EX:No.1 221501042

25/01/25

**Program to implement time series data for import library, load data,Preprocessing and visualising**

**Aim:**

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

**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: hour, dayofweek, and month.
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 necessary libraries

import pandas as pd

import matplotlib.pyplot as plt

# Function to load dataset with date as the index

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')

print("Using date as datetime index")

return df

# Function to clean dataset by handling missing values

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

print("Missing Values Handled!")

return df

# Function to normalize dataset using Min-Max Scaling

def normalize\_data(df):

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

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

print("Data Normalized!")

return df

# Function to add time-based features like hour, day, and month

def add\_features(df):

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

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

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

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

print("Features Added!")

return df

# Function to visualize data for a specific column

def visualize\_data(df, column='T'):

"""Plots a time-series visualization of the given column."""

if column not in df.columns:

print(f"Error: Column '{column}' not found in dataset. Available columns: {df.columns}")

return

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

plt.plot(df.index, df[column], label=column, color='blue')

plt.xlabel('Time')

plt.ylabel('Value')

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

plt.legend()

plt.show()

# Main function to execute the pipeline

def main():

df = load\_data()

df = clean\_data(df)

df = normalize\_data(df)

df = add\_features(df)

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

# Check if 'T' exists before visualization

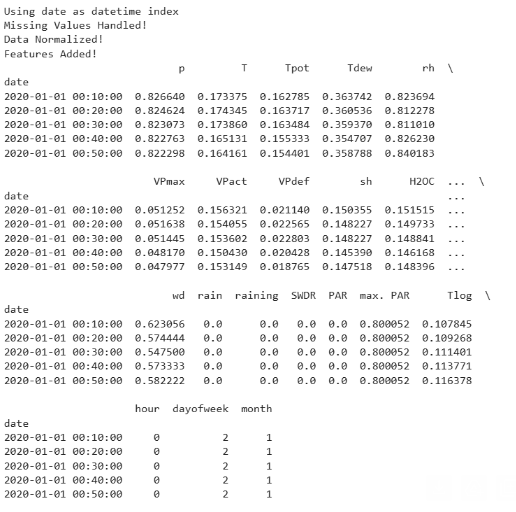
visualize\_data(df, 'T')

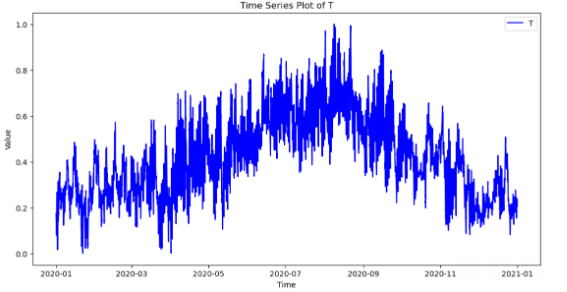
# Run the script

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

main()

**Output:**





**Result:**

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