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

|  |  |
| --- | --- |
| **EX.N0 : 9** | **Develop neural network-based time series forecasting model** |
| **DATE : 012/04/2025** |

**AIM:**

To Develop neural network-based time series forecasting model

**PROGRAM:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, LSTM

# ----- Step 1: Create Dataset -----

# Population data (you can replace with real data if needed)

population = np.array([

689000, 705000, 720000, 735000, 752000, 768000, 784000, 801000, 819000, 837000,

855000, 873000, 892000, 911000, 931000, 951000, 972000, 993000, 1015000, 1037000,

1060000, 1083000, 1106000, 1129000, 1153000, 1177000, 1201000, 1225000, 1250000,

1275000, 1300000

])

years = np.arange(1980, 1980 + len(population))

# Create DataFrame

df = pd.DataFrame({'Year': years, 'Population': population})

# ----- Step 2: Normalize the Data -----

scaler = MinMaxScaler()

scaled\_data = scaler.fit\_transform(df[['Population']])

# ----- Step 3: Create Time Series Dataset -----

def create\_dataset(data, time\_step=3):

X, y = [], []

for i in range(len(data) - time\_step):

X.append(data[i:i + time\_step])

y.append(data[i + time\_step])

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

time\_step = 3

X, y = create\_dataset(scaled\_data, time\_step)

# Reshape input to [samples, time steps, features]

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

# ----- Step 4: Build the LSTM Model -----

model = Sequential()

model.add(LSTM(64, return\_sequences=True, input\_shape=(time\_step, 1)))

model.add(LSTM(32))

model.add(Dense(1))

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

# ----- Step 5: Train the Model -----

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

# ----- Step 6: Predict the Next Population Value -----

last\_input = scaled\_data[-time\_step:]

last\_input = last\_input.reshape(1, time\_step, 1)

predicted\_scaled = model.predict(last\_input)

predicted = scaler.inverse\_transform(predicted\_scaled)

print(f"\n📈 Predicted next year's population: {predicted[0][0]:,.0f}")

# ----- Step 7: Plot the Results -----

train\_predict = model.predict(X)

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

actual\_plot = df['Population'].values[time\_step:]

plt.figure(figsize=(10, 6))

plt.plot(df['Year'][time\_step:], actual\_plot, label='Actual Population', linewidth=2)

plt.plot(df['Year'][time\_step:], train\_predict\_plot, label='Predicted Population', linestyle='--')

plt.axvline(df['Year'].iloc[-1], color='gray', linestyle=':', alpha=0.6)

plt.title("📊 Population Forecasting using Neural Network (LSTM)")

plt.xlabel("Year")

plt.ylabel("Population")

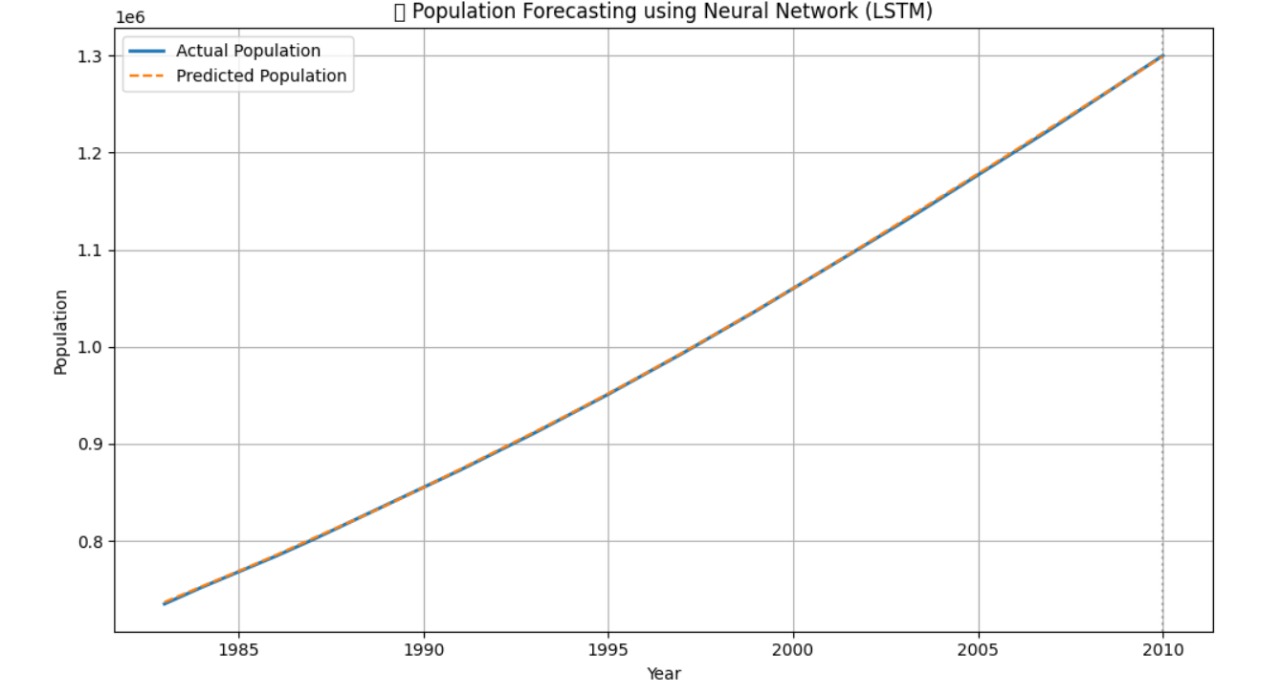
plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**OUTPUT:**



**RESULT:**

Thus, the program for Create an ARIMA model for time series forecasting is executed successfully.