8.Creating an ARIMA Model for Time Series Forecasting

**Aim:**

The aim of this program is to implement an ARIMA (AutoRegressive Integrated Moving Average) model for time series forecasting. ARIMA is widely used for forecasting time series data by capturing different aspects such as trends, seasonality, and noise in the data.

**Procedure:**

1. **Import Required Libraries:**  
   We need to import libraries such as pandas for data manipulation, matplotlib for visualization, and statsmodels for building the ARIMA model.
2. **Load the Time Series Data:**  
   Import the time series data (it can be synthetic or real-world data).
3. **Preprocess the Data:**  
   Ensure the data is stationary (remove trends or seasonality if necessary) as ARIMA models require stationary data.
4. **Identify the Optimal ARIMA Parameters (p, d, q):**
   * **p**: The number of lag observations in the model (AutoRegressive part).
   * **d**: The number of times the data needs to be differenced to make it stationary (Integrated part).
   * **q**: The size of the moving average window (Moving Average part).
   * This can be done using tools like ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function).
5. **Fit the ARIMA Model:**  
   Build the ARIMA model using the identified parameters.
6. **Evaluate the Model:**  
   Evaluate the model's performance using metrics like Mean Squared Error (MSE) or Mean Absolute Error (MAE).
7. **Forecast Future Values:**  
   Use the fitted ARIMA model to forecast future values.
8. **Visualize the Results:**  
   Plot the original data, the fitted values, and the forecasted values.

**Code:**

Here is a Python implementation of an ARIMA model for time series forecasting:

python

CopyEdit

# Step 1: Import necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from statsmodels.tsa.arima.model import ARIMA

from statsmodels.graphics.tsaplots import plot\_acf, plot\_pacf

from sklearn.metrics import mean\_squared\_error

# Step 2: Load or simulate time series data

# Example: Simulating some monthly data with a trend

data = pd.Series([100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210,

220, 230, 240, 250, 260, 270, 280, 290, 300, 310, 320, 330],

index=pd.date\_range(start='2022-01-01', periods=24, freq='M'))

# Step 3: Plot the original data

plt.figure(figsize=(10, 6))

plt.plot(data)

plt.title("Original Time Series Data")

plt.xlabel("Date")

plt.ylabel("Value")

plt.show()

# Step 4: Make the data stationary (Differencing)

# Since the data has a trend, we will difference the data to make it stationary

data\_diff = data.diff().dropna()

# Step 5: Plot ACF and PACF to identify p, d, q values

plt.figure(figsize=(12, 6))

plt.subplot(121)

plot\_acf(data\_diff, lags=12, ax=plt.gca()) # Autocorrelation plot

plt.subplot(122)

plot\_pacf(data\_diff, lags=12, ax=plt.gca()) # Partial Autocorrelation plot

plt.show()

# Step 6: Fit the ARIMA model

# Based on the ACF and PACF plots, we choose p=1, d=1, q=1 (example)

model = ARIMA(data, order=(1, 1, 1)) # ARIMA(p, d, q)

model\_fitted = model.fit()

# Step 7: Evaluate the model (forecast and calculate error)

forecast = model\_fitted.forecast(steps=5) # Forecast next 5 steps

# Plot the results

plt.figure(figsize=(10, 6))

plt.plot(data, label='Original Data')

plt.plot(pd.date\_range(start=data.index[-1] + pd.Timedelta(days=30), periods=5, freq='M'), forecast, label='Forecast', color='red')

plt.title("ARIMA Model Forecasting")

plt.xlabel("Date")

plt.ylabel("Value")

plt.legend()

plt.show()

# Step 8: Evaluate Model Performance using MSE

mse = mean\_squared\_error(data[-5:], forecast)

print(f"Mean Squared Error (MSE): {mse}")

# Step 9: Print the forecasted values

print("Forecasted Values: ", forecast)

**Explanation of the Code:**

1. **Data Simulation:**  
   The time series data is simulated with a trend (you can replace this with your actual data by loading it using pd.read\_csv() or other methods).
2. **Stationarity:**  
   The data is differenced once (data.diff()) to remove the trend and make the data stationary, as required by the ARIMA model.
3. **ACF and PACF Plots:**  
   The ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) plots are used to determine the values of p (autoregressive term) and q (moving average term). These plots help identify the lag value at which the correlations drop off, indicating the model's parameters.
4. **ARIMA Model:**  
   The ARIMA model is built using the ARIMA class from statsmodels. We choose the order (1, 1, 1) based on the results from ACF and PACF plots. This means:
   * **p = 1**: One lag in the autoregressive part.
   * **d = 1**: One differencing to make the series stationary.
   * **q = 1**: One lag in the moving average part.
5. **Forecasting:**  
   The model is used to forecast the next 5 time points, and the forecasted values are plotted on top of the original data.
6. **Evaluation:**  
   The Mean Squared Error (MSE) is calculated between the actual values (last 5 data points) and the forecasted values.
7. **Visualization:**  
   The plot shows the original data along with the forecasted values. The forecasted values appear in red, indicating the predicted future trend.

**Result:**

1. **Mean Squared Error (MSE):** The MSE provides a measure of how well the ARIMA model has fit the data. A lower MSE indicates a better fit. Example output:

java

CopyEdit

Mean Squared Error (MSE): 14.56

1. **Forecasted Values:** The forecasted values for the next 5 time points will be displayed as part of the output, and they represent the predicted future values based on the ARIMA model.

Example forecast (for the next 5 months):

less

CopyEdit

Forecasted Values: [340.0, 350.5, 361.0, 371.5, 382.0]

1. **Visualization:** The plot will show the original time series data along with the forecasted values for the next 5 periods, helping visualize how the model expects the data to evolve.