**Create an ARIMA Model for time series forecasting**

**EX:No.8 DATE:12/04/25**

# AIM:

To Create an ARIMA Model for time series forecasting.

# ALGORITHM:

1. ADF Test – Checks if the PM2.5 time series is stationary using statistical significance.
2. Differencing – Transforms non-stationary data to stationary by subtracting consecutive values.
3. ARIMA Model Selection – Chooses ARIMA(p,d,q) model where p = autoregressive lags, d = differencing, q = moving average lags.
4. Model Training – Fits the ARIMA model to historical PM2.5 data using specified parameters.
5. Forecasting – Predicts future PM2.5 values for the next 30 days using the trained model.
6. Visualization – Plots actual vs forecasted PM2.5 levels to visualize model performance.

# Code:

# import pandas as pd

# import matplotlib.pyplot as plt

# from statsmodels.tsa.arima.model import ARIMA

# from statsmodels.tsa.stattools import adfuller

# from pandas.plotting import register\_matplotlib\_converters

# register\_matplotlib\_converters()

# df = pd.read\_csv('/content/Super\_Store\_data.csv', encoding='ISO-8859-1')

# print("Columns:\n", df.columns)

# print("\nOrder Date and Sales preview:\n", df[['Order Date', 'Sales']].head())

# df['Order Date'] = pd.to\_datetime(df['Order Date'])

# df.set\_index('Order Date', inplace=True)

# monthly\_sales = df['Sales'].resample('M').sum()

# plt.figure(figsize=(10, 4))

# monthly\_sales.plot(title='Monthly Sales')

# plt.ylabel('Sales')

# plt.show()

# print("\nPerforming Augmented Dickey-Fuller Test...")

# adf\_result = adfuller(monthly\_sales.dropna())

# print(f"ADF Statistic: {adf\_result[0]}")

# print(f"p-value: {adf\_result[1]}")

# if adf\_result[1] < 0.05:

# print("The series is stationary.")

# else:

# print("The series is non-stationary.")

# print("\nFitting ARIMA(1,1,1) model...")

# model = ARIMA(monthly\_sales, order=(1, 1, 1))

# model\_fit = model.fit()

# print(model\_fit.summary())

# forecast = model\_fit.forecast(steps=12)

# plt.figure(figsize=(10, 4))

# plt.plot(monthly\_sales, label='Historical Sales')

# plt.plot(forecast.index, forecast, label='Forecast', color='red')

# plt.title('12-Month Sales Forecast')

# plt.ylabel('Sales')

# plt.legend()

# plt.show()

# OUTPUT:

# 

SARIMAX Results

==============================================================================

Dep. Variable: Sales No. Observations: 48

Model: ARIMA(1, 1, 1) Log Likelihood -494.372

Date: Fri, 18 Apr 2025 AIC 994.745

Time: 06:38:38 BIC 1000.295

Sample: 01-31-2014 HQIC 996.833

- 12-31-2017

Covariance Type: opg

==============================================================================

coef std err z P>|z| [0.025 0.975]

------------------------------------------------------------------------------

ar.L1 0.2602 0.283 0.919 0.358 -0.295 0.815

ma.L1 -0.8855 0.162 -5.475 0.000 -1.202 -0.568

sigma2 9.203e+07 1.07e-09 8.61e+16 0.000 9.2e+07 9.2e+07

===================================================================================

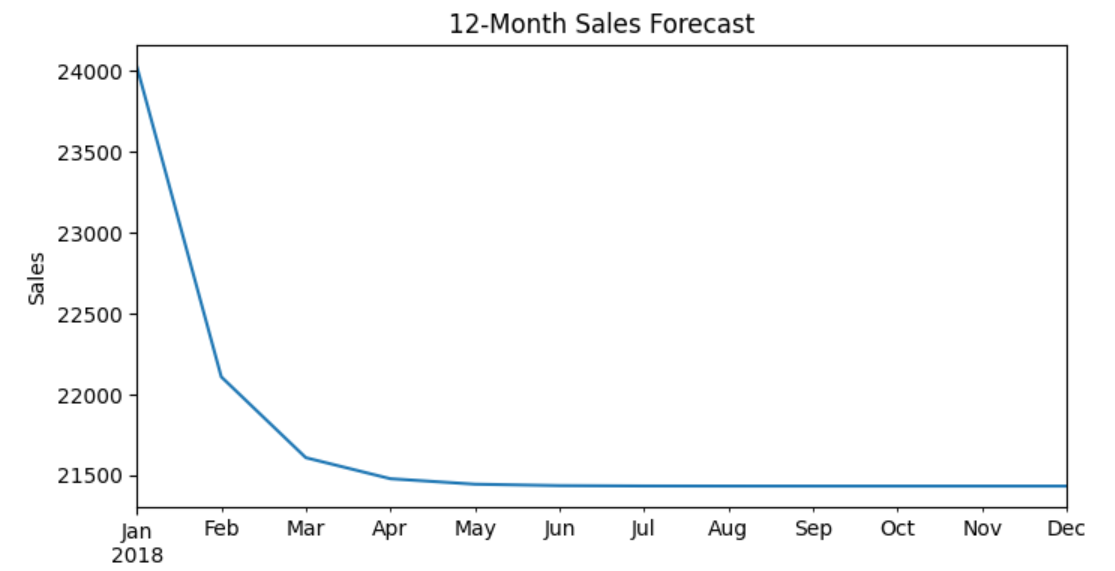
Ljung-Box (L1) (Q): 0.22 Jarque-Bera (JB): 1.13

Prob(Q): 0.64 Prob(JB): 0.57

Heteroskedasticity (H): 1.88 Skew: 0.32

Prob(H) (two-sided): 0.22 Kurtosis: 2.58

===================================================================================



# RESULT:

Thus, the program using the time series data implementation has been done successfully.