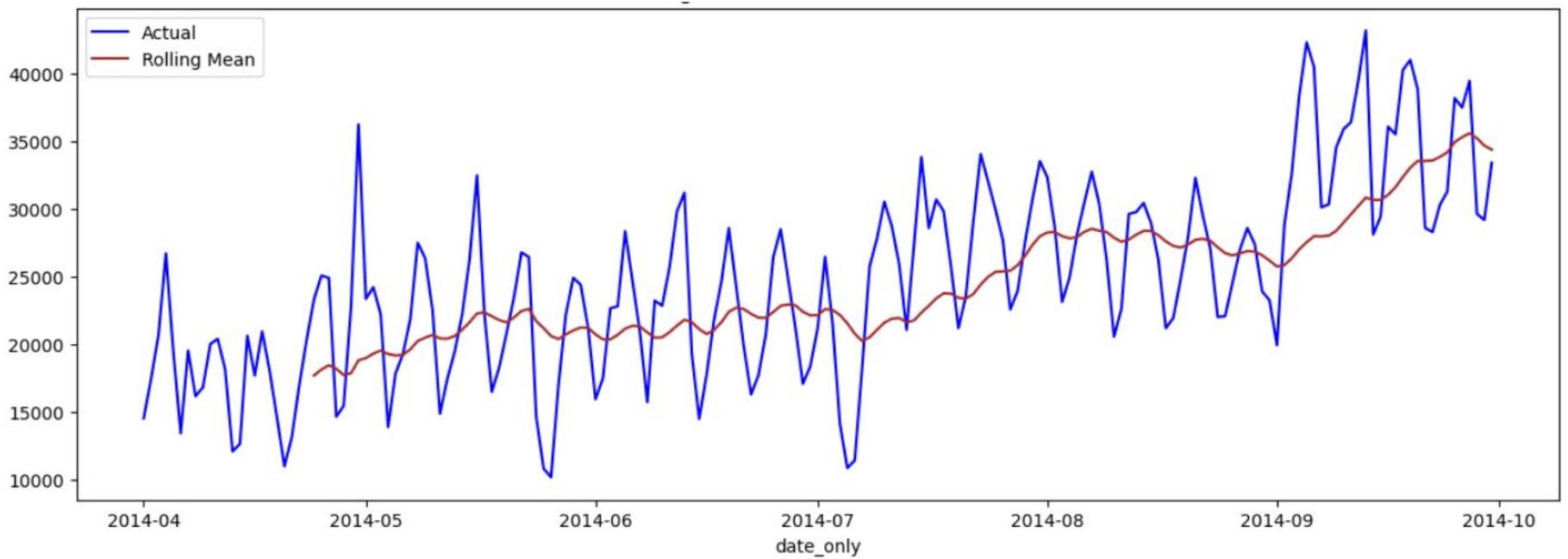
Step 2 : Next, for invoking the **ARIMA model** we have used the AR Model(Auto Regressive) model. This model is based on the principle of of linear regression, that predicts the future values as the function of the previous year values. **Order of Autoregressive model** is **no of the preceding values** that we are going to use to predict the present value. The AR model is employed by putting the value of q = 0 and then plotting the graph.

Step 3: Next, we invoke the Moving Average (MA) model by putting the value of p=0. In our project, we have visualized the moving average graphically by observing it for the first 10 observations. This is a model that proposes that the current values can be predicted as a linear regression of previous errors.

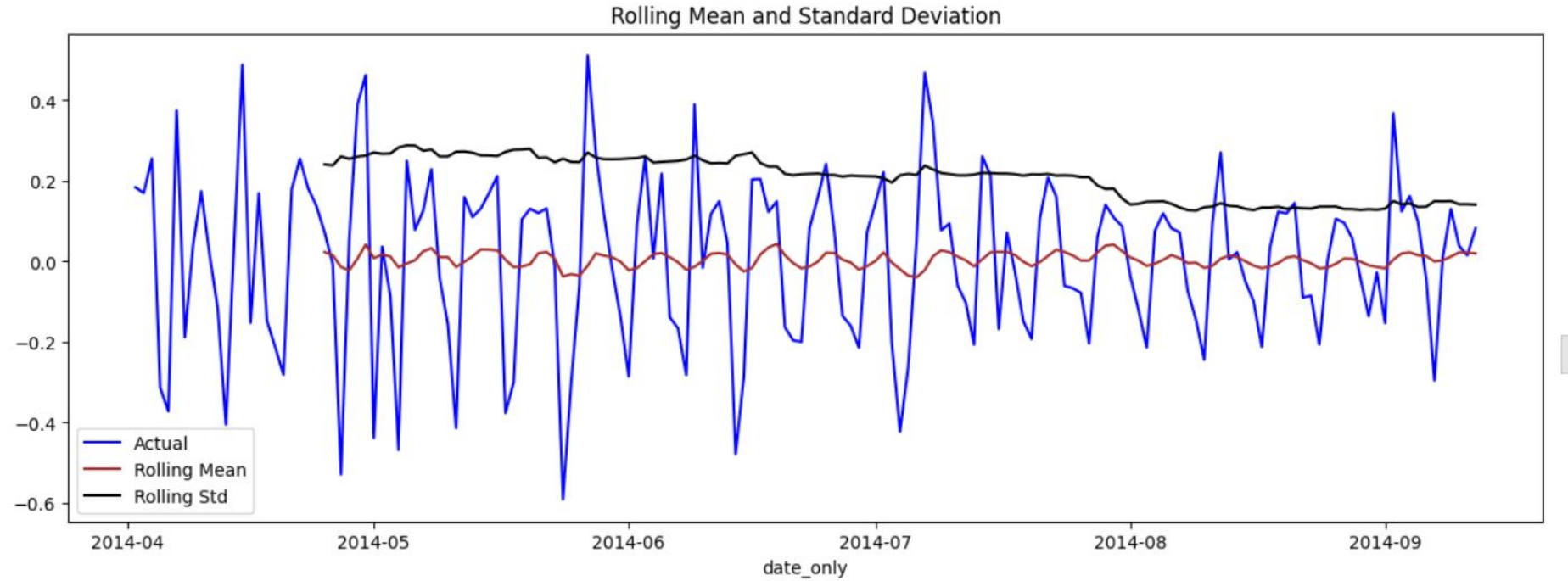
Step 4 : Next, we evaluate the accuracy metrics for time series forecast. For this , we have chosen **Root Mean square error (RMSE)** for checking the accuracy of the model. From the results, we have found that the RMSE value of ARIMA model was found out to be large. Hence, this model is considered unsuitable for time series forecasting.

Results and Observations Of the ARIMA MODEL

Step 1 : Ensuring that the time series is stationary by plotting the rolling mean and standard deviation. The mean of the series should be constant. The variance or standard deviation of the series should not vary with time. Since, it wasn't stationary ,we have used differencing to eliminate it.



```
Train_log = np.log(train_ts['Date/Time'])
train_log_diff = Train_log - Train_log.shift(1)
rolmean = train_log_diff.rolling(24).mean()
rolstd = train_log_diff.rolling(24).std()
```

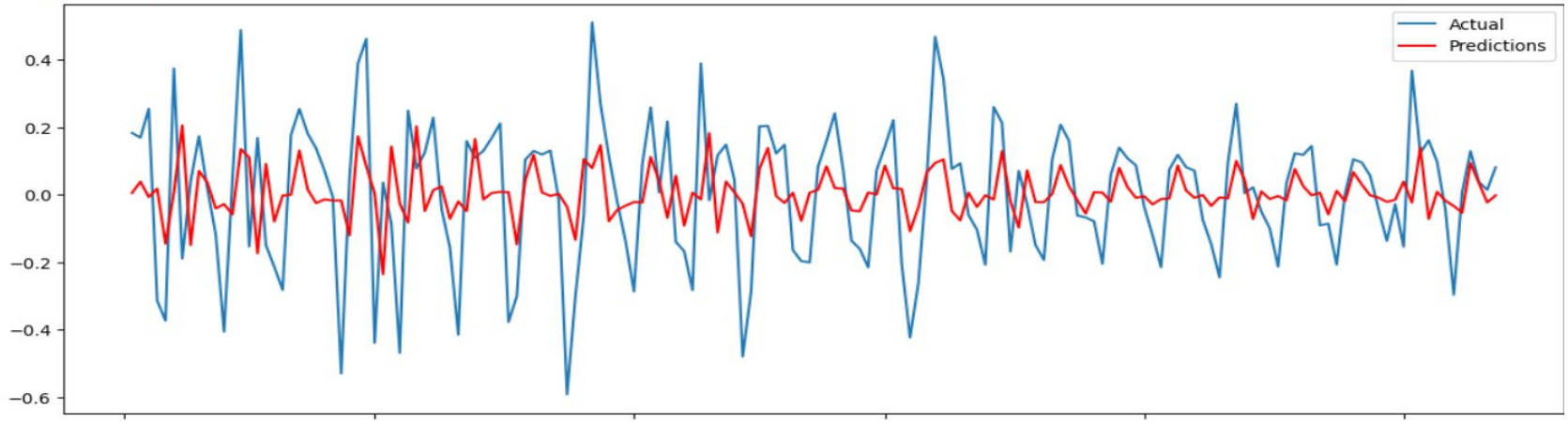


Results and Observations :

Step 2 : Implementing AR model and building it .

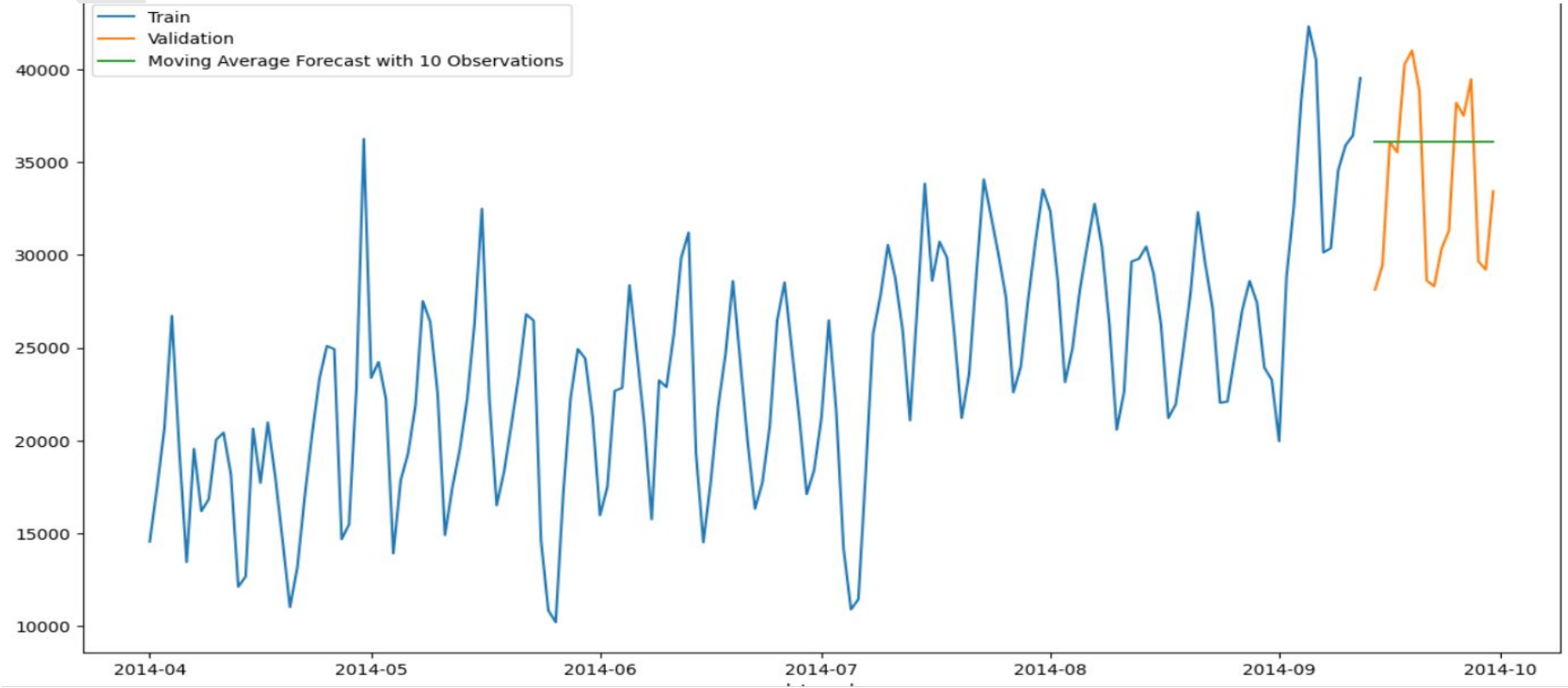
```
[ ]  plt.figure(figsize = (15,5))
     model = ARIMA(Train_log, order = (2,1,0))
     results_AR = model.fit(disp=-1)
     train_log_diff.dropna().plot(kind='line', label = "Actual")
     results_AR.fittedvalues.plot(kind='line', color = 'red', label = 'Predictions')
     plt.legend(loc = 'upper right')
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:165: ValueWarning: No frequency information was provided, so inferred f
  % freq, ValueWarning)
/usr/local/lib/python3.7/dist-packages/statsmodels/tsa/base/tsa_model.py:165: ValueWarning: No frequency information was provided, so inferred f
  % freq, ValueWarning)
<matplotlib.legend.Legend at 0x7f63bb0ea850>
```



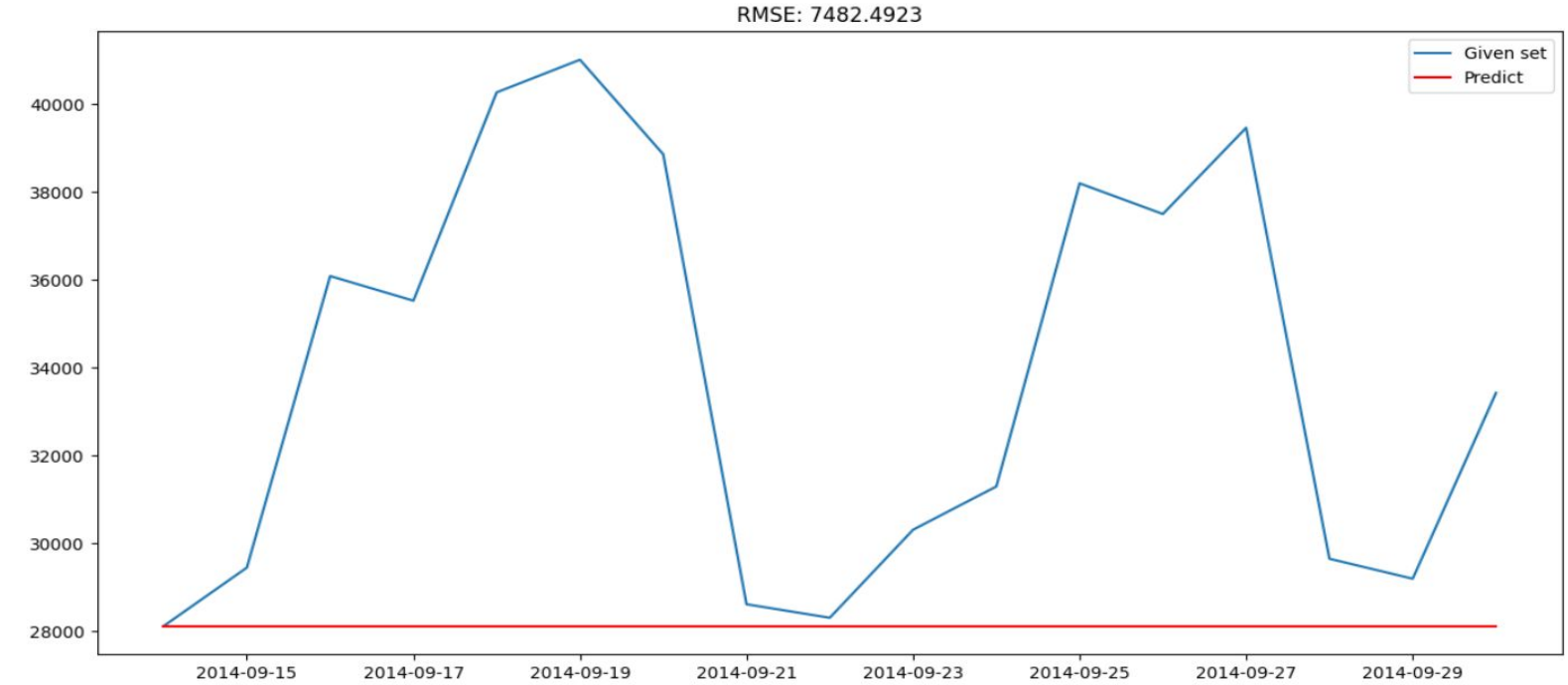
Results and observations

Step 3 : Implementing the Moving Average (MA) by taking the first 10 observations.



Results and Observations

Step 4 : Evaluating the accuracy metrics using root mean square error



Artificial Neural Networks

What are artificial neural networks ?

Artificial neural networks (**ANNs**), usually simply called **neural networks** (**NNs**), are computing systems inspired by the biological neural networks that constitute animal brains

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives a signal then processes it and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs

Implementation in Time Series Forecasting

Step1 : Artificial neural networks is an advanced machine learning technique. Here, we will need to use the Keras model for applying the ANN analysis. So, we will first scale the dataset and split it into train and test data. Next step involves building the model.

Step 2 : While building the Keras Sequential model, the dense layer and callbacks are added which are needed to be used for ANN analysis.

Step 3 : Evaluating the accuracy metrics using the help of Root mean square error (RMSE).

Results :

```
model = Sequential()
model.add(Dense(9, input_dim=1, activation='relu'))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
```

```
y_pred_test_ann = model.predict(X_test)
y_train_pred_ann = model.predict(X_train)
rmse = sqrt(mean_squared_error(y_train,y_train_pred_ann))
print("Train : {:0.3f}".format(rmse))

rmse = sqrt(mean_squared_error(y_test,y_pred_test_ann))
print("Test : {:0.3f}".format(rmse))

model.save('Uber_Data_analysis_ANN')
```

```
Train : 0.089
Test : 0.093
INFO:tensorflow:Assets written to: Uber_Data_analysis_ANN/assets
```

Future Scope and Applications

Time series data analysis and forecasting have become increasingly important due to the massive production of time series data, and as continuous monitoring and collection of such data becomes more common, the need for more efficient analysis and forecasting will only increase. The applications of EDA are equally important in controlling traffic, evaluating the shortest path so as to find the shortest route. a time series analysis is a process of analyzing an observation of data points collected over a period of time, i.e time series data. In time series analysis, data analysts record data observations in constant intervals for a set of time periods instead of recording data observations randomly. The rate of observation (time interval) can be from milliseconds to several years.

In order to inspect “how variables change over time”, a time series data describes the phenomenon under inspection over specific points of time to analyze fluctuations in variables over time. The parameters of interest can vary across domains such as;

- Values recorded by scientific instruments per day
- Number of hits at some websites on a daily basis
- Weekly share values on a stock market
- Number of rainy days per year