Course Name:	**Data Analysis Laboratory (216H03L501)**	**Semester:**	**V**
Date of Performance:	**13/10/2025**	**DIV/ Batch No:**	**HDA2**
Student Name:	**Aaryan Sharma**	**Roll No:**	**16010123012**

TITLE: Perform forecasting/predict using time series analysis (AR/MA/ARIMASARIMA)

AIM: To perform forecasting using time series analysis

Expected OUTCOME of Experiment:

CO4: Perform Time series Analytics and forecasting

Books/ Journals/ Websites referred:

https://www.geeksforgeeks.org/machine-learning/time-series-analysis-and-forecasting/

Pre Lab/ Prior Concepts:

Students should have a basic understanding of: Time series Analytics and forecasting

Procedure:

Data set Used: AirQuality dataset

Note: google colab is shared please refer

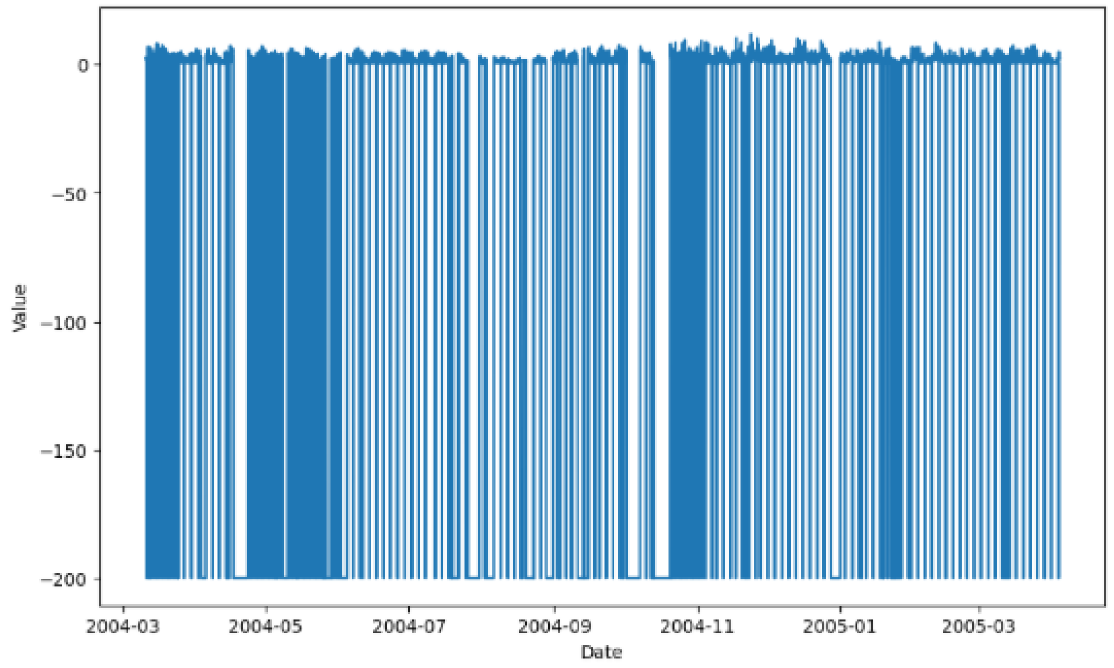
Step1: Select and Load the dataset

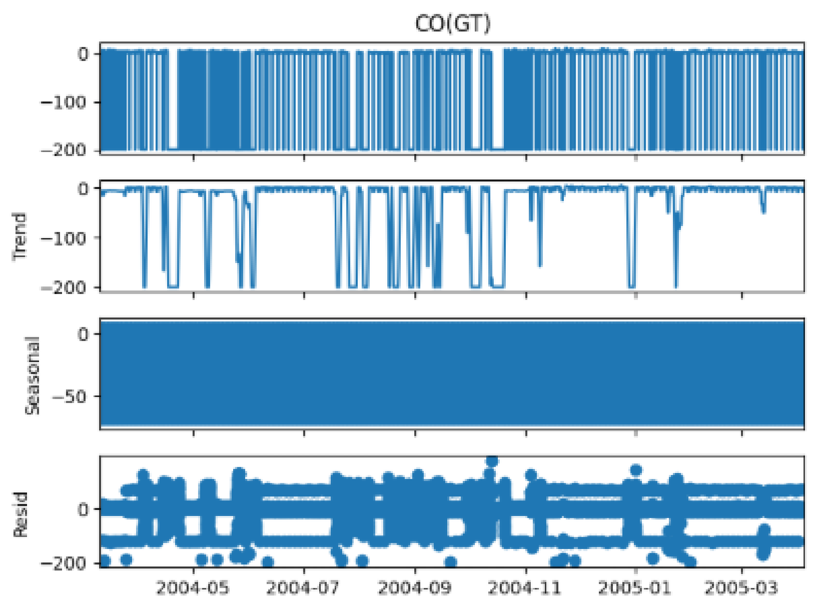
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
# Load the data
import pandas as pd
# Load the dataset
data = pd.read_csv('AirQualityUCIfinal.csv')
# Combine 'Date' and 'Time' into a single 'datetime' column
data['datetime'] = pd.to_datetime(data['Date'] + ' ' + data['Time'], format='%d-%m-%Y %H:%M:%S')
# Set the 'datetime' column as the index
data.set_index('datetime', inplace=True)
# Drop the original 'Date' and 'Time' columns as they are no longer needed
data.drop(columns=['Date', 'Time'], inplace=True)
# Optional: Convert columns to numeric if they are not already
data = data.apply(pd.to_numeric, errors='coerce')
# Display the first few rows of the dataset
print(data.head())
series = data['CO(GT)']  # Adjust column names as needed
```

```
                     CO(GT)  PT08.S1(CO)  NMHC(GT)  C6H6(GT)  PT08.S2(NMHC)  \
datetime
2004-03-10 18:00:00     2.6         1360       150      11.9           1046
2004-03-10 19:00:00     2.0         1292       112       9.4            955
2004-03-10 20:00:00     2.2         1402        88       9.0            939
2004-03-10 21:00:00     2.2         1376        80       9.2            948
2004-03-10 22:00:00     1.6         1272        51       6.5            836

                     NOx(GT)  PT08.S3(NOx)  NO2(GT)  PT08.S4(NO2)  \
datetime
2004-03-10 18:00:00      166          1056      113          1692
2004-03-10 19:00:00      103          1174       92          1559
2004-03-10 20:00:00      131          1140      114          1555
2004-03-10 21:00:00      172          1092      122          1584
2004-03-10 22:00:00      131          1205      116          1490

                     PT08.S5(O3)     T    RH      AH
datetime
2004-03-10 18:00:00         1268  13.6  48.9  0.7578
2004-03-10 19:00:00          972  13.3  47.7  0.7255
2004-03-10 20:00:00         1074  11.9  54.0  0.7502
2004-03-10 21:00:00         1203  11.0  60.0  0.7867
2004-03-10 22:00:00         1110  11.2  59.6  0.7888
```

Step2: Visualize the data

```python
# Visualize the data
plt.figure(figsize=(10, 6))
plt.plot(series)
plt.title('Time Series Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.show()
# Decompose the data
decomposition = seasonal_decompose(series, model='additive')
fig = decomposition.plot()
plt.show()
```





Step 3: Fit the model (ARIMA Model is Used)

```
# Fit ARIMA model
model = ARIMA(series, order=(3, 1, 0))  # Adjust the order as needed
model_fit = model.fit()
# Summary of the model
print(model_fit.summary())
```

```
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency h will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency h will be used.
  self._init_dates(dates, freq)
/usr/local/lib/python3.12/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: No frequency information was provided, so inferred frequency h will be used.
  self._init_dates(dates, freq)
                               SARIMAX Results
==============================================================================
Dep. Variable:                  CO(GT)   No. Observations:                 9357
Model:                  ARIMA(3, 1, 0)   Log Likelihood              -46789.114
Date:                 Mon, 13 Oct 2025   AIC                          93586.227
Time:                         05:22:42   BIC                          93614.802
Sample:                     03-10-2004   HQIC                         93595.932
                          - 04-04-2005
Covariance Type:                   opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.4973      0.004   -120.724      0.000      -0.505      -0.489
ar.L2         -0.2637      0.007    -36.600      0.000      -0.278      -0.250
ar.L3         -0.1195      0.010    -11.975      0.000      -0.139      -0.100
sigma2      1292.5334      5.528    233.823      0.000    1281.699    1303.368
===================================================================================
Ljung-Box (L1) (Q):                   0.83   Jarque-Bera (JB):            191749.55
Prob(Q):                              0.36   Prob(JB):                         0.00
Heteroskedasticity (H):               0.69   Skew:                            -2.37
Prob(H) (two-sided):                  0.00   Kurtosis:                        24.66
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

Step4: Forecast future values

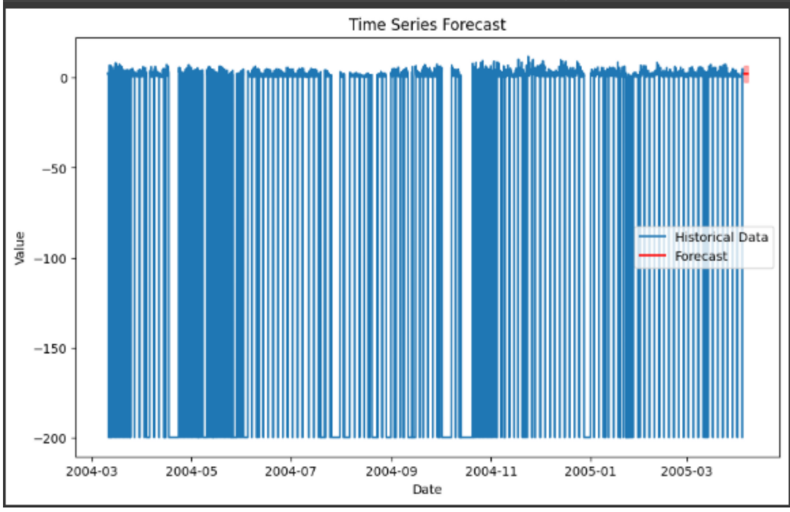
```
# Forecast future values
forecast_steps = 3  # Number of periods to forecast
forecast, stderr, conf_int = model_fit.forecast(steps=forecast_steps)
```

Step 5: Create a DataFrame for the forecast

```
# Create a DataFrame for the forecast
forecast_index = pd.date_range(start=series.index[-1] + pd.Timedelta(days=1), periods=forecast_steps, freq='D')
forecast_series = pd.Series(forecast, index=forecast_index)
```

Step 6: Plot the results

```
# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(series, label='Historical Data')
plt.plot(forecast_series, color='red', label='Forecast')
plt.fill_between(forecast_series.index, forecast_series - 1.96 * stderr, forecast_series + 1.96 * stderr, color='red', alpha=0.3)
plt.title('Time Series Forecast')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.show()
```



Students have to perform all the tasks illustrated above by choosing any other time series related dataset.

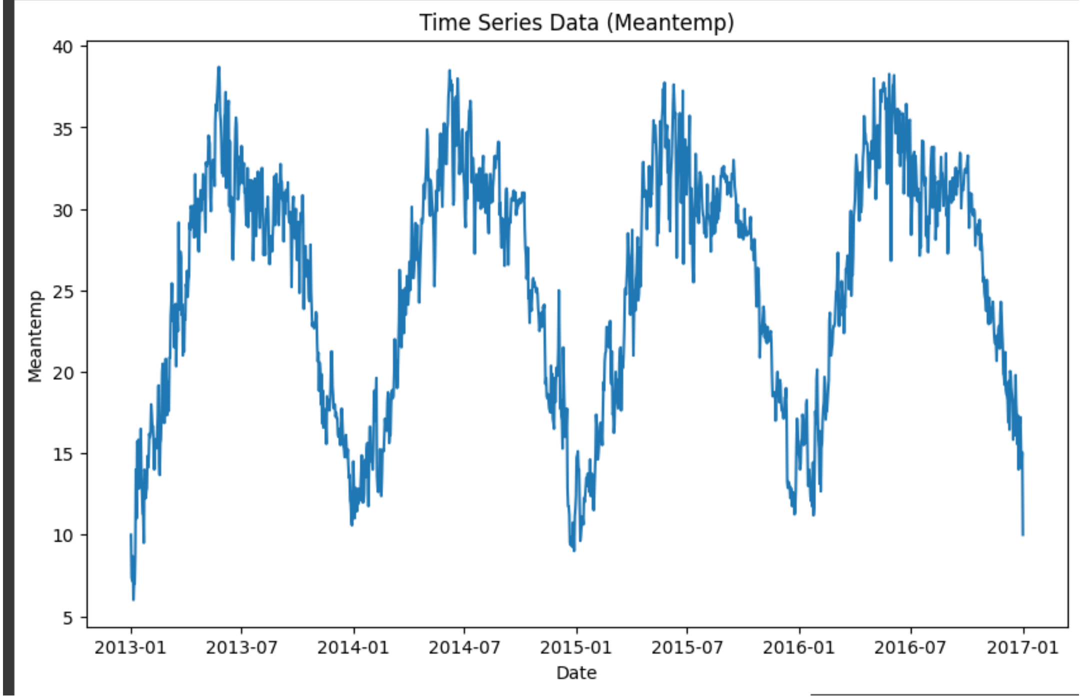
Implementation details:

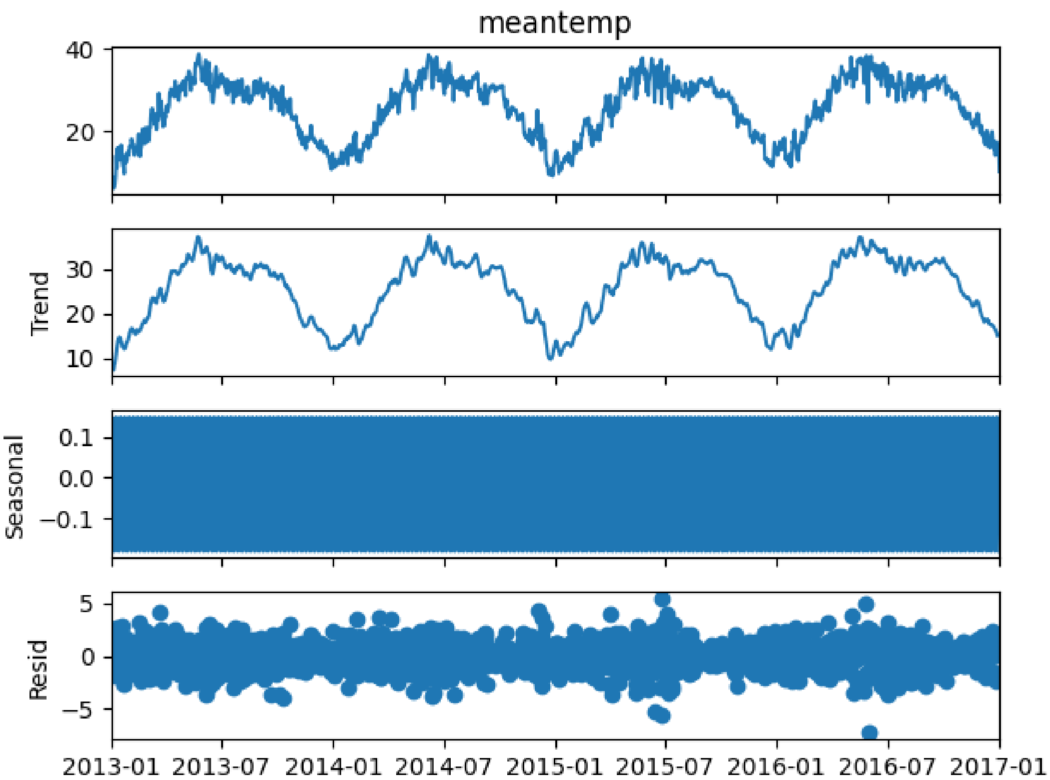
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.seasonal import seasonal_decompose
import kagglehub

# Load the dataset
file_path = "DailyDelhiClimateTrain.csv"
download_path = kagglehub.dataset_download(
  "sumanthvrao/daily-climate-time-series-data"
)
full_file_path = f"{download_path}/{file_path}"

data = pd.read_csv(full_file_path)


# Combine 'Date' and 'Time' into a single 'datetime' column
data['datetime'] = pd.to_datetime(data['date'])

# Set the 'datetime' column as the index
data.set_index('datetime', inplace=True)

# Drop the original 'Date' and 'Time' columns as they are no longer needed
data.drop(columns=['date'], inplace=True)

# Optional: Convert columns to numeric if they are not already
data = data.apply(pd.to_numeric, errors='coerce')

# Display the first few rows of the dataset
print(data.head())

# Adjust column names as needed - using 'meantemp' as an example
series = data['meantemp']

# Visualize the data
plt.figure(figsize=(10, 6))
plt.plot(series)
```

```
plt.title('Time Series Data (Meantemp)')
plt.xlabel('Date')
plt.ylabel('Meantemp')
plt.show()

# Decompose the data
decomposition = seasonal_decompose(series, model='additive')
fig = decomposition.plot()
plt.show()

# Fit ARIMA model
model = ARIMA(series, order=(5, 1, 0))
model_fit = model.fit()

# Summary of the model
print(model_fit.summary())

# Forecast future values
forecast_steps = 30
forecast_series = model_fit.forecast(steps=forecast_steps)
# Plot the results
plt.figure(figsize=(10, 6))
plt.plot(series, label='Historical Data')
plt.plot(forecast_series, color='red', label='Forecast')
plt.title('Time Series Forecast (Meantemp)')
plt.xlabel('Date')
plt.ylabel('Meantemp')
plt.legend()
plt.show()
```

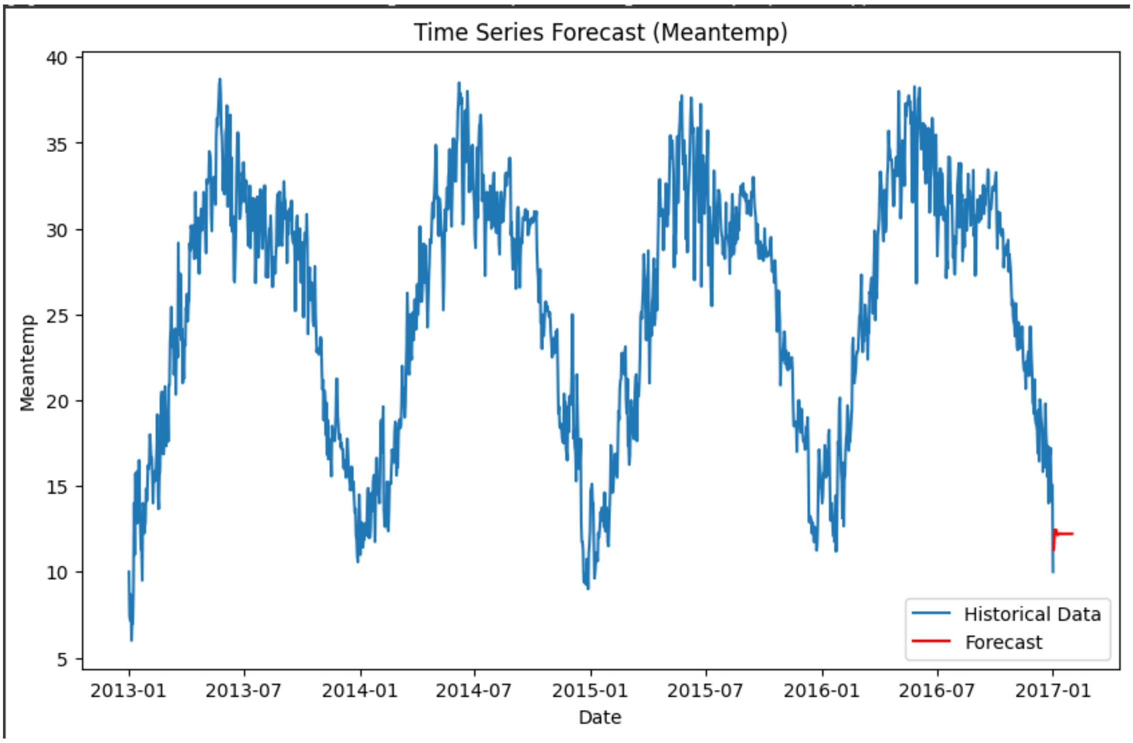
Output:

datetime	meantemp	humidity	wind_speed	meanpressure
2013-01-01	10.000000	84.500000	0.000000	1015.666667
2013-01-02	7.400000	92.000000	2.980000	1017.800000
2013-01-03	7.166667	87.000000	4.633333	1018.666667
2013-01-04	8.666667	71.333333	1.233333	1017.166667
2013-01-05	6.000000	86.833333	3.700000	1016.500000





```
                               SARIMAX Results
==============================================================================
Dep. Variable:               meantemp   No. Observations:                 1462
Model:                 ARIMA(5, 1, 0)   Log Likelihood               -2770.149
Date:                Mon, 13 Oct 2025   AIC                           5552.297
Time:                        04:21:26   BIC                           5584.019
Sample:                    01-01-2013   HQIC                          5564.130
                         - 01-01-2017
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.2122      0.021     -9.896      0.000      -0.254      -0.170
ar.L2         -0.1524      0.024     -6.317      0.000      -0.200      -0.105
ar.L3         -0.1827      0.025     -7.229      0.000      -0.232      -0.133
ar.L4         -0.0950      0.024     -3.902      0.000      -0.143      -0.047
ar.L5         -0.0667      0.024     -2.837      0.005      -0.113      -0.021
sigma2         2.5964      0.072     35.833      0.000       2.454       2.738
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):               277.34
Prob(Q):                              0.94   Prob(JB):                         0.00
Heteroskedasticity (H):               0.79   Skew:                            -0.48
Prob(H) (two-sided):                  0.01   Kurtosis:                         4.91
===================================================================================
```



Date: 13/10/25 **Signature of faculty in-charge**

Post Lab Descriptive Questions:

1. **What are the key components of a time series, and how do they affect the analysis?**
   The key components of a time series are trend, seasonality, and residual (or noise). The trend reflects the long-term direction or pattern in the data, seasonality captures regular, repeating fluctuations over fixed periods, and residuals represent random, unpredictable variations. Understanding these components is crucial because they help analysts identify underlying patterns, separate systematic behavior from randomness, and build more accurate forecasting models. Ignoring any component can lead to misleading conclusions or poor predictions.

2. **What is the purpose of decomposing a time series into trend, seasonal, and residual components?**
   The purpose of decomposing a time series into trend, seasonal, and residual components is to break down the data into simpler parts to better understand its underlying patterns. By isolating the trend, we can see the overall direction, while separating the seasonal component reveals repeating cycles. The residual captures random noise or irregularities. This decomposition helps improve forecasting accuracy, makes it easier to identify anomalies, and allows for targeted modeling of each component.

3. **Explain how the ARIMA model works and what the terms (p, d, q) represent.**
   The ARIMA model forecasts time series data by combining three elements: autoregression (AR), differencing (I), and moving average (MA). Autoregression (p) uses past values of the series to predict current values. Differencing (d) involves subtracting previous observations to make the series stationary by removing trends or seasonality. Moving average (q) incorporates past forecast errors to refine predictions. Together, the parameters ppp, ddd, and qqq specify the number of lagged observations, the number of differences applied, and the size of the moving average window, enabling the model to capture various patterns in the data for better forecasting.