EX:No.2 221501042

1/02/25

**Implement programs for visualizing time series data.**

**Aim:**

Write a program to implement time series data for import library, load data, Preprocessing and visualising.

**Alogrithm:**

1. Start by loading the dataset from a CSV file and parse the 'date' column as a datetime index.
2. Clean the data by filling missing values using forward fill and backward fill methods, and then drop any remaining NaN values.
3. Normalize the data using Min-Max Scaling to scale values between 0 and 1.
4. Extract additional time-based features such as hour of the day, day of the week, and month of the year from the datetime index.
5. Visualize the data by generating a histogram with a density curve for a selected feature using Seaborn’s histplot function.
6. Create a box plot grouped by a time interval such as month to identify trends, seasonality, and outliers.
7. Plot autocorrelation and partial autocorrelation functions to examine dependencies and remove indirect effects in time-series data.
8. Compute and display a correlation heatmap to visualize relationships between numerical features.
9. Generate a lag plot to detect patterns and relationships between consecutive values in time-series data.
10. Finally, run the program to load, clean, preprocess, analyze, and visualize the data using these techniques.

**Code:**

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from pandas.plotting import lag\_plot, autocorrelation\_plot

import statsmodels.api as sm

def load\_data():

"""Loads the dataset with datetime index."""

df = pd.read\_csv(r"C:\Users\harsh\Downloads\cleaned\_weather.csv", parse\_dates=['date'], index\_col='date')

return df

def clean\_data(df):

"""Handles missing values properly."""

df = df.ffill().bfill() # Forward and backward fill

df.dropna(inplace=True) # Ensure no NaN values remain

return df

def normalize\_data(df):

"""Normalizes the data using Min-Max Scaling."""

df = (df - df.min()) / (df.max() - df.min())

return df

def add\_features(df):

"""Adds useful time-based features."""

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

df['dayofweek'] = df.index.dayofweek

df['month'] = df.index.month

return df

def histogram\_density\_plot(df, column='T'):

"""Plots a histogram and density plot of a selected column."""

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

sns.histplot(df[column], bins=30, kde=True, color='blue')

plt.title(f'Histogram & Density Plot of {column}')

plt.xlabel(column)

plt.ylabel('Frequency')

plt.show()

def box\_plot\_by\_time(df, column='T', time\_period='month'):

"""Creates a box plot grouped by time intervals (month, day of the week, etc.)."""

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

sns.boxplot(x=df[time\_period], y=df[column], palette='coolwarm')

plt.title(f'Box Plot of {column} by {time\_period}')

plt.xlabel(time\_period.capitalize())

plt.ylabel(column)

plt.show()

def autocorrelation\_pacf\_plot(df, column='T'):

"""Plots autocorrelation and partial autocorrelation of a selected column."""

fig, ax = plt.subplots(1, 2, figsize=(15, 5))

sm.graphics.tsa.plot\_acf(df[column], lags=40, ax=ax[0])

sm.graphics.tsa.plot\_pacf(df[column], lags=40, ax=ax[1])

ax[0].set\_title(f'Autocorrelation of {column}')

ax[1].set\_title(f'Partial Autocorrelation of {column}')

plt.show()

def heatmap\_correlation(df):

"""Plots a heatmap to show correlations between numerical columns."""

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

sns.heatmap(df.corr(), annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)

plt.title('Feature Correlation Heatmap')

plt.show()

def lag\_plot\_visualization(df, column='T'):

"""Creates a lag plot to check for patterns over time."""

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

lag\_plot(df[column])

plt.title(f'Lag Plot of {column}')

plt.show()

def main():

df = load\_data()

df = clean\_data(df)

df = normalize\_data(df)

df = add\_features(df)

print(df.head()) # Display first few rows

# Visualizations

histogram\_density\_plot(df, 'T') # 1. Histogram & Density Plot

box\_plot\_by\_time(df, 'T', 'month') # 2. Box Plot by Month

autocorrelation\_pacf\_plot(df, 'T') # 3. ACF & PACF

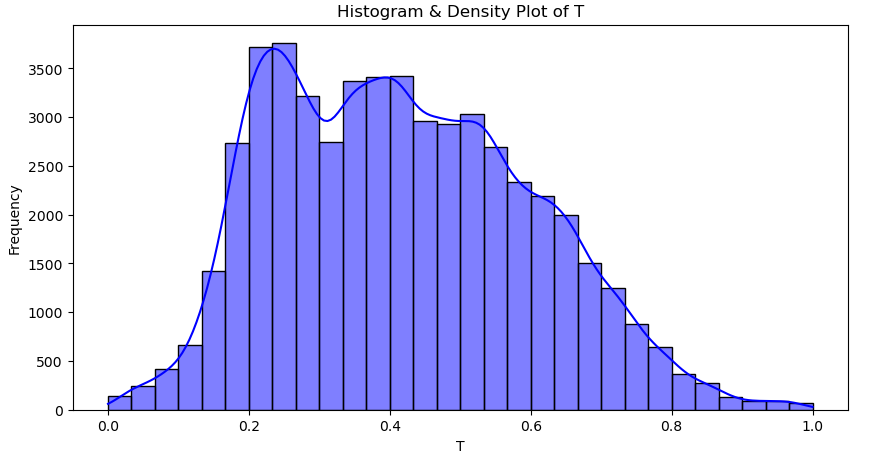
heatmap\_correlation(df) # 4. Correlation Heatmap

lag\_plot\_visualization(df, 'T') # 5. Lag Plot

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**



A graph of a box plot

Description automatically generated

A graph of a graph of a graph

Description automatically generated with medium confidence

A graph with numbers and a graph

Description automatically generated with medium confidence

A graph of a line

Description automatically generated

**Result:**

Thus, the program for visualization of time series data implementation has been done successfully.