K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch :- D-2 **Roll No. :-** 16010122151

Experiment :- 05

TITLE : To perform forecasting using time series analysis

AIM: To perform forecasting using time series analysis

Expected OUTCOME of Experiment:

CO4: Perform Time series Analytics and forecasting

Books/ Journals/ Websites referred:

Students have to list.

Pre Lab/ Prior Concepts:

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

Procedure:

Data set Used: Temperature data

Step1: Select and Load the dataset

(Students should write the code and output)

```
df=pd.read_csv('AirQualityUCIfinal.csv',index_col='Date',parse_date
s=True)
df=df.dropna()
print('Shape of data',df.shape)
df.head()
```

```
pip install pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (7.8 kB)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.4.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.11)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.26.4)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.2)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.13.1)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.4)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.2.3)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (71.0.4)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (24.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.5.0)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.6->statsmodels>=0.13.2->pmdarima) (1.16.0)
Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 18.6 MB/s eta 0:00:00
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
```

```
[2] import pandas as pd
    import numpy as np
```

```
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)
df = data
df=df.dropna()
print('Shape of data',df.shape)
df.head()
```

Shape of data (9357, 13)

datetime	CO(GT)	PT08.S1(CO)	NMHC(GT)	C6H6(GT)	PT08.S2(NMHC)	NOx(GT)	PT08.S3(NOx)	NO2(GT)	PT08.S4(NO2)	PT08.S5(O3)	T	RH	AH
2004-03-10 18:00:00	2.6	1360	150	11.9	1046	166	1056	113	1692	1268	13.6	48.9	0.7578
2004-03-10 19:00:00	2.0	1292	112	9.4	955	103	1174	92	1559	972	13.3	47.7	0.7255
2004-03-10 20:00:00	2.2	1402	88	9.0	939	131	1140	114	1555	1074	11.9	54.0	0.7502
2004-03-10 21:00:00	2.2	1376	80	9.2	948	172	1092	122	1584	1203	11.0	60.0	0.7867
2004-03-10 22:00:00	1.6	1272	51	6.5	836	131	1205	116	1490	1110	11.2	59.6	0.7888

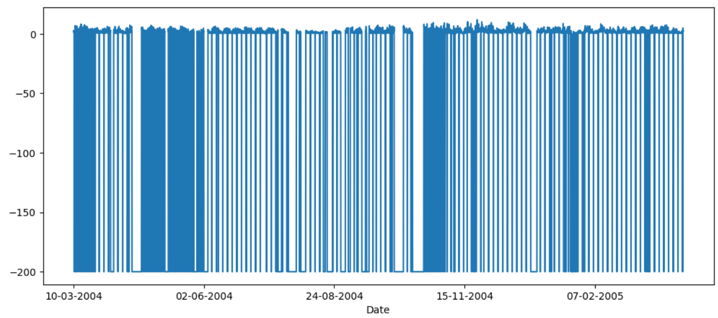
Step2: Visualize the data

(Students should write the code and output)

```
df['CO(GT)'].plot(figsize=(12,5))
```

```
df['CO(GT)'].plot(figsize=(12,5))
```

```
<Axes: xlabel='Date'>
```



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

(Students should write the code and output)

```
[15] from statsmodels.tsa.stattools import adfuller

     def adf_test(dataset):
       dftest = adfuller(dataset, autolag = 'AIC')
       print("1. ADF : ",dftest[0])
       print("2. P-Value : ", dftest[1])
       print("3. Num Of Lags : ", dftest[2])
       print("4. Num Of Observations Used For ADF Regression and Critical Values Calculation :", dftest[3])
       print("5. Critical Values :")
       for key, val in dftest[4].items():
           print("\t",key, ": ", val)
```

```
adf_test(df['CO(GT)'])
```

```
1. ADF :  -9.552224712392979
2. P-Value :  2.553762126201297e-16
3. Num Of Lags :  28
4. Num Of Observations Used For ADF Regression and Critical Values Calculation : 9328
5. Critical Values :
	 1% :  -3.43105123289464
	 5% :  -2.861849900767839
	 10% :  -2.566934955096094
```

```
stepwise_fit = auto_arima(df['CO(GT)'],
                          suppress_warnings=True)

stepwise_fit.summary()
```

SARIMAX Results

Dep. Variable:	y	**No. Observations:**	9357
Model:	SARIMAX(3, 1, 2)	**Log Likelihood**	-46701.382
Date:	Thu, 10 Oct 2024	**AIC**	93416.764
Time:	11:40:35	**BIC**	93466.770
Sample:	03-10-2004 - 04-04-2005	**HQIC**	93433.747
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-0.0005	0.006	-0.083	0.934	-0.012	0.011
ar.L1	1.0042	0.044	22.732	0.000	0.918	1.091
ar.L2	-0.0461	0.020	-2.365	0.018	-0.084	-0.008
ar.L3	-0.0054	0.019	-0.290	0.772	-0.042	0.031
ma.L1	-1.5276	0.045	-34.298	0.000	-1.615	-1.440
ma.L2	0.5376	0.042	12.868	0.000	0.456	0.620
sigma2	1298.2335	9.829	132.086	0.000	1278.970	1317.497

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	191365.77
Prob(Q):	0.92	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.68	**Skew:**	-2.54
Prob(H) (two-sided):	0.00	**Kurtosis:**	24.57

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

```python
from statsmodels.tsa.arima.model import ARIMA

# Fitting the ARIMA model
model = ARIMA(train['CO(GT)'], order=(1, 0, 5))
model = model.fit()

# Display the model summary
model.summary()
```

SARIMAX Results

Dep. Variable:	CO(GT)	**No. Observations:**	9327
Model:	ARIMA(1, 0, 5)	**Log Likelihood**	-46528.131
Date:	Thu, 10 Oct 2024	**AIC**	93072.261
Time:	11:43:22	**BIC**	93129.386
Sample:	03-10-2004	**HQIC**	93091.665
	- 04-03-2005		
Covariance Type:	opg		

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
const	-34.3011	16.546	-2.073	0.038	-66.730	-1.872
ar.L1	0.9762	0.007	147.139	0.000	0.963	0.989
ma.L1	-0.5021	0.005	-102.595	0.000	-0.512	-0.492
ma.L2	-0.0305	0.008	-3.964	0.000	-0.046	-0.015
ma.L3	-0.0088	0.012	-0.752	0.452	-0.032	0.014
ma.L4	-0.0085	0.013	-0.651	0.515	-0.034	0.017
ma.L5	-0.0068	0.013	-0.522	0.602	-0.033	0.019
sigma2	1260.2427	17.611	71.561	0.000	1225.726	1294.759

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	183860.95
Prob(Q):	0.99	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.67	**Skew:**	-2.56
Prob(H) (two-sided):	0.00	**Kurtosis:**	24.14

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

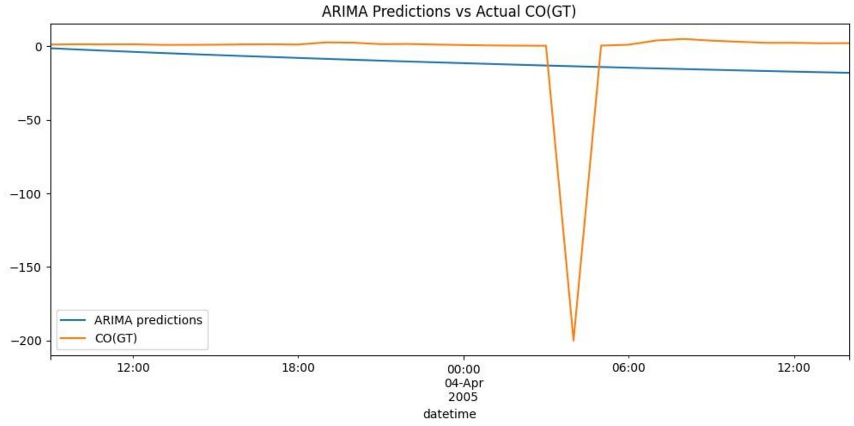
Step4: Forecast future values

(Students should write the code and output)

```python
# Determine the start and end points for predictions
start = len(train)
end = len(train) + len(test) - 1

# Predict the values using the ARIMA model
pred = model.predict(start=start, end=end, typ='levels').rename('ARIMA predictions')

# Align the predicted index with the test set's datetime index
pred.index = test.index  # Align with the test set's datetime index

# Print the predicted values
print(pred)

# Plot the ARIMA predictions vs the actual CO(GT) values from the test set
pred.plot(legend=True, figsize=(12, 5), title='ARIMA Predictions vs Actual CO(GT)')
test['CO(GT)'].plot(legend=True)
```

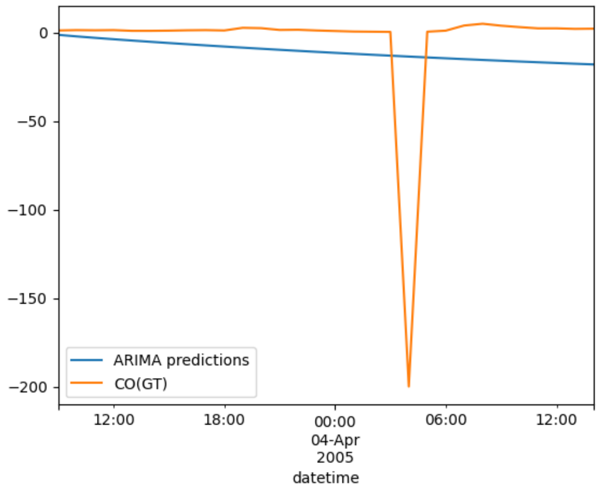
```
datetime
2005-04-03 09:00:00    -1.259783
2005-04-03 10:00:00    -2.153126
2005-04-03 11:00:00    -2.966880
2005-04-03 12:00:00    -3.745082
2005-04-03 13:00:00    -4.487613
2005-04-03 14:00:00    -5.198477
2005-04-03 15:00:00    -5.892391
2005-04-03 16:00:00    -6.569760
2005-04-03 17:00:00    -7.230978
2005-04-03 18:00:00    -7.876430
2005-04-03 19:00:00    -8.506492
2005-04-03 20:00:00    -9.121531
2005-04-03 21:00:00    -9.721906
2005-04-03 22:00:00   -10.307965
2005-04-03 23:00:00   -10.880050
2005-04-04 00:00:00   -11.438495
2005-04-04 01:00:00   -11.983624
2005-04-04 02:00:00   -12.515755
2005-04-04 03:00:00   -13.035199
2005-04-04 04:00:00   -13.542257
2005-04-04 05:00:00   -14.037224
2005-04-04 06:00:00   -14.520390
2005-04-04 07:00:00   -14.992036
2005-04-04 08:00:00   -15.452435
```

<Axes: title={'center': 'ARIMA Predictions vs Actual CO(GT)'}, xlabel='datetime'>




```python
pred.plot(legend='ARIMA Predictions')
test['CO(GT)'].plot(legend=True)
```

<Axes: xlabel='datetime'>



```
model2=ARIMA(df['AvgTemp'],order=(1,0,5))
model2=model2.fit()
df.tail()
```

DATE	MinTemp	MaxTemp	AvgTemp	Sunrise	Sunset
2018-12-26	35.0	45.0	40.0	654	1752
2018-12-27	33.0	44.0	39.0	655	1752
2018-12-28	33.0	47.0	40.0	655	1753
2018-12-29	36.0	47.0	42.0	655	1753
2018-12-30	39.0	52.0	46.0	656	1754

For Future Dates

```
index_future_dates=pd.date_range(start='2018-12-30',end='2019-01-29')
#print(index_future_dates)
pred=model2.predict(start=len(df),end=len(df)+30,typ='levels').rename('ARIMA Predictions')
#print(comp_pred)
pred.index=index_future_dates
print(pred)
```

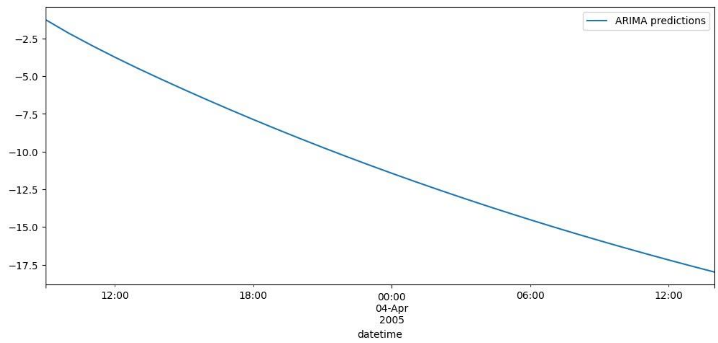
```
2018-12-30    46.418166
2018-12-31    46.113912
2019-01-01    45.617874
2019-01-02    45.249566
```

Step 5: Create a DataFrame for the forecast

(Students should write the code and output)

```
pred.plot(figsize=(12,5),legend=True)
```

```
<Axes: xlabel='datetime'>
```



Step 6: Plot the results

(Students should write the code and output)

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

Air Quality Data set shared.

Date:______________ **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?

A time series is composed of several key components that affect analysis:

- **Trend**: This represents the long-term movement or direction in the data (upward, downward, or constant). It affects the analysis by indicating the general direction in which the data is moving over time.
  - Example: A steady increase in sales over several years.
- **Seasonality**: These are recurring patterns or cycles that occur at regular intervals, usually due to seasonal effects (daily, weekly, monthly, yearly). Seasonality affects how predictions are made, as models need to account for these repeating patterns.
  - Example: Increased air conditioning sales every summer.
- **Cyclic Patterns**: These refer to long-term oscillations that are not tied to seasonality but occur over periods longer than a year. They are influenced by economic or environmental factors. Cycles affect analysis by indicating broader patterns beyond short-term changes.
  - Example: Economic growth and recession periods.
- **Noise/Residuals**: This is random variability in the data that cannot be explained by trend, seasonality, or cycles. Residuals affect the analysis because high noise levels can reduce the model's ability to make accurate predictions.
  - Example: Sudden, unpredictable spikes or drops in stock prices.

Understanding these components helps in selecting the right model and preprocessing data effectively.

2. What is the purpose of decomposing a time series into trend, seasonal, and residual components?

Decomposing a time series into **trend**, **seasonal**, and **residual** components helps in better understanding and modeling the data. Here's why:

- **Trend**: Extracting the trend helps in identifying the long-term movement in the data. Once the trend is isolated, it becomes easier to focus on the underlying behavior without short-term fluctuations.

- **Seasonality**: Decomposing seasonality helps in identifying repetitive patterns. This is crucial for models that need to capture these cyclical behaviors, especially for data that follows yearly, monthly, or weekly cycles.
- **Residuals**: Decomposing the residuals helps in understanding the randomness or noise in the data. By examining the residuals after removing trend and seasonality, we can better assess how much of the data is unpredictable or random.

In essence, decomposing the time series enables more accurate forecasting by allowing you to model and account for different components separately.

3. Explain how the ARIMA model works and what the terms (p, d, q) represent.

The **ARIMA (AutoRegressive Integrated Moving Average)** model is a popular forecasting technique for time series data. It works by combining three main components:

1. **AR (AutoRegressive)**: This component models the relationship between the current value and its previous values (lags). It captures the dependency between an observation and a certain number of lagged observations.
   - **p**: The number of lag observations included in the model (order of the autoregressive part).
2. **I (Integrated)**: This component refers to differencing the data to make it stationary (i.e., removing trends and seasonality). Differencing helps to stabilize the mean of a time series by removing changes in the level of the series.
   - **d**: The number of differencing steps required to make the series stationary.
3. **MA (Moving Average)**: This component models the relationship between an observation and a residual error from previous time steps. It helps smooth out noise in the time series by averaging past errors.
   - **q**: The number of lagged forecast errors used in the model (order of the moving average part).

The ARIMA model is expressed as **ARIMA(p, d, q)**, where:

- **p**: The number of autoregressive terms (AR).
- **d**: The number of differences to make the series stationary (I).
- **q**: The number of moving average terms (MA).

How it works:

1. **Model selection**: Choose values for `p`, `d`, and `q` based on autocorrelation, partial autocorrelation plots, or cross-validation.

2. **Fitting**: The model is fitted to the training data to estimate the coefficients.
3. **Prediction**: The model generates predictions by combining past values (AR part) and errors (MA part), adjusted for stationarity (I part).

By tuning `p`, `d`, and `q`, ARIMA can handle a variety of time series patterns, such as trends and noise.