EX:No.9 221501042

**Develop neural network-based time series forecasting model.**

**Aim:**

Write a program to develop neural network-based time series forecasting model.

**Algorithm:**

1.**Import required libraries**:  
Import numpy, pandas, matplotlib, MinMaxScaler from sklearn, and neural network layers from tensorflow.keras.

2.**Load the dataset**:  
Read the CSV file containing weather data and parse the 'Date' column as datetime while setting it as the index.

3.**Select the target column**:  
Extract the 'Temperature' column as the target for forecasting.

4.**Normalize the data**:  
Scale the temperature values to a range between 0 and 1 using MinMaxScaler to improve neural network performance.

5.**Create sequences**:  
Define a function that converts the scaled data into sequences of a specified number of time steps (TIME\_STEPS = 10) and corresponding labels (next temperature value).

6.**Split the data**:  
Divide the sequence data into training (80%) and testing (20%) sets.

7.**Build the LSTM model**:  
Create a sequential model with an LSTM layer (50 units, ReLU activation) followed by a dense output layer with one neuron. Compile the model using the Adam optimizer and mean squared error loss.

8.**Train the model**:  
Fit the model on the training data for 20 epochs.

9.**Make predictions**:  
Use the trained model to predict temperature values on the test set.

10.**Inverse scale the predictions**:  
Convert the scaled predictions and actual values back to the original scale using the inverse of MinMaxScaler.

11.**Visualize the results**:  
Plot the actual vs. predicted temperature values to evaluate model performance visually.

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, LSTM

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

data = df[['T']].values

scaler = MinMaxScaler(feature\_range=(0, 1))

data\_scaled = scaler.fit\_transform(data)

def create\_sequences(data, time\_steps=10):

X, y = [], []

for i in range(len(data) - time\_steps):

X.append(data[i:i + time\_steps])

y.append(data[i + time\_steps])

return np.array(X), np.array(y)

TIME\_STEPS = 10

X, y = create\_sequences(data\_scaled, TIME\_STEPS)

split = int(0.8 \* len(X))

X\_train, X\_test = X[:split], X[split:]

y\_train, y\_test = y[:split], y[split:]

model = Sequential()

model.add(LSTM(50, activation='relu', input\_shape=(TIME\_STEPS, 1)))

model.add(Dense(1))

model.compile(optimizer='adam', loss='mse')

model.fit(X\_train, y\_train, epochs=20, verbose=1)

predicted = model.predict(X\_test)

predicted\_unscaled = scaler.inverse\_transform(predicted)

y\_test\_unscaled = scaler.inverse\_transform(y\_test)

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

plt.plot(y\_test\_unscaled, label='Actual Temperature')

plt.plot(predicted\_unscaled, label='Predicted Temperature')

plt.title('Neural Network Time Series Forecasting')

plt.xlabel('Time')

plt.ylabel('Temperature')

plt.legend()

plt.grid(True)

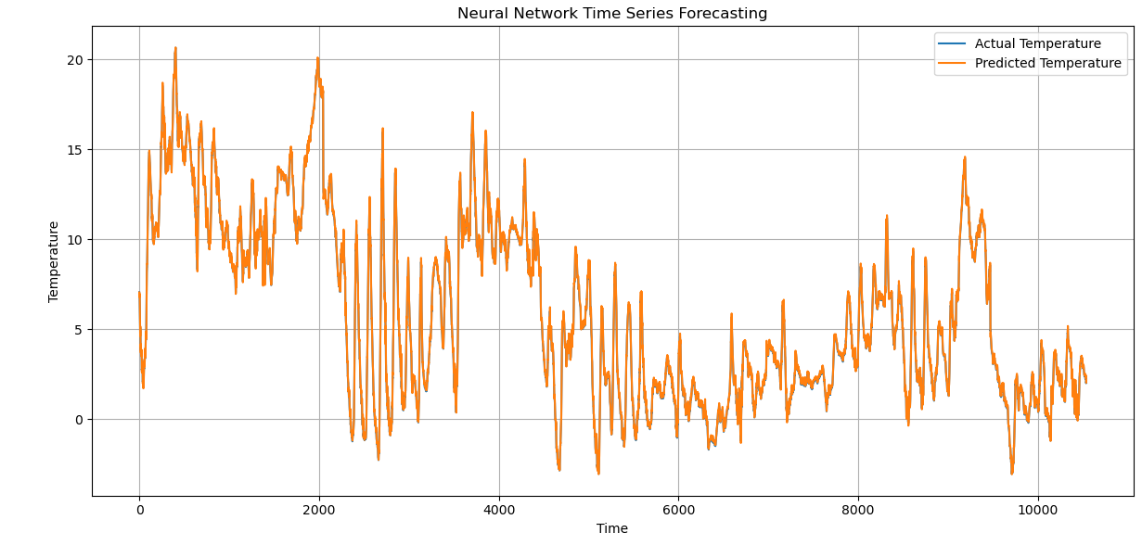
plt.tight\_layout()

plt.show()

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

****

**Result:**

Thus, the program to estimating & eliminating trend in time series data- aggregation, smoothing was done.