EX:No.8 221501060

12/04/25

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

**Aim:**

ToCreate an ARIMA model for time series forecasting.

**Algorithm:**

1. **Load the Data**:
   * Read the CSV file containing the weather data.
   * Parse the date column as a datetime index.
2. **Clean the Data**:
   * Handle missing values by performing forward and backward filling.
   * Drop any remaining NaN values.
3. **Normalize the Data**:
   * Apply **Min-Max Scaling** to normalize each column's values between 0 and 1.
4. **Add Time-Based Features**:
   * Extract additional features from the datetime index: day, month and year
5. **Visualize the Data**:
   * Plot the time series for a specific column (e.g., temperature T) over time.
6. **Execute the Program**:
   * Sequentially call the functions to load, clean, normalize, add features, and visualize the data.

**Code:**

import pandas as pd

import numpy as np

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

np.random.seed(42)

date\_range = pd.date\_range(start='2018-01-01', periods=60, freq='M')

energy\_consumption = np.random.normal(loc=300, scale=20, size=len(date\_range)).cumsum()

df = pd.DataFrame({'Date': date\_range, 'Energy\_Consumption': energy\_consumption})

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

df.plot(title='Simulated Energy Consumption')

plt.xlabel('Date')

plt.ylabel('Consumption (kWh)')

plt.show()

plot\_acf(df['Energy\_Consumption'], lags=20)

plot\_pacf(df['Energy\_Consumption'], lags=20)

plt.show()

train\_size = int(len(df) \* 0.8)

train, test = df.iloc[:train\_size], df.iloc[train\_size:]

model = ARIMA(train, order=(2, 1, 2)) # example values for (p,d,q)

model\_fit = model.fit()

forecast = model\_fit.forecast(steps=len(test))

test['Predicted'] = forecast.values

plt.figure(figsize=(10,5))

plt.plot(train.index, train['Energy\_Consumption'], label='Train')

plt.plot(test.index, test['Energy\_Consumption'], label='Actual')

plt.plot(test.index, test['Predicted'], label='Forecast')

plt.legend()

plt.title('ARIMA Forecast on Energy Consumption')

plt.xlabel('Date')

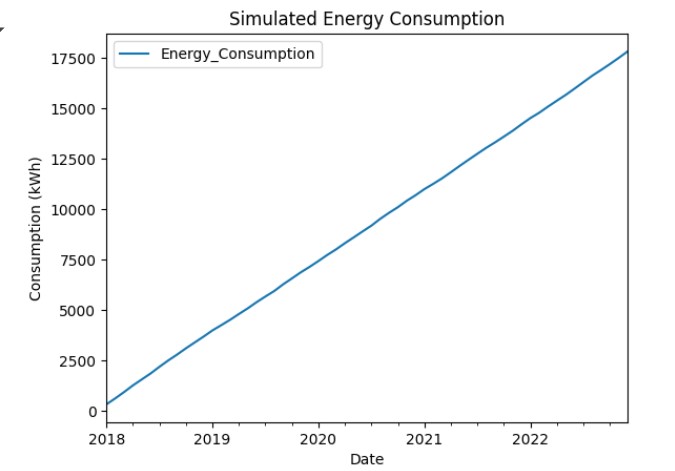
plt.ylabel('Consumption (kWh)')

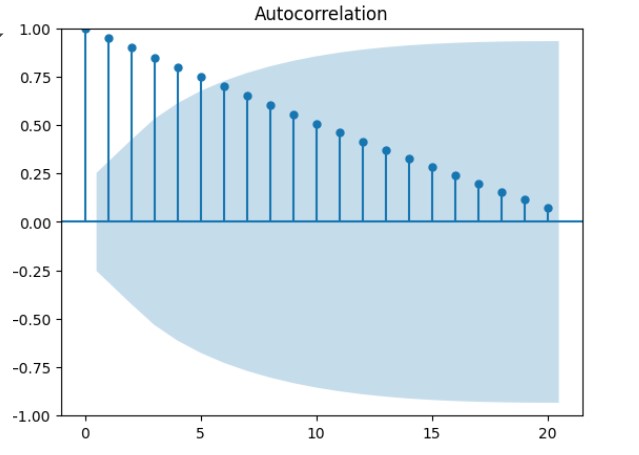
plt.show()

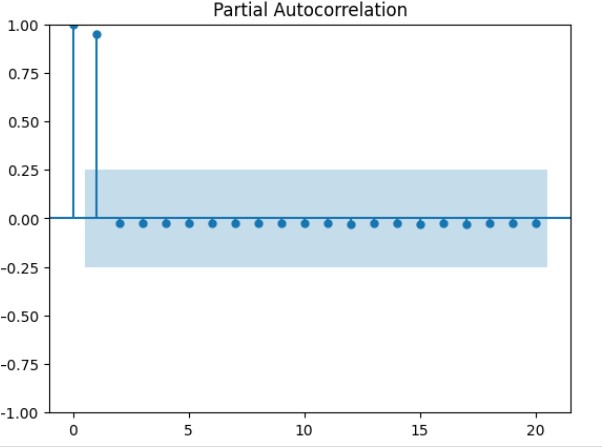
mse = mean\_squared\_error(test['Energy\_Consumption'], test['Predicted'])

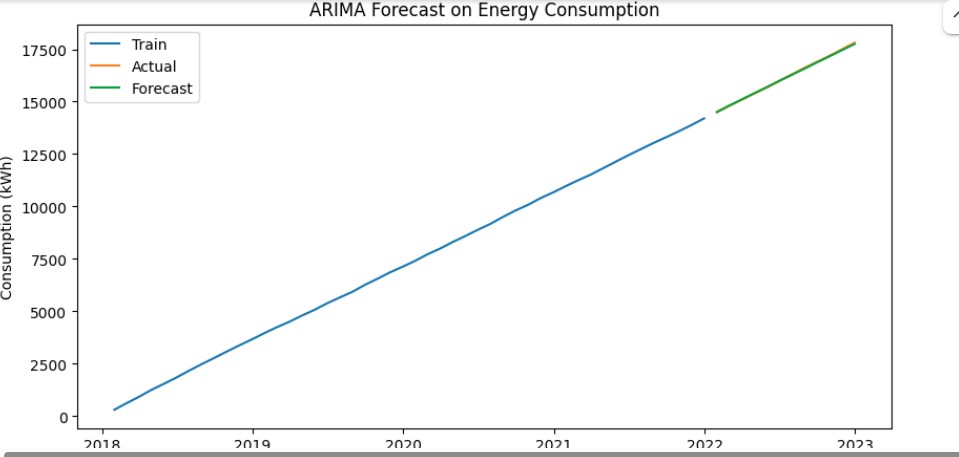
print(f"Mean Squared Error: {mse:.2f}")

**Output:**









**Result:**

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