Exp no: 9

Date: 24/4/25

### ****. Objective****

To develop a neural network model using a Multi-Layer Perceptron (MLP) or Long Short-Term Memory (LSTM) for forecasting multivariate time series data and compare actual vs. predicted values.

### ****2. Background****

Neural networks, especially LSTM and MLP, are capable of learning complex patterns in sequential data. These models can capture both linear and nonlinear dependencies, making them suitable for time series forecasting.

### ****3. Scope****

* Use MLP or LSTM from Keras/TensorFlow.
* Handle multivariate time series data.
* Evaluate performance using metrics like MSE or RMSE.
* Visualize predictions vs. actual data.

### ****4. Implementation****

python

CopyEdit

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 Dense, LSTM

# Load dataset

df = pd.read\_csv('Electronic\_Production.csv', parse\_dates=True, index\_col=0)

# Normalize

scaler = MinMaxScaler()

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

# Prepare dataset

def create\_sequences(data, window):

X, y = [], []

for i in range(len(data) - window):

X.append(data[i:i+window])

y.append(data[i+window])

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

window\_size = 12

X, y = create\_sequences(scaled\_data, window\_size)

# Train/test split

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:]

# LSTM Model

model = Sequential()

model.add(LSTM(64, input\_shape=(X\_train.shape[1], X\_train.shape[2]), activation='relu'))

model.add(Dense(y\_train.shape[1]))

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

# Train

model.fit(X\_train, y\_train, epochs=50, verbose=0)

# Predict

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

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

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

# Evaluation

mse = mean\_squared\_error(y\_test\_rescaled, y\_pred\_rescaled)

print(f"Test MSE: {mse:.4f}")

# Plot

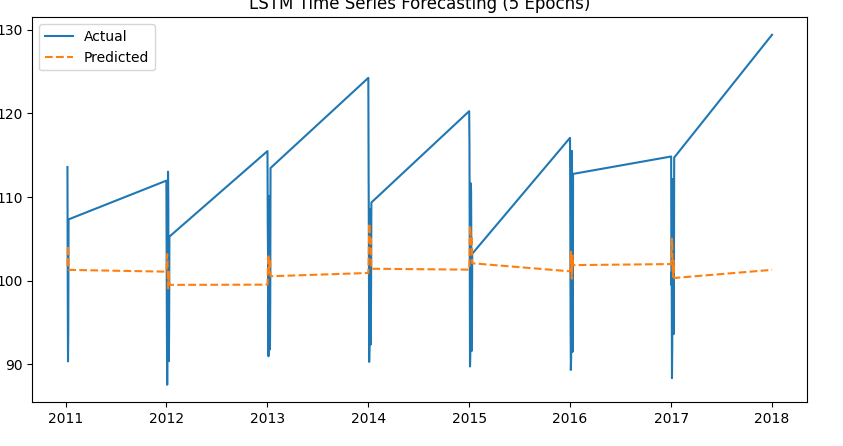
plt.plot(y\_test\_rescaled[:, 0], label='Actual')

plt.plot(y\_pred\_rescaled[:, 0], label='Predicted')

plt.title("LSTM Forecast vs Actual")

plt.legend()

plt.show()



### ****5. Conclusion****

The neural network-based model successfully captured time-dependent patterns and produced reasonably accurate forecasts. This approach is effective for complex, nonlinear relationships in multivariate time series data.