EX:No.2 221501060

25/01/25

**Program to analyze and visualize stock trends using time series plots, moving averages, volume analysis, and daily returns.**

**Aim:**

Write a program to analyze and visualize stock trends using time series plots, moving averages, volume analysis, and daily returns.

**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

import seaborn as sns

from statsmodels.tsa.seasonal import seasonal\_decompose

date\_rng = pd.date\_range(start='2024-01-01', end='2024-03-01', freq='H')

np.random.seed(42)

energy\_usage = np.random.normal(loc=1.5, scale=0.5, size=len(date\_rng)) + np.linspace(0, 3, len(date\_rng))

df = pd.DataFrame({'timestamp': date\_rng, 'energy\_kWh': energy\_usage})

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

plt.figure(figsize=(14, 5))

df['energy\_kWh'].plot(title='Energy Consumption Over Time')

plt.xlabel('Timestamp')

plt.ylabel('Energy (kWh)')

plt.grid(True)

plt.show()

daily\_data = df.resample('D').mean()

plt.figure(figsize=(14, 5))

daily\_data.plot(title='Daily Average Energy Consumption')

plt.xlabel('Date')

plt.ylabel('Energy (kWh)')

plt.grid(True)

plt.show()

rolling\_mean = daily\_data.rolling(window=7).mean()

rolling\_std = daily\_data.rolling(window=7).std()

plt.figure(figsize=(14, 5))

plt.plot(daily\_data, label='Original')

plt.plot(rolling\_mean, label='7-day Mean', color='orange')

plt.plot(rolling\_std, label='7-day Std Dev', color='green')

plt.legend()

plt.title('Rolling Mean and Standard Deviation')

plt.grid(True)

plt.show()

decompose\_result = seasonal\_decompose(daily\_data['energy\_kWh'], model='additive', period=7)

decompose\_result.plot()

plt.suptitle('Time Series Decomposition (7-day)', fontsize=16)

plt.show()

df['hour'] = df.index.hour

plt.figure(figsize=(12, 5))

sns.boxplot(x='hour', y='energy\_kWh', data=df)

plt.title('Energy Consumption by Hour of Day')

plt.xlabel('Hour')

plt.ylabel('Energy (kWh)')

plt.grid(True)

plt.show()

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

df['energy\_kWh'].hist(bins=30, color='skyblue', edgecolor='black')

plt.title('Histogram of Energy Consumption')

plt.xlabel('Energy (kWh)')

plt.ylabel('Frequency')

plt.grid(True)

plt.show()

df['date'] = df.index.date

hourly\_avg = df.pivot\_table(index='date', columns='hour', values='energy\_kWh')

plt.figure(figsize=(14, 6))

sns.heatmap(hourly\_avg, cmap='YlGnBu')

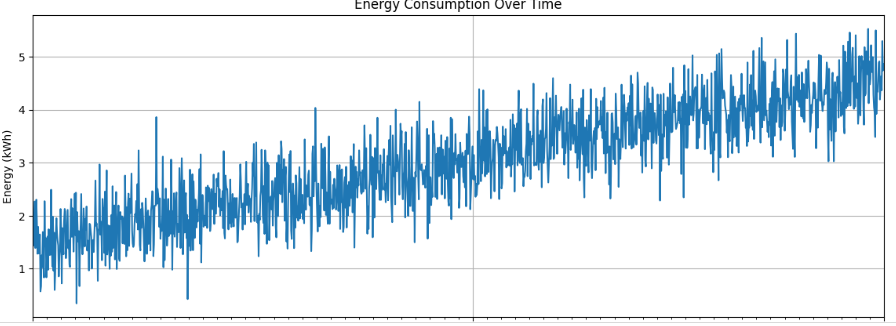
plt.title('Hourly Average Energy Consumption (Heatmap)')

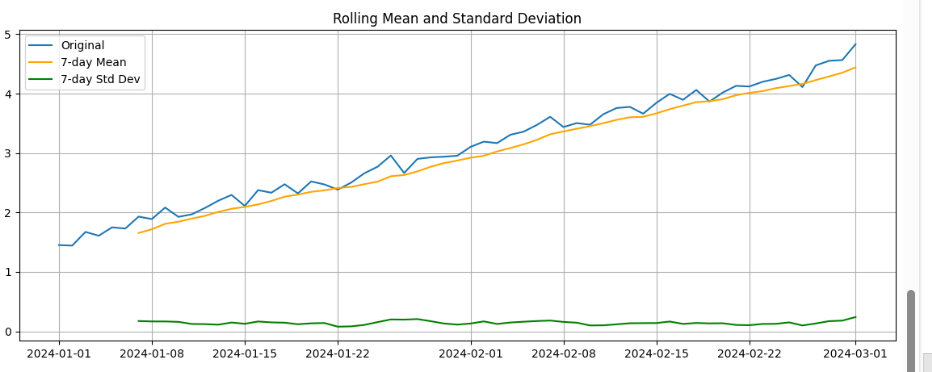
plt.xlabel('Hour of Day')

plt.ylabel('Date')

plt.show()

**Output:**

****

****

**Result:**

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