**EX:No.9 NEURAL NETWORK BASED MODEL**

**221501016**

**AIM :** To create an neural network based model for time series forecasting using LSTM.

**PROCEDURE:**

 Read the dataset and convert the date column to datetime format.

 Set the date column as the index for time series analysis.

 Resample or aggregate the data monthly (if needed).

 Normalize the data to scale values between 0 and 1.

 Prepare the data for LSTM.

 Split the data into training and testing sets.

 Build and compile the LSTM model.

 Train the model.

 Make predictions and inverse transform to original scale.

**IMPLEMENTATION :**

**Import libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import mean\_squared\_error

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

from math import sqrt

**Load dataset**

df = pd.read\_csv('/content/Electric\_Production.csv')

df.columns = ['Date', 'Production']

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

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

**Normalize the data**

scaler = MinMaxScaler()

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

**Create sequences**

def create\_sequences(data, seq\_length):

X, y = [], []

for i in range(seq\_length, len(data)):

X.append(data[i-seq\_length:i])

y.append(data[i])

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

sequence\_length = 12 # previous 12 months to predict next month

X, y = create\_sequences(scaled\_data, sequence\_length)

**Train-test split**

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

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

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

**Reshape for LSTM [samples, timesteps, features]**

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

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

**Build LSTM model**

model = Sequential()

model.add(LSTM(64, activation='relu', return\_sequences=False, input\_shape=(sequence\_length, 1)))

model.add(Dense(1))

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

**Train the model**

history = model.fit(X\_train, y\_train, epochs=50, batch\_size=16, validation\_split=0.1, verbose=1)

**Predict and inverse transform**

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

y\_pred\_inv = scaler.inverse\_transform(y\_pred)

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

**Plot results**

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

plt.plot(y\_test\_inv, label='Actual')

plt.plot(y\_pred\_inv, label='LSTM Forecast', color='red')

plt.title('LSTM Forecast vs Actual')

plt.legend()

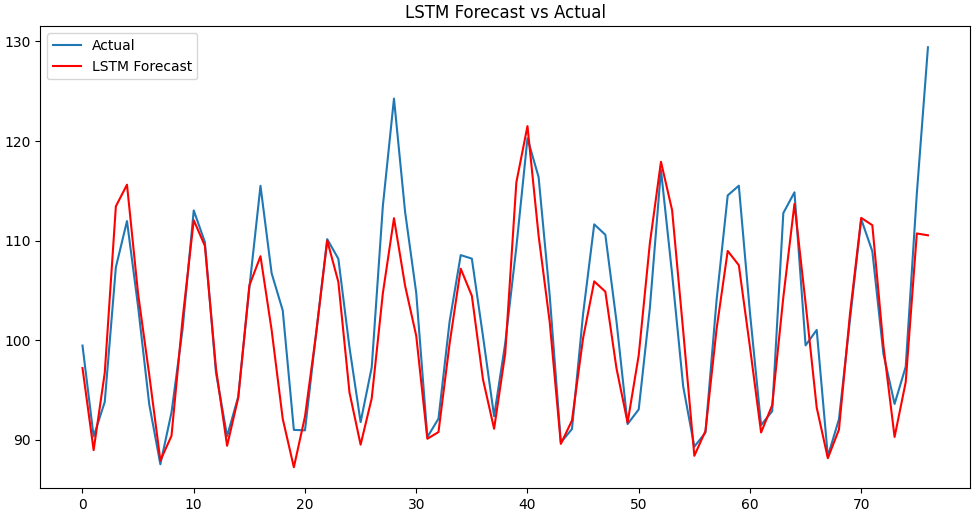
plt.show()

**Evaluate performance**

rmse = sqrt(mean\_squared\_error(y\_test\_inv, y\_pred\_inv))

print(f'RMSE: {rmse:.2f}')

**OUTPUT:**

****

**RESULT :** Thus neural network has been successfully created using LSTM for time series forecasting.