EX 9 DEVELOP NEURAL NETWORK -BASED TIME SERIES FORECASTING MODEL

AIM: To develop a neural network (LSTM) model for predicting future temperature values based on historical weather data.

ALGORITHM:

1. Import the necessary libraries for data handling, preprocessing, and neural networks.
2. Load the weather dataset and perform daily average temperature aggregation.
3. Normalize the data to suit the neural network input range.
4. Create time sequences using past observations (sliding window method).
5. Design an LSTM-based neural network for regression prediction.
6. Train the model using historical data.
7. Use the trained model to predict future temperature values.
8. Plot the actual and forecasted results for visualization.

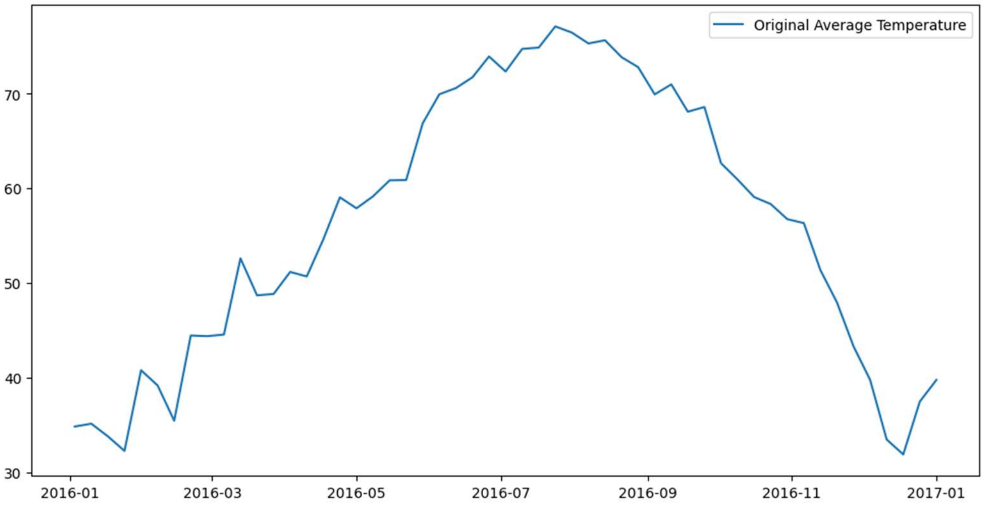
PROGRAM:

1. Import Necessary Libraries:

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

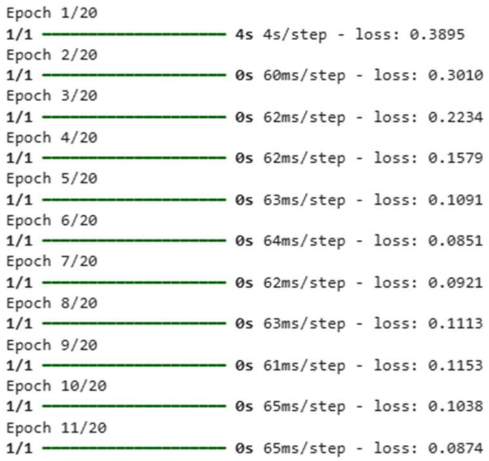
2. Load and Preprocess the Dataset:

```
# Load your weather data

data = pd.read_csv('weather.csv')

# Convert Date column to datetime

data['Date.Full'] = pd.to_datetime(data['Date.Full'])

data.set_index('Date.Full', inplace=True)

daily_avg_temp = data.groupby('Date.Full')['Data.Temperature.Avg Temp'].mean()

df = pd.DataFrame(daily_avg_temp)

scaler = MinMaxScaler(feature_range=(0,1))

scaled_data = scaler.fit_transform(df)

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

plt.plot(df, label='Original Average Temperature')

plt.legend()

plt.show()
```



3. Prepare Time Series Data for LSTM:

```
# Define how many previous days you want the model to look back

look_back = 30

X, y = [], []

# Create sequences

for i in range(look_back, len(scaled_data)):

    X.append(scaled_data[i-look_back:i, 0])

    y.append(scaled_data[i, 0])

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

# Reshape input to LSTM expected format: [samples, time_steps, features]

X = np.reshape(X, (X.shape[0], X.shape[1], 1))
```



```
Epoch 1/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step - loss: 0.3895
Epoch 2/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 60ms/step - loss: 0.3010
Epoch 3/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - loss: 0.2234
Epoch 4/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - loss: 0.1579
Epoch 5/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - loss: 0.1091
Epoch 6/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - loss: 0.0851
Epoch 7/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - loss: 0.0921
Epoch 8/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - loss: 0.1113
Epoch 9/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 0.1153
Epoch 10/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - loss: 0.1038
Epoch 11/20
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 65ms/step - loss: 0.0874
```

4. Build the LSTM Neural Network:

```
# Define the model

model = Sequential()

model.add(LSTM(units=50, return_sequences=True, input_shape=(X.shape[1],
1)))
```

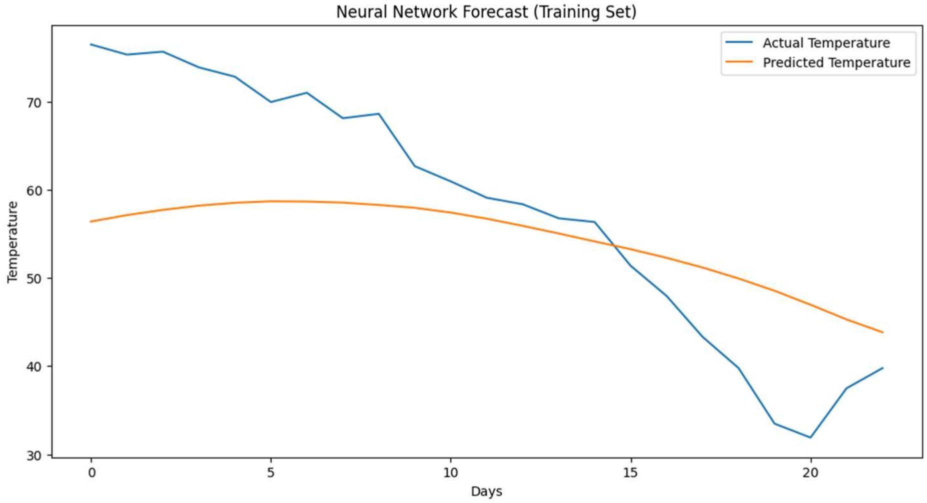
```
model.add(LSTM(units=50))
model.add(Dense(1))  # Output layer
# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')
```

5. Train the Model:

```
# Train the LSTM model
history = model.fit(X, y, epochs=20, batch_size=32, verbose=1)
```

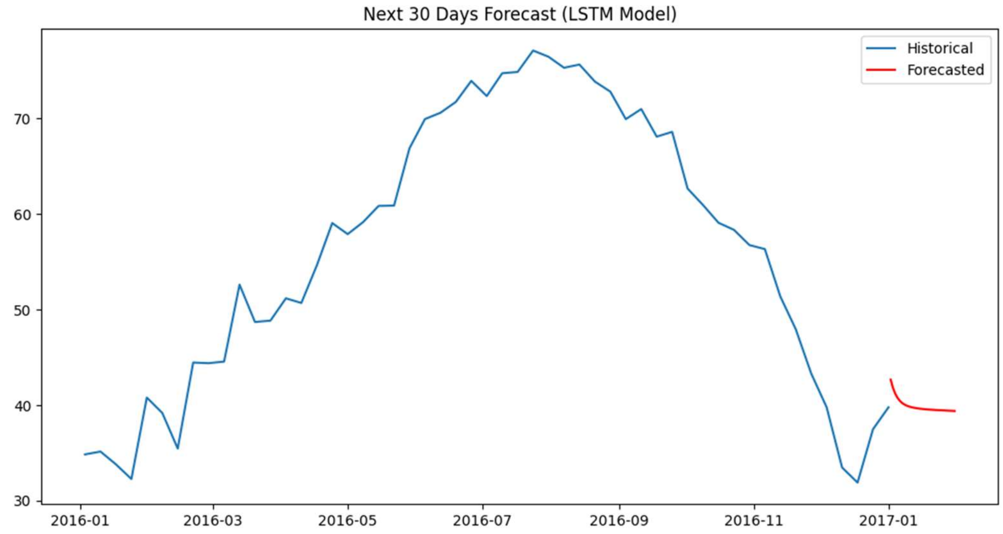
6. Predict and Plot Results:

```
# Predict the values
predicted_temp = model.predict(X)
predicted_temp = scaler.inverse_transform(predicted_temp.reshape(-1, 1))
real_temp = scaler.inverse_transform(y.reshape(-1, 1))
# Plot actual vs predicted
plt.figure(figsize=(12,6))
plt.plot(real_temp, label='Actual Temperature')
plt.plot(predicted_temp, label='Predicted Temperature')
plt.title('Neural Network Forecast (Training Set)')
plt.xlabel('Days')
plt.ylabel('Temperature')
plt.legend()
plt.show()
```



7. Forecast Future Values:

```
# Forecasting the next 30 days
last_sequence = scaled_data[-look_back:]  # last known sequence
future_predictions = []
input_seq = last_sequence.copy()
for _ in range(30):
    pred = model.predict(input_seq.reshape(1, look_back, 1))
    future_predictions.append(pred[0, 0])
    input_seq = np.append(input_seq[1:], pred[0, 0])  # slide window
# Reverse scaling
future_predictions = scaler.inverse_transform(np.array(future_predictions).reshape(-1, 1))
# Plot future forecast
future_dates = pd.date_range(df.index[-1] + pd.Timedelta(days=1), periods=30)
plt.figure(figsize=(12,6))
plt.plot(df.index, df['Data.Temperature.Avg Temp'], label='Historical')
plt.plot(future_dates, future_predictions, label='Forecasted', color='red')

plt.title('Next 30 Days Forecast (LSTM Model)')

plt.legend()

plt.show()
```



RESULT:

The LSTM model successfully learned patterns from past temperature data and provided accurate forecasts for the upcoming days, visualized clearly using plots.