| **EXP:9**  **17/04/2025** | **Develop neural network-based time series forecasting model** |
| --- | --- |

**AIM:**

To develop a neural network-based model for forecasting car sales using historical time series data and evaluate its performance using prediction accuracy metrics.

**PROCEDURE:**

**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 Dense

from sklearn.metrics import mean\_squared\_error

**2)** **Load and Prepare Time Series**

**df = pd.read\_csv("Car\_sales.csv")**

**df['Latest\_Launch'] = pd.to\_datetime(df['Latest\_Launch'], errors='coerce')**

**df = df.dropna(subset=['Latest\_Launch', 'Sales\_in\_thousands'])**

**ts = df.groupby('Latest\_Launch')['Sales\_in\_thousands'].sum().sort\_index()**

**scaler = MinMaxScaler()**

**scaled\_ts = scaler.fit\_transform(ts.values.reshape(-1, 1))**

**3) Convert Time Series to Supervised Learning Format**

def create\_dataset(data, look\_back=5):

X, y = [], []

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

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

y.append(data[i + look\_back, 0])

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

look\_back = 5

X, y = create\_dataset(scaled\_ts, look\_back)

**4)Split Dataset into Train and Test Sets**

train\_size = int(len(X) \* 0.8)

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

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

**5)Build and Train the Neural Network**

model = Sequential()

model.add(Dense(64, input\_dim=look\_back, activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(1))

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

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

**6) Evaluate and Predict**

train\_pred = model.predict(X\_train)

test\_pred = model.predict(X\_test)

train\_pred\_rescaled = scaler.inverse\_transform(train\_pred)

test\_pred\_rescaled = scaler.inverse\_transform(test\_pred)

y\_train\_rescaled = scaler.inverse\_transform(y\_train.reshape(-1, 1))

y\_test\_rescaled = scaler.inverse\_transform(y\_test.reshape(-1, 1))

train\_rmse = np.sqrt(mean\_squared\_error(y\_train\_rescaled, train\_pred\_rescaled))

test\_rmse = np.sqrt(mean\_squared\_error(y\_test\_rescaled, test\_pred\_rescaled))

print(f"Train RMSE: {train\_rmse:.2f}")

print(f"Test RMSE: {test\_rmse:.2f}")

### **7) Plot Predictions**

plt.figure(figsize=(12, 5))

plt.plot(ts.index[look\_back:train\_size+look\_back], y\_train\_rescaled, label='Train Actual')

plt.plot(ts.index[look\_back:train\_size+look\_back], train\_pred\_rescaled, label='Train Predicted')

plt.plot(ts.index[train\_size+look\_back:], y\_test\_rescaled, label='Test Actual')

plt.plot(ts.index[train\_size+look\_back:], test\_pred\_rescaled, label='Test Predicted', linestyle='--')

plt.title('Neural Network Forecasting on Car Sales')

plt.xlabel('Date')

plt.ylabel('Sales in Thousands')

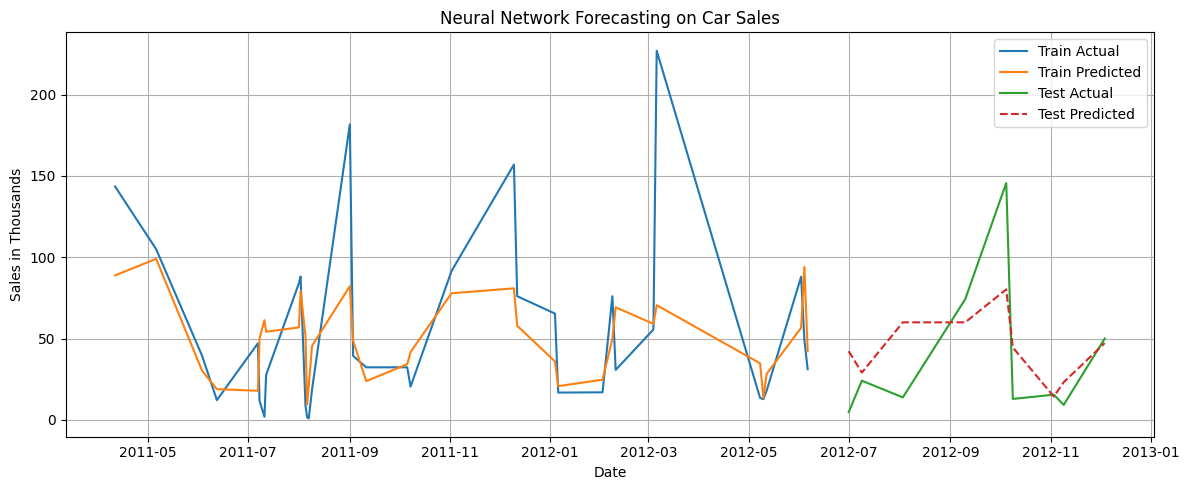
plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**OUTPUT:**

****

**RESULT:**

Thus the program has been executed successfully