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

**EX:No.9**

**DATE: 14/04/25**

# AIM:

To implement a program for developing neural network-based time series forecasting model.

## ALGORITHM:

* Import required libraries: pandas, numpy, matplotlib, MinMaxScaler, and Keras modules (LSTM, Dense, Sequential)
* Load and preprocess the office supply sales dataset
* Convert the date column to datetime format and set it as the index
* Resample the sales data to daily frequency
* Normalize the sales values using MinMaxScaler
* Create supervised sequences using a sliding window of the past 10 days to predict the next day
* Split the data into training and testing sets (80% train, 20% test)
* Build an LSTM neural network model with one LSTM layer and one Dense output layer
* Compile and train the model using the training data
* Predict sales on the test set and inverse transform the predicted and actual values
* Plot and compare the actual vs predicted sales to visualize the model performance

**CODE:**

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

# Load the dataset

df = pd.read\_csv('cleaned\_sales\_data.csv', parse\_dates=['Order Date'])

df.set\_index('Order Date', inplace=True)

# Resample daily sales

df\_daily = df['Sales'].resample('D').sum().to\_frame()

# Drop missing values

df\_daily.dropna(inplace=True)

# Normalize the data

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(df\_daily)

# Convert to supervised learning format

X, y = [], []

for i in range(10, len(scaled\_data)):

X.append(scaled\_data[i-10:i])

y.append(scaled\_data[i])

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

# Split into training and testing sets

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

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

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

# Build the LSTM model

model = Sequential()

model.add(LSTM(50, activation='relu', input\_shape=(X.shape[1], 1)))

model.add(Dense(1))

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

# Train the model

model.fit(X\_train, y\_train, epochs=20, batch\_size=32, validation\_data=(X\_test, y\_test))

# Predict

pred = model.predict(X\_test)

# Inverse scale the results

pred\_inv = scaler.inverse\_transform(pred)

actual\_inv = scaler.inverse\_transform(y\_test)

# Plot actual vs predicted sales

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

plt.plot(actual\_inv, label='Actual Sales')

plt.plot(pred\_inv, label='Predicted Sales')

plt.title('Sales Forecasting using LSTM')

plt.xlabel('Time Step')

plt.ylabel('Sales')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

# OUTPUT:

# **Screenshot 2025-04-30 060838**

**RESULT:**

Thus, the program is implemented and verified successfully.