CO461 - Data Warehousing and Data Mining

PROJECT REPORT

Weather Forecasting of Time Series data using ARIMA and Auto ARIMA Model

By

Niwedita (171CO227) and Shweta Hariharan Iyer (171CO245)

Final year B-Tech Computer Science and Engineering

National Institute of Technology, Karnataka

27 November 2020

Abstract

Weather Forecasting has become necessary in recent times due to changing and unpredictable weather conditions. Weather predictions are used to protect life and property. The agricultural field is completely dependent on temperature and precipitation forecasts. ARIMA is a popular model for weather forecasting. ARIMA is applied on a univariate time series data to predict the observations' future values. Since ARIMA works only on stationary data, its variant Auto ARIMA is introduced for performing weather forecasting on non-stationary data. In this project, we implement ARIMA and Auto ARIMA for stationary and nonstationary data respectively. We also introduce a slight modification in the data cleaning process. We then compare the results obtained by ARIMA and Auto ARIMA before and after the introduction of the modification.

TABLE OF CONTENTS

List of Figures

List of Tables

I. Introduction

Weather forecasting is performed on Time Series Data. A time series is a sequence where a metric is recorded over regular time intervals. We use this type of series to forecast any event in the future such as temperature, rainfall, humidity, budgets etc.

For prediction we are going to use one of the most popular models for time series, Autoregressive Integrated Moving Average (ARIMA) which is a standard statistical model for time series forecast and analysis. An ARIMA model can be understood by outlining each of its components as follows:

- Autoregression (AR) - refers to a model that shows a changing variable that regresses on its own lagged, or prior, values. The notation AR(p) indicates an autoregressive model of order p.

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_p Y_{t-p} + \varepsilon_1$$

- Integrated (I) - represents the differencing of raw observations to allow for the time series to become stationary, i.e., data values are replaced by the difference between the data values and the previous values.

- Moving average (MA) - incorporates the dependency between an observation and a residual error from a moving average model applied to lagged observations. The notation MA(q) refers to the moving average model of order q.

$$Y = \alpha + \varepsilon_t + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + ... + \phi_q \varepsilon_{t-q}$$

Equation of the ARIMA model- Combination of AR and MA models [2]

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + ... + \beta_p Y_{t-p} \varepsilon_t + \phi_1 \varepsilon_{t-1} + \phi_2 \varepsilon_{t-2} + ... + \phi_q \varepsilon_{t-q}$$

Before using the ARIMA model, we need to check whether the dataset is stationary or not. Check for below necessary conditions:

- Constant mean
- Constant variance
- An auto covariance that does not depend on time

If we have constant Mean and Variance, and our Test statistic is less than Critical Values, so we already have a stationary Time series. So our 'd' value will become 0 in the ARIMA Model.
And if it was non-stationary, in that case we would use below techniques to make it stationary by using any of the below techniques:

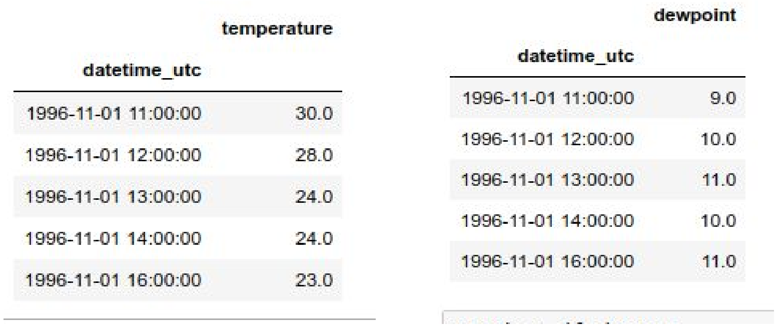
- Decomposing
- Differencing [1]

Auto ARIMA is a variance of ARIMA that is particularly useful for non-stationary dataset. Auto ARIMA saves the task of differencing and computing p, q, d values of ARIMA. Forecasting is done directly by fitting the Auto ARIMA model on the univariate time series data.

The organization of this work follows the order below. Section II describes the dataset. Section III explains the methodology involved in weather forecasting. Section IV summarizes the methods and discusses the modifications . Finally, Section V summarizes the model.

II. Dataset Description

A time series weather dataset is used to implement the ARIMA model of forecasting. The dataset contains weather data for New Delhi, India from year 1996 to 2017.This weather dataset includes several attributes such as temperature, dewpoint, humidity, wind direction etc. We apply the ARIMA model on various univariate time series from the New Delhi weather dataset. Univariate time series is a time series that consists of only single observations recorded sequentially over equal time increments. Here, the ARIMA model of weather forecasting is applied to the temperature data from the weather dataset. The Auto ARIMA model of weather forecasting is applied to the dewpoint data from the weather dataset.
Following are some data values from temperature and dewpoint dataset respectively.



	temperature
datetime_utc	
1996-11-01 11:00:00	30.0
1996-11-01 12:00:00	28.0
1996-11-01 13:00:00	24.0
1996-11-01 14:00:00	24.0
1996-11-01 16:00:00	23.0

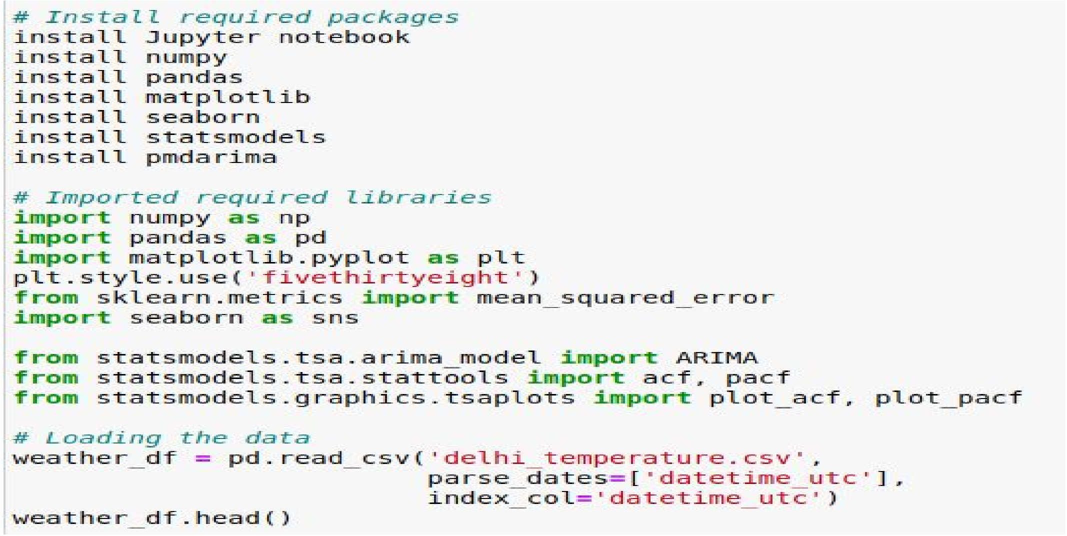
	dewpoint
datetime_utc	
1996-11-01 11:00:00	9.0
1996-11-01 12:00:00	10.0
1996-11-01 13:00:00	11.0
1996-11-01 14:00:00	10.0
1996-11-01 16:00:00	11.0

Fig1: Temperature and Dewpoint dataset

III. Materials and Methods

The following process is used to implement the ARIMA model of weather forecasting on temperature data of New Delhi.

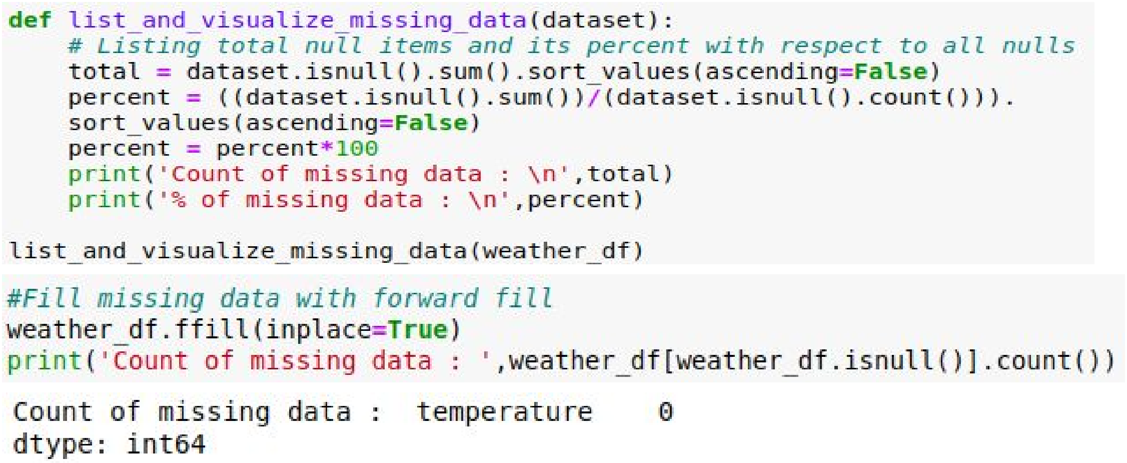
1. Import statsmodels and pmdarima Python module for loading the ARIMA model. Import numpy, pandas, matplotlib, seaborn Python libraries for implementation and load the temperature dataset for ARIMA forecasting.



```
# Install required packages
install Jupyter notebook
install numpy
install pandas
install matplotlib
install seaborn
install statsmodels
install pmdarima

# Imported required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
from sklearn.metrics import mean_squared_error
import seaborn as sns

from statsmodels.tsa.arima_model import ARIMA
from statsmodels.tsa.stattools import acf, pacf
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf

# Loading the data
weather_df = pd.read_csv('delhi_temperature.csv',
                         parse_dates=['datetime_utc'],
                         index_col='datetime_utc')
weather_df.head()
```

Fig2: Import libraries and load dataset

2. Perform data cleaning on the dataset to fill the missing values.



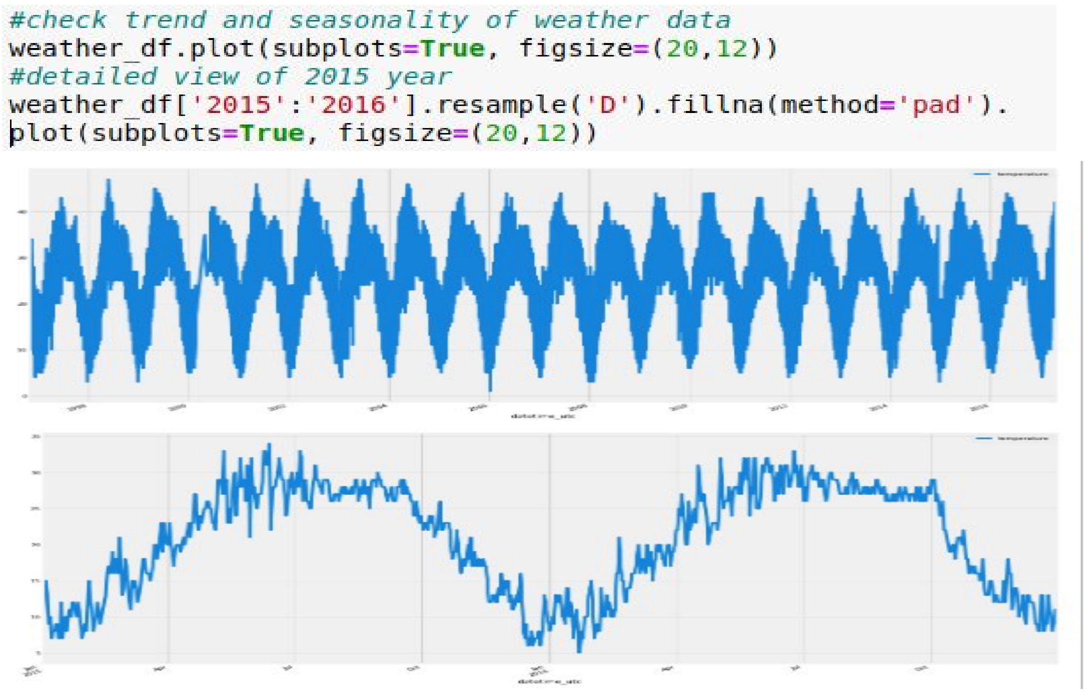
```
def list_and_visualize_missing_data(dataset):
    # Listing total null items and its percent with respect to all nulls
    total = dataset.isnull().sum().sort_values(ascending=False)
    percent = ((dataset.isnull().sum())/(dataset.isnull().count())).
    sort_values(ascending=False)
    percent = percent*100
    print('Count of missing data : \n',total)
    print('% of missing data : \n',percent)

list_and_visualize_missing_data(weather_df)
```

```
#Fill missing data with forward fill
weather_df.ffill(inplace=True)
print('Count of missing data : ',weather_df[weather_df.isnull()].count())
```

```
Count of missing data :  temperature    0
dtype: int64
```

Fig3: Data cleaning

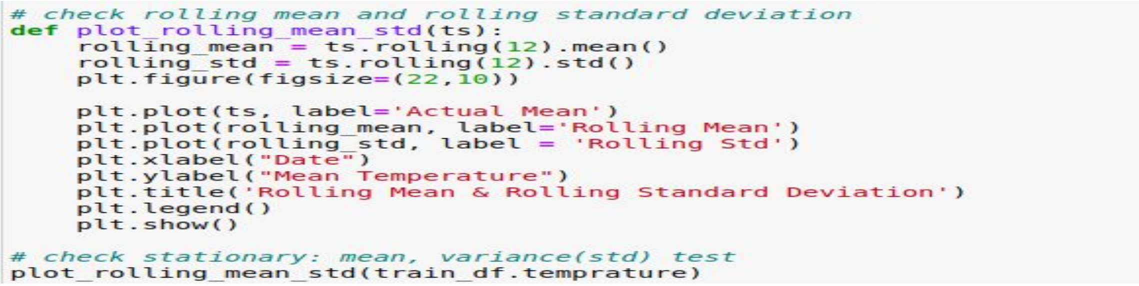
3. Plot the data to check the trend and seasonality of the time series weather data.



```
#check trend and seasonality of weather data
weather_df.plot(subplots=True, figsize=(20,12))
#detailed view of 2015 year
weather_df['2015':'2016'].resample('D').fillna(method='pad').
plot(subplots=True, figsize=(20,12))
```

Fig4: Seasonality of the data

4. Plot the actual mean, rolling mean and standard deviation of the training data to check if the data is stationary. This is required as the ARIMA model can only work only stationary data. For forecasting non-stationary data, the Auto ARIMA model is used.



```
# check rolling mean and rolling standard deviation
def plot_rolling_mean_std(ts):
    rolling_mean = ts.rolling(12).mean()
    rolling_std = ts.rolling(12).std()
    plt.figure(figsize=(22,10))

    plt.plot(ts, label='Actual Mean')
    plt.plot(rolling_mean, label='Rolling Mean')
    plt.plot(rolling_std, label = 'Rolling Std')
    plt.xlabel("Date")
    plt.ylabel("Mean Temperature")
    plt.title('Rolling Mean & Rolling Standard Deviation')
    plt.legend()
    plt.show()

# check stationary: mean, variance(std) test
plot_rolling_mean_std(train_df.temprature)
```

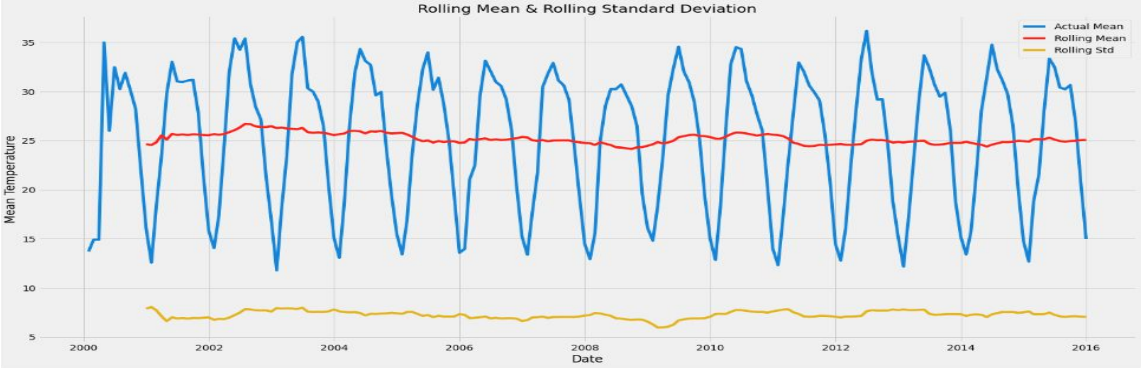
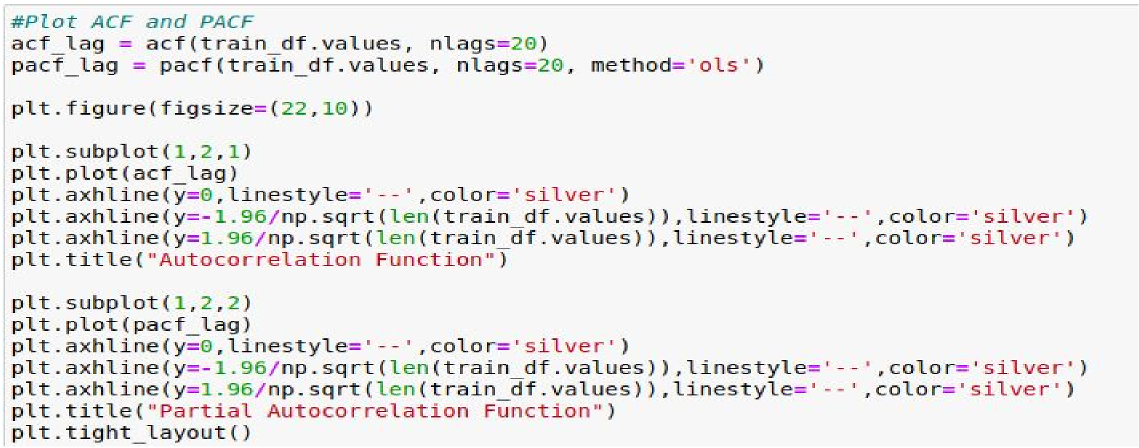


Fig5: Mean, rolling mean and standard deviation of data

5. For stationary data, plot ACF and PACF plots to get parameter values for the ARIMA model.



```
#Plot ACF and PACF
acf_lag = acf(train_df.values, nlags=20)
pacf_lag = pacf(train_df.values, nlags=20, method='ols')

plt.figure(figsize=(22,10))

plt.subplot(1,2,1)
plt.plot(acf_lag)
plt.axhline(y=0,linestyle='--',color='silver')
plt.axhline(y=-1.96/np.sqrt(len(train_df.values)),linestyle='--',color='silver')
plt.axhline(y=1.96/np.sqrt(len(train_df.values)),linestyle='--',color='silver')
plt.title("Autocorrelation Function")

plt.subplot(1,2,2)
plt.plot(pacf_lag)
plt.axhline(y=0,linestyle='--',color='silver')
plt.axhline(y=-1.96/np.sqrt(len(train_df.values)),linestyle='--',color='silver')
plt.axhline(y=1.96/np.sqrt(len(train_df.values)),linestyle='--',color='silver')
plt.title("Partial Autocorrelation Function")
plt.tight_layout()
```

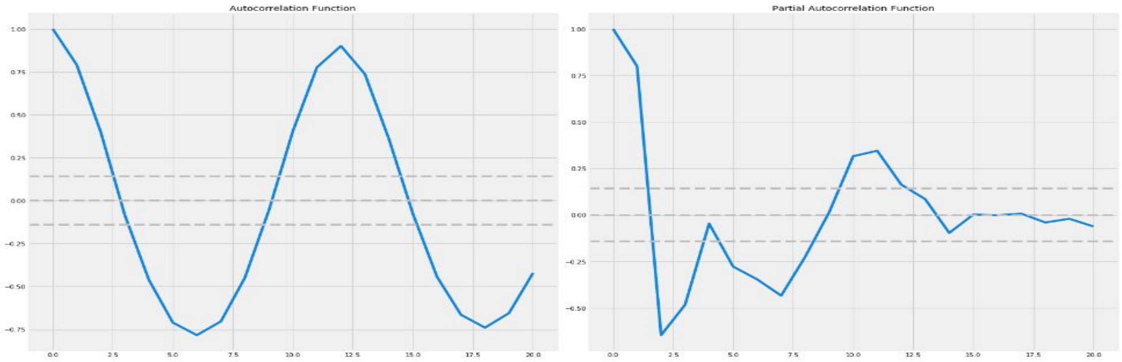
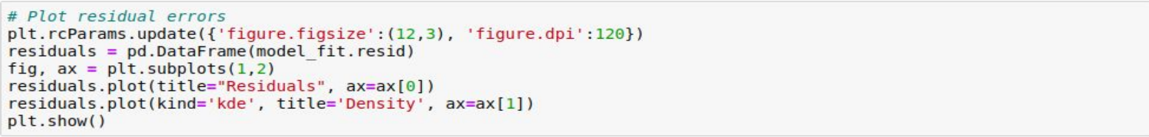


Fig6: ACF and PACF plots

Grey dotted lines are confidence intervals which are used to find the value of p and q.

p - the point where PACF crosses the upper confidence level. In our case, *p = 2.
q - the point where ACF crosses the upper confidence level. In our case, *q = 2.
d - number of nonseasonal differences needed for stationarity. In this case it is 0, since this series is already stationary.

6. Apply ARIMA model on the stationary training data and perform model fitting. Plot the model's residual errors.



```
# Plot residual errors
plt.rcParams.update({'figure.figsize':(12,3), 'figure.dpi':120})
residuals = pd.DataFrame(model_fit.resid)
fig, ax = plt.subplots(1,2)
residuals.plot(title="Residuals", ax=ax[0])
residuals.plot(kind='kde', title='Density', ax=ax[1])
plt.show()
```

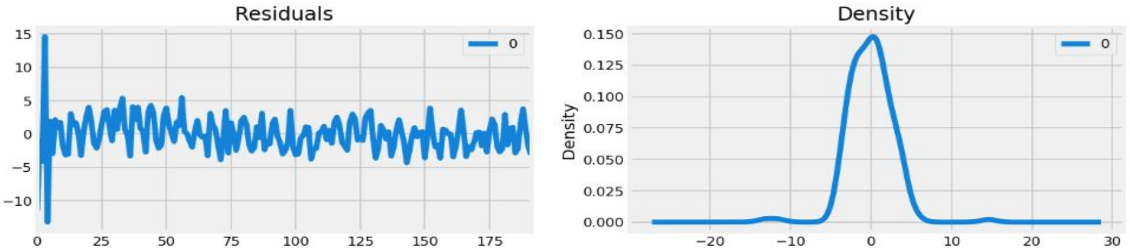
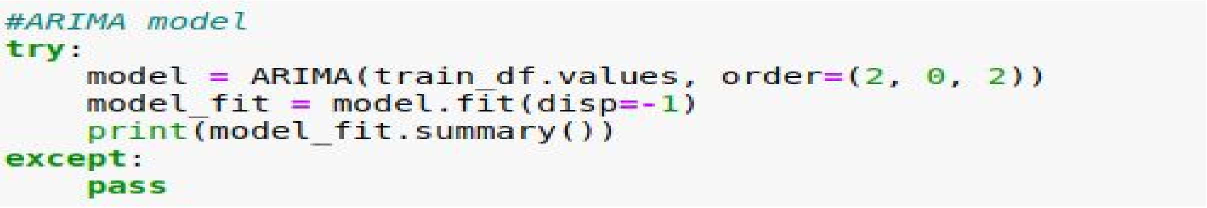
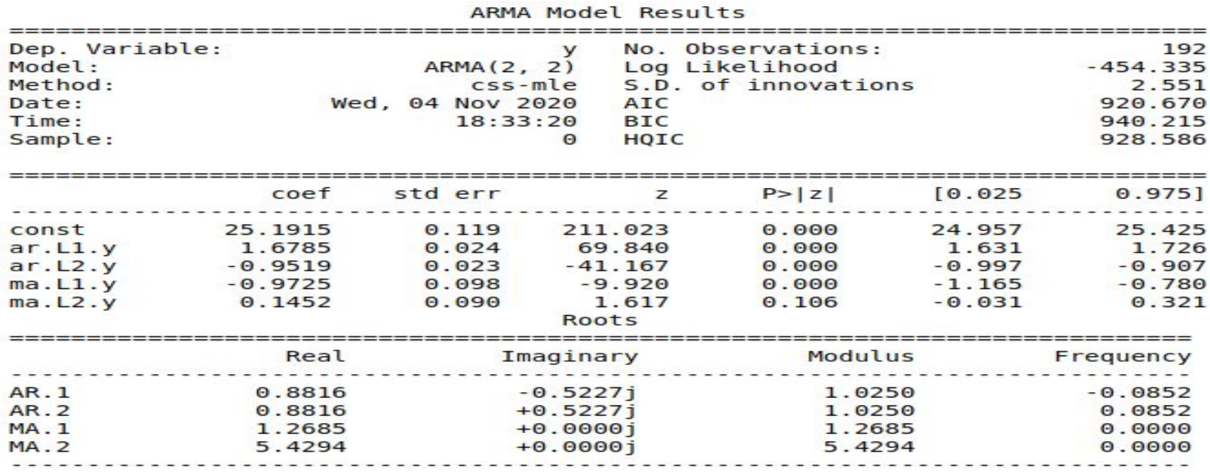


Fig7: Residual error plots

7. Perform weather forecasting using the above model on the testing data. Plot the actual and forecast weather values of testing data.



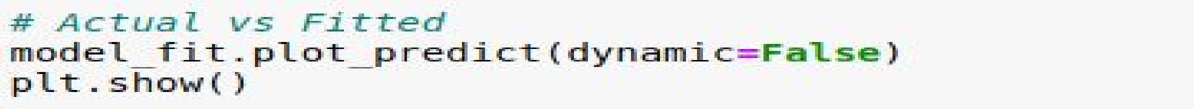
```
#ARIMA model
try:
    model = ARIMA(train_df.values, order=(2, 0, 2))
    model_fit = model.fit(disp=-1)
    print(model_fit.summary())
except:
    pass
```



```
                              ARMA Model Results
==============================================================================
Dep. Variable:                      y   No. Observations:                  192
Model:                     ARMA(2, 2)   Log Likelihood                -454.335
Method:                       css-mle   S.D. of innovations              2.551
Date:                Wed, 04 Nov 2020   AIC                            920.670
Time:                        18:33:20   BIC                            940.215
Sample:                             0   HQIC                           928.586

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         25.1915      0.119    211.023      0.000      24.957      25.425
ar.L1.y        1.6785      0.024     69.840      0.000       1.631       1.726
ar.L2.y       -0.9519      0.023    -41.167      0.000      -0.997      -0.907
ma.L1.y       -0.9725      0.098     -9.920      0.000      -1.165      -0.780
ma.L2.y        0.1452      0.090      1.617      0.106      -0.031       0.321
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.8816            -0.5227j            1.0250           -0.0852
AR.2            0.8816            +0.5227j            1.0250            0.0852
MA.1            1.2685            +0.0000j            1.2685            0.0000
MA.2            5.4294            +0.0000j            5.4294            0.0000
-----------------------------------------------------------------------------
```

Fig8: ARIMA model results

The 'coef' column in the ARIMA model results summary gives the coefficients of AR and MA models. They are then combined to form the equation for the ARIMA model. This ARIMA equation is then used to forecast weather values.



```
# Actual vs Fitted
model_fit.plot_predict(dynamic=False)
plt.show()
```

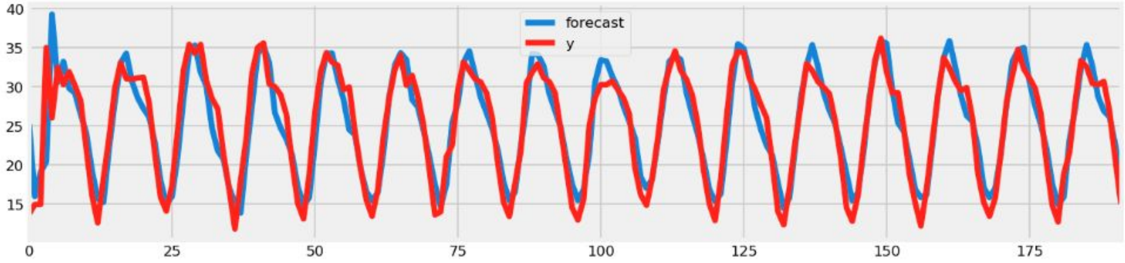


Fig9: Actual versus forecast plot

8. To numerically compute the accuracy of the model, calculate Mean Squared Error(MSE) between the actual and forecast weather values of testing data.

Non- Stationary Data :

For Non-stationary data, the ARIMA model of weather forecasting will not give accurate results. For this, the Auto ARIMA model is used. Auto ARIMA bypasses the need to have the data stationary and computing the parameter values for the ARIMA model. Here, the time series data of dewpoint from the New Delhi dataset is non-stationary. The data will not have a constant rolling mean and standard deviation. For such data, using the same above mentioned procedure, Auto ARIMA model is applied. ACF and PACF plots need not be plotted as Auto ARIMA bypasses it.

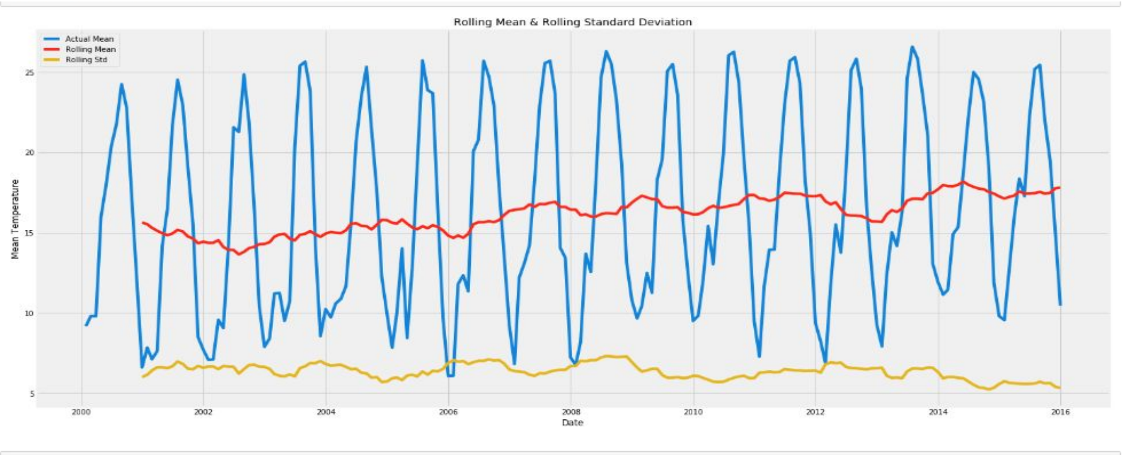


Fig10 : Rolling mean and standard deviation of dewpoint dataset.

We have implemented ARIMA and Auto ARIMA on stationary temperature data and non-stationary dewpoint data respectively. In the implementation, data cleaning is performed using Forward Fill. Forward Fill method propagates the last valid observation forward to fill the missing values.

We propose a model to modify this Data Cleaning process. The mean of all the observations is used to fill the missing values. This modification is applied to both the stationary temperature data and non-stationary dewpoint data.

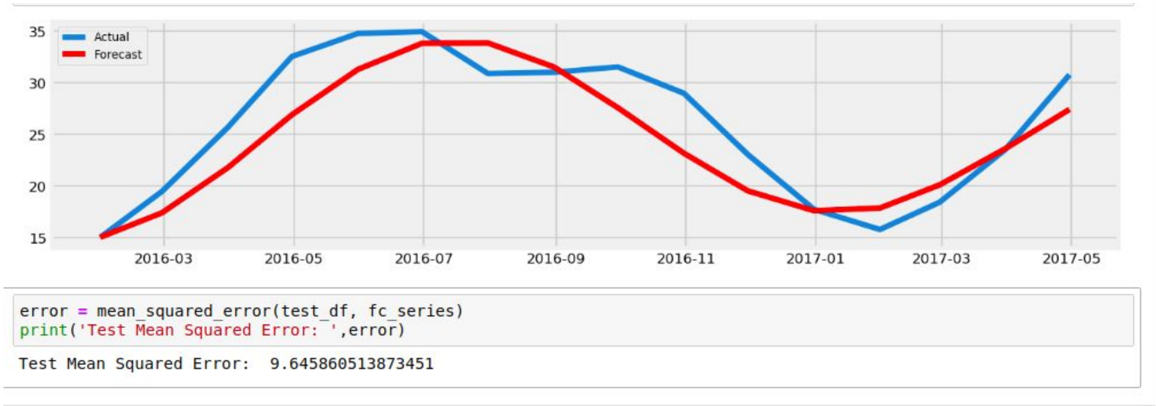
IV. Results and Discussions

The weather forecasting experiment is conducted on two observations from the New Delhi time series weather dataset- temperature data which is stationary and dewpoint data which is non-stationary. The ARIMA model of weather forecasting is used for temperature data and the Auto ARIMA model is used for dewpoint data. Mean Squared Error(MSE) is computed to evaluate the model's performance. Further, the data cleaning process is modified to fill the missing data values using the mean of the observations. The following table summarises the results obtained. It shows the MSE obtained from applying ARIMA and Auto ARIMA on temperature and dewpoint data respectively, with once the Forward Fill and then the Mean method for Data Cleaning, to fill missing values.

	Temperature data (Stationary)	Dewpoint data (Non- Stationary)
Forward Fill	9.645860513873451	2.245820662235592
Mean of the Observations	9.785734936454212	2.1717549429236898

Table1: Comparison of MSE of temperature and dewpoint data with Forward Fill and Mean of the Observations data cleaning methods

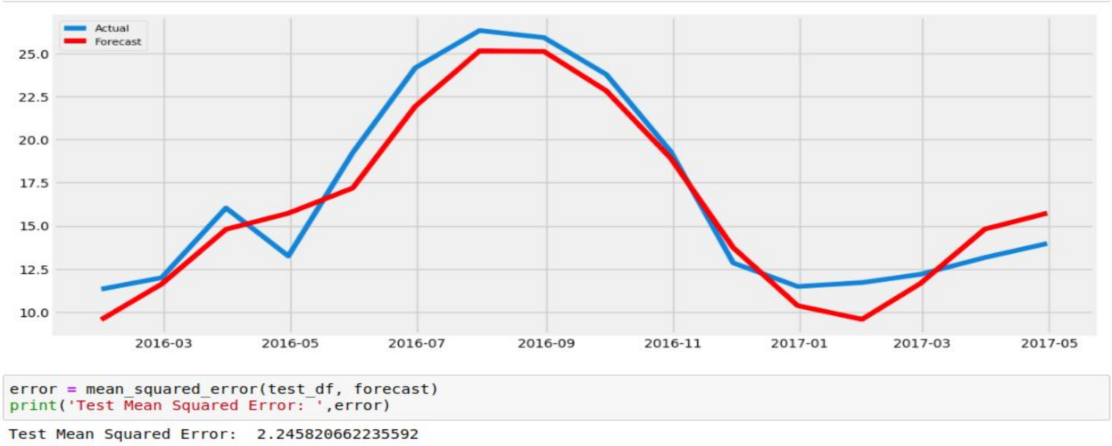
The following graphs are plots of Actual vs Forecast Values of temperature and dewpoint using ARIMA and Auto ARIMA. The first two graphs show the ARIMA and Auto ARIMA implementation using Forward Fill Data Cleaning process to fill the missing values. The next two graphs show the ARIMA and Auto ARIMA implementation using Mean of Observations Data Cleaning process to fill the missing values.



```
error = mean_squared_error(test_df, fc_series)
print('Test Mean Squared Error: ',error)
```

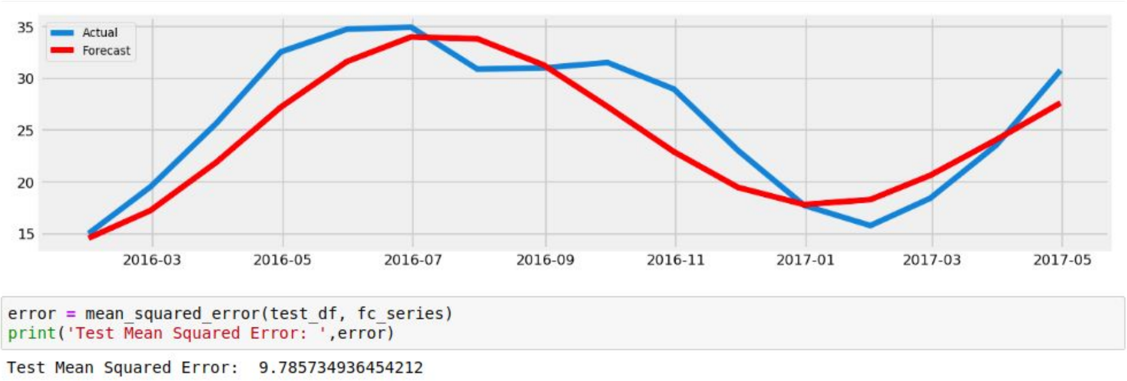
```
Test Mean Squared Error:  9.645860513873451
```

Fig11 : Plot of Actual vs ARIMA forecast values of temperature data using Forward Fill Data Cleaning process



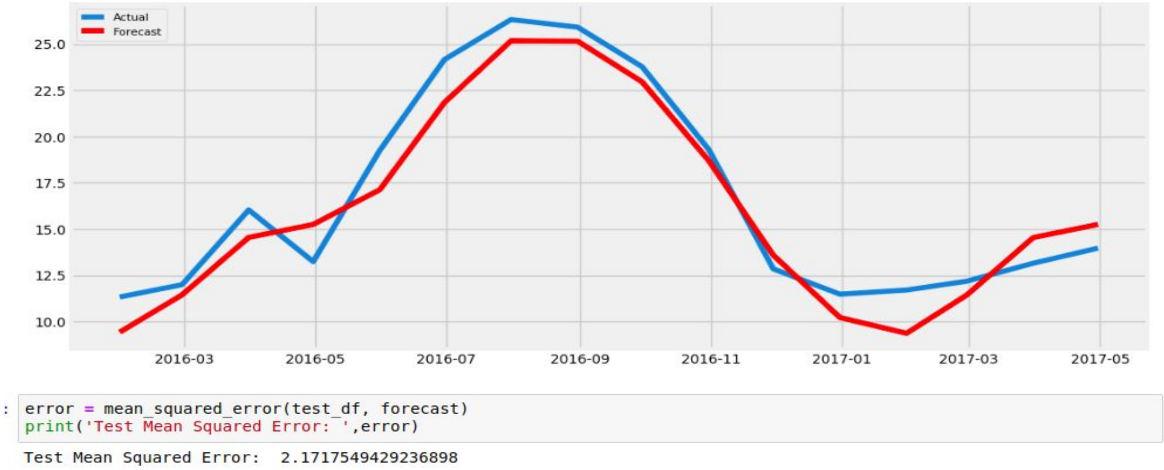
```
error = mean_squared_error(test_df, forecast)
print('Test Mean Squared Error: ',error)

Test Mean Squared Error:  2.245820662235592
```

Fig12 : Plot of Actual vs Auto ARIMA forecast values of dewpoint data using Forward Fill Data Cleaning process



```
error = mean_squared_error(test_df, fc_series)
print('Test Mean Squared Error: ',error)

Test Mean Squared Error:  9.785734936454212
```

Fig13 : Plot of Actual vs ARIMA forecast values of temperature data using Mean of the Observations Data Cleaning process



```
: error = mean_squared_error(test_df, forecast)
  print('Test Mean Squared Error: ',error)

  Test Mean Squared Error:  2.1717549429236898
```

Fig14 : Plot of Actual vs Auto ARIMA forecast values of dewpoint data using Mean of the Observations Data Cleaning process

The above table and graph results show that for Stationary data, Forward fill Data Cleaning in ARIMA forecasting is a better approach than Mean of the Observations to fill missing values. For Non-stationary data, Mean of the Observations Data Cleaning process in Auto ARIMA forecasting is a better approach than Forward fill to fill missing values.

V. Conclusion

We have implemented the ARIMA model of weather forecasting on New Delhi's weather dataset. The ARIMA model is applied on New Delhi's temperature data as the data is stationary. Auto ARIMA model is applied on New Delhi's dewpoint data as the data is non-stationary. The implementation uses Forward Fill method in the data cleaning process to fill missing values. We proposed a modification in the data cleaning process- to fill missing values using the mean of the observations. ARIMA and Auto ARIMA are then applied on the modified temperature and dewpoint data. Mean Squared Error(MSE) is used to evaluate the model's performance for a certain data cleaning method.

We observe that for Stationary data, Forward fill Data Cleaning in ARIMA forecasting is a better approach than Mean of the Observations to fill missing values. For Non-stationary data, Mean of the Observations Data Cleaning process in Auto ARIMA forecasting is a better approach than Forward fill to fill missing values.

VI. References

1) Krishna, G.V., 2015. An integrated approach for weather forecasting based on data mining and forecasting analysis. *International Journal of Computer Applications*, *120*(11).

2) Saikhu, A., Arifin, A.Z. and Fatichah, C., 2017, October. Rainfall forecasting by using autoregressive integrated moving average, single input and multi input transfer function. In *2017 11th International Conference on Information & Communication Technology and System (ICTS)* (pp. 85-90). IEEE.

3) Yang, Y., Lin, H., Guo, Z. and Jiang, J., 2007. A data mining approach for heavy rainfall forecasting based on satellite image sequence analysis. *Computers & geosciences*, *33*(1), pp.20-30.