**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 pandas.plotting import register\_matplotlib\_converters

from statsmodels.tsa.stattools import adfuller

import seaborn as sns

register\_matplotlib\_converters()

# Step 1: Load the dataset

df = pd.read\_csv('/content/MSFT.csv', parse\_dates=['Date'])

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

# Step 2: Handle encoding issues

# Print the available columns to check for the correct name

print(df.columns)

df.columns = [col.encode('utf-8').decode('utf-8').replace("Â", "") for col in df.columns]

df = df.apply(pd.to\_numeric, errors='coerce') # convert all to numeric, force errors to NaN

# Step 3: Drop missing values

df = df.dropna()

# Use the correct column name identified from the print statement

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

# Replace 'PM2.5 (µg/m³)' with the actual column name from your CSV file

plt.plot(df['Close'], label='PM2.5')

plt.title('PM2.5 over time')

plt.legend()

plt.show()

# Step 5: Check stationarity using ADF test

# Replace 'PM2.5 (µg/m³)' with the actual column name

result = adfuller(df['Close'])

print('ADF Statistic:', result[0])

print('p-value:', result[1])

# Step 6: Differencing (if p-value > 0.05)

# Replace 'PM2.5 (µg/m³)' with the actual column name

df['PM2.5\_diff'] = df['Close'].diff().dropna()

# Step 7: Fit ARIMA model (you can tune p,d,q manually or use auto\_arima)

# Replace 'PM2.5 (µg/m³)' with the actual column name

model = ARIMA(df['Close'], order=(1,1,1)) # Example (p=1, d=1, q=1)

model\_fit = model.fit()

# Step 8: Summary

print(model\_fit.summary())

# Step 9: Forecast

forecast = model\_fit.forecast(steps=30) # Forecasting next 30 time points

# Step 10: Plot forecast

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

# Replace 'PM2.5 (µg/m³)' with the actual column name

plt.plot(df['Close'], label='Historical')

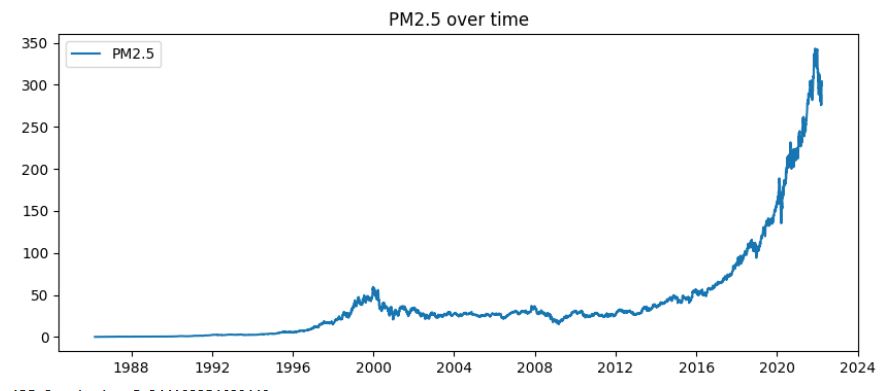
plt.plot(pd.date\_range(start=df.index[-1], periods=31, freq='D')[1:], forecast, label='Forecast', color='red')

plt.legend()

plt.title('PM2.5 Forecast')

plt.show()

# OUTPUT:

****

ADF Statistic: 5.344103354680442

p-value: 1.0

/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa\_model.py:471: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.

self.\_init\_dates(dates, freq)

/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa\_model.py:471: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.

self.\_init\_dates(dates, freq)

/usr/local/lib/python3.11/dist-packages/statsmodels/tsa/base/tsa\_model.py:471: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.

self.\_init\_dates(dates, freq)

SARIMAX Results

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

Dep. Variable: Close No. Observations: 9083

Model: ARIMA(1, 1, 1) Log Likelihood -15043.860

Date: Sat, 12 Apr 2025 AIC 30093.721

Time: 07:28:57 BIC 30115.063

Sample: 0 HQIC 30100.979

- 9083

Covariance Type: opg

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

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

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

ar.L1 -0.3316 0.015 -22.436 0.000 -0.361 -0.303

ma.L1 0.1709 0.016 10.963 0.000 0.140 0.201

sigma2 1.6080 0.005 299.855 0.000 1.598 1.619

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

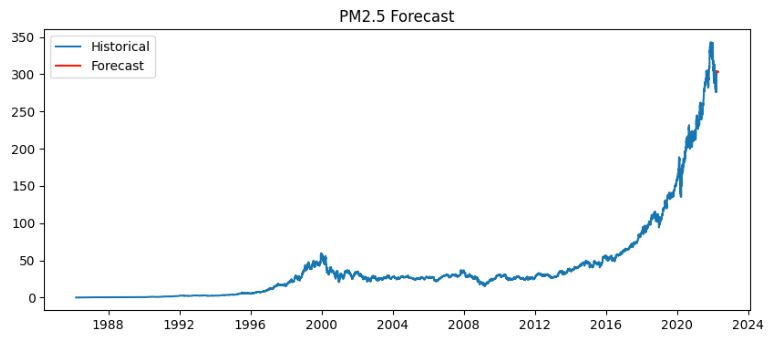
Ljung-Box (L1) (Q): 0.00 Jarque-Bera (JB): 657344.01

Prob(Q): 1.00 Prob(JB): 0.00

Heteroskedasticity (H): 303.96 Skew: -0.37

Prob(H) (two-sided): 0.00 Kurtosis: 44.67

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



# RESULT:

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