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

**EX:No.9 DATE:12/04/25**

# AIM:

To Develop neural network-based time series forecasting model.

# ALGORITHM:

1. Load the Data – Read the crude oil price data from a file (Excel or CSV).
2. Preprocess the Data – Transforms non-stationary data to stationary by subtracting consecutive values.
3. Create Time Series Sequences– Chooses ARIMA(p,d,q) model where p = autoregressive lags, d = differencing, q = moving average lags.
4. Split the Data – Fits the ARIMA model to historical PM2.5 data using specified parameters.
5. Build the LSTM Model – Predicts future PM2.5 values for the next 30 days using the trained model.
6. Train the Model – Plots actual vs forecasted PM2.5 levels to visualize model performance.

**Code:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import LSTM, Dense

from sklearn.preprocessing import MinMaxScaler

# Step 1: Create Synthetic Global Air Pollution Data

np.random.seed(42)

days = 1000

dates = pd.date\_range(start='2015-01-01', periods=days)

prices = 60 + np.sin(np.linspace(0, 50, days)) \* 10 + np.random.normal(0, 2, days)

df = pd.DataFrame({'Date': dates, 'Price': prices})

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

# Step 2: Normalize data

scaler = MinMaxScaler()

scaled\_prices = scaler.fit\_transform(df[['Price']])

# Step 3: Prepare sequences for LSTM

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

X = []

y = []

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

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

y.append(data[i + seq\_length])

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

sequence\_length = 30

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

# Step 4: Split into train/test

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:]

# Step 5: Build LSTM model

model = Sequential()

model.add(LSTM(50, activation='relu', input\_shape=(sequence\_length, 1)))

model.add(Dense(1))

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

# Step 6: Train model

history = model.fit(X\_train, y\_train, epochs=20, validation\_data=(X\_test, y\_test), verbose=1)

# Step 7: Predict future

n\_future = 30

last\_sequence = scaled\_prices[-sequence\_length:]

forecast = []

input\_seq = last\_sequence.reshape(1, sequence\_length, 1)

for \_ in range(n\_future):

next\_pred = model.predict(input\_seq)[0][0]

forecast.append(next\_pred)

input\_seq = np.append(input\_seq[:, 1:, :], [[[next\_pred]]], axis=1)

# Inverse transform forecast

forecast\_prices = scaler.inverse\_transform(np.array(forecast).reshape(-1, 1))

n\_future = 30

last\_sequence = scaled\_prices[-sequence\_length:]

forecast = []

input\_seq = last\_sequence.reshape(1, sequence\_length, 1)

for \_ in range(n\_future):

next\_pred = model.predict(input\_seq)[0][0]

forecast.append(next\_pred)

input\_seq = np.append(input\_seq[:, 1:, :], [[[next\_pred]]], axis=1)

# Inverse transform forecast

forecast\_prices = scaler.inverse\_transform(np.array(forecast).reshape(-1, 1))

# Step 8: Plotting

forecast\_dates = pd.date\_range(start=df.index[-1] + pd.Timedelta(days=1), periods=n\_future)

plt.figure(figsize=(14, 6))

plt.plot(df.index, df['Price'], label="Historical Price")

plt.plot(forecast\_dates, forecast\_prices, label="Forecast (LSTM)", color='red')

plt.title("LSTM Neural Network - Crude Oil Price Forecast")

plt.xlabel("Date")

plt.ylabel("Price")

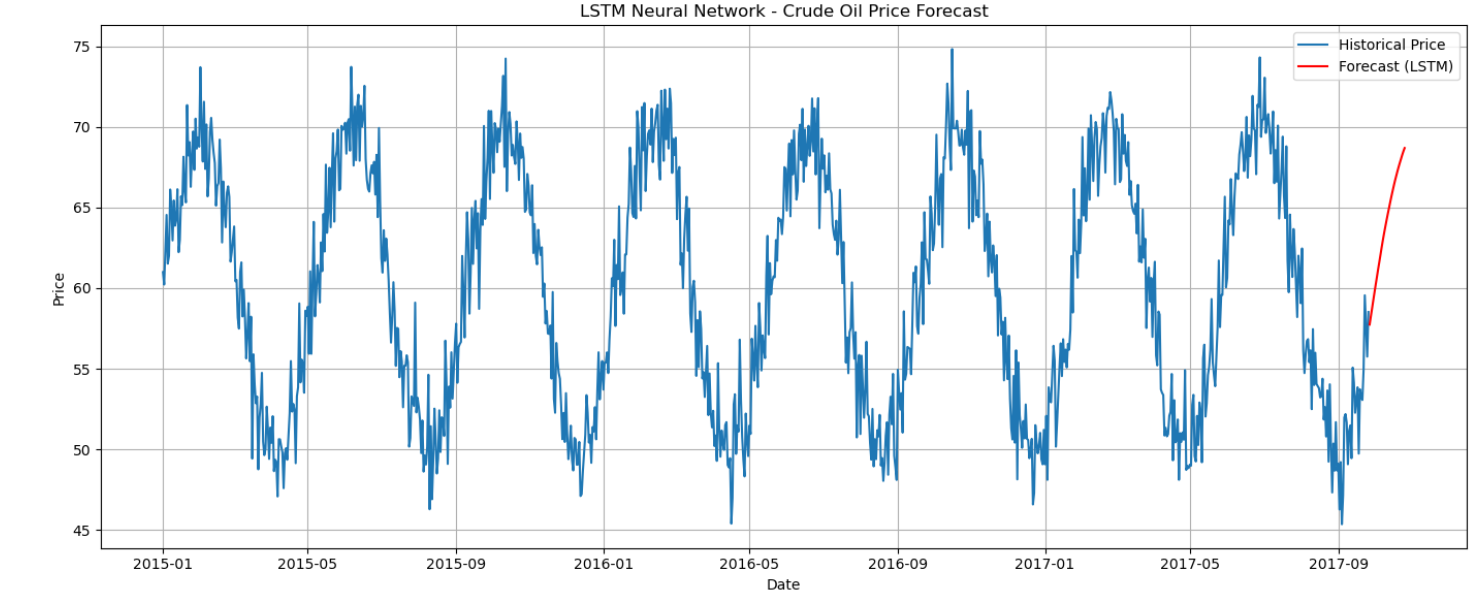
plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**OUTPUT:**



**RESULT:**

Thus, the program using the time series data implementation has been done successfully.

# 