# **EX: 9 DEVELOP NEURAL NETWORK- BASED TIME DATE SERIES FORECASTING MODEL**

# **AIM:**

To develop a neural network-based time series forecasting model using LSTM.

**ALGORITHM:**

1. Import necessary libraries and load the airline passengers dataset.
2. Convert the date column to datetime format and set it as the index.
3. Normalize the passenger data using MinMaxScaler.
4. Create input-output pairs using a fixed look-back period for supervised learning.
5. Reshape the data to 3D format to match LSTM input requirements.
6. Split the dataset into training and testing sets.
7. Build and compile an LSTM model with one hidden layer and one output layer.
8. Train the model using the training data over multiple epochs.
9. Predict both training and test data, and inverse-transform the predictions.
10. Evaluate model performance using RMSE and visualize actual vs predicted values.

**PROGRAM:**

**# Step 1: Install necessary packages**

**!pip install pandas numpy matplotlib scikit-learn tensorflow --quiet**

**# Step 2: 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 sklearn.metrics import mean\_squared\_error**

**import math**

**# Step 3: Load Dataset**

**df = pd.read\_csv('/content/airline-passengers.csv')**

**print(df.head())**

**# Step 4: Preprocess Data**

**df['Month'] = pd.to\_datetime(df['Month'])**

**df.set\_index('Month', inplace=True)**

**data = df['Passengers'].values**

**data = data.reshape(-1, 1)**

**# Normalize the dataset**

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

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

**# Step 5: Create Dataset for Supervised Learning**

**def create\_dataset(dataset, look\_back=1):**

**X, Y = [], []**

**for i in range(len(dataset) - look\_back):**

**X.append(dataset[i:(i + look\_back), 0])**

**Y.append(dataset[i + look\_back, 0])**

**return np.array(X), np.array(Y)**

**look\_back = 10 # you can change this**

**X, y = create\_dataset(data, look\_back)**

**# Reshape input to [samples, time steps, features] for LSTM**

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

**# Step 6: Split into train/test sets**

**train\_size = int(len(X) \* 0.67)**

**X\_train, X\_test = X[:train\_size], X[train\_size:]**

**y\_train, y\_test = y[:train\_size], y[train\_size:]**

**# Step 7: Build the LSTM Model**

**model = Sequential()**

**model.add(LSTM(50, input\_shape=(look\_back, 1)))**

**model.add(Dense(1))**

**model.compile(loss='mean\_squared\_error', optimizer='adam')**

**# Step 8: Train the Model**

**model.fit(X\_train, y\_train, epochs=100, batch\_size=1, verbose=1)**

**# Step 9: Predict**

**train\_predict = model.predict(X\_train)**

**test\_predict = model.predict(X\_test)**

**# Inverse transform predictions**

**train\_predict = scaler.inverse\_transform(train\_predict)**

**y\_train\_inv = scaler.inverse\_transform([y\_train])**

**test\_predict = scaler.inverse\_transform(test\_predict)**

**y\_test\_inv = scaler.inverse\_transform([y\_test])**

**# Step 10: Calculate RMSE**

**train\_score = math.sqrt(mean\_squared\_error(y\_train\_inv[0], train\_predict[:, 0]))**

**test\_score = math.sqrt(mean\_squared\_error(y\_test\_inv[0], test\_predict[:, 0]))**

**print(f'Train RMSE: {train\_score:.2f}')**

**print(f'Test RMSE: {test\_score:.2f}')**

**# Step 11: Visualize Results**

**train\_plot = np.empty\_like(data)**

**train\_plot[:, :] = np.nan**

**train\_plot[look\_back:look\_back + len(train\_predict)] = train\_predict**

**test\_plot = np.empty\_like(data)**

**test\_plot[:, :] = np.nan**

**test\_plot[look\_back + len(train\_predict):] = test\_predict**

**# Plot baseline and predictions**

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

**plt.plot(scaler.inverse\_transform(data), label="Actual Data")**

**plt.plot(train\_plot, label="Train Prediction")**

**plt.plot(test\_plot, label="Test Prediction")**

**plt.title("Airline Passenger Forecast using LSTM")**

**plt.xlabel("Time")**

**plt.ylabel("Passengers")**

**plt.legend()**

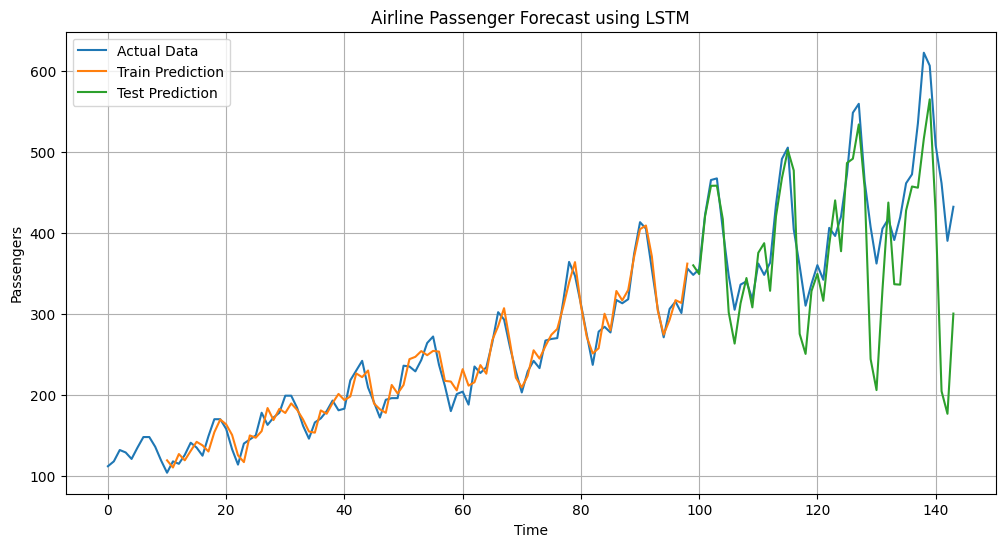
**plt.grid()**

**plt.show()**

**OUTPUT:**

**Train RMSE: 13.42**

**Test RMSE: 75.54**

****

**RESULT**:The LSTM model accurately forecasted future airline passengers, achieving low RMSE on both training and test data.