|  |  |
| --- | --- |
| **EX.N0 : 9** | **Develop neural network-based time series forecasting model.** |
| **DATE : 27/03/2025** |

**AIM:**

# Develop neural network-based time series forecasting model.

# ALGORITHM:

# Step 1: Import Libraries Step 2: Load the Dataset Step 3: Preprocess the Data Step 4: Create Dataset for LSTM Model Step 5: Define and Train the Neural Network Step 6: Forecast Future Values Step 7: Plot Forecast

# PROGRAM:

# Step 1: Import 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 Dense, LSTM

from tensorflow.keras.optimizers import Adam

# Step 2: Load the Dataset

file\_path = '/mnt/data/climate\_change\_data.csv'

df = pd.read\_csv(file\_path)

# Step 3: Preprocess the Data

df.columns = [col.strip() for col in df.columns]

date\_col = df.columns[0]

df[date\_col] = pd.to\_datetime(df[date\_col], errors='coerce')

df.set\_index(date\_col, inplace=True)

# Select the first numeric column

df\_numeric = df.select\_dtypes(include='number')

value\_col = df\_numeric.columns[0]

data = df\_numeric[[value\_col]].dropna()

# Normalize the data

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(data)

# Step 4: Create Dataset for LSTM

def create\_sequences(data, window\_size):

x, y = [], []

for i in range(len(data) - window\_size):

x.append(data[i:i + window\_size])

y.append(data[i + window\_size])

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

window\_size = 12

X, y = create\_sequences(scaled\_data, window\_size)

# Step 5: Define and Train the Neural Network

model = Sequential()

model.add(LSTM(64, activation='relu', input\_shape=(window\_size, 1)))

model.add(Dense(1))

model.compile(optimizer=Adam(learning\_rate=0.01), loss='mse')

model.fit(X, y, epochs=100, verbose=0)

# Step 6: Forecast Future Values

last\_sequence = scaled\_data[-window\_size:]

predictions = []

for \_ in range(12): # Forecast next 12 months

input\_seq = last\_sequence.reshape((1, window\_size, 1))

next\_val = model.predict(input\_seq, verbose=0)[0]

predictions.append(next\_val)

last\_sequence = np.append(last\_sequence[1:], [next\_val], axis=0)

# Inverse transform predictions

forecast = scaler.inverse\_transform(predictions)

# Step 7: Plot Forecast

forecast\_index = pd.date\_range(start=data.index[-1], periods=13, freq='M')[1:]

forecast\_series = pd.Series(forecast.flatten(), index=forecast\_index)

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

plt.plot(data, label='Original')

plt.plot(forecast\_series, label='Neural Forecast', color='red', linestyle='--')

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

plt.grid(True)

plt.legend()

plt.show()

**OUTPUT:**

A blue line graph with white text

AI-generated content may be incorrect.

# RESULT:

Thus, the program for Implement programs for a time series data is executed successfully.