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 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 LSTM, Dense
from tensorflow.keras.callbacks import EarlyStopping

file_path = r"D:/221501507/TIME SERIES ANALYSIS AND FORECASTING/EX06/archive (1)
(1)/FINAL_USO.csv"
df = pd.read_csv(file_path, parse_dates=["Date"], index_col="Date")
df.columns = df.columns.str.strip()

target = "Adj Close"
if target not in df.columns:
    raise ValueError(f"'{target}' column not found in dataset.")

ts = df[[target]].dropna()

scaler = MinMaxScaler()
ts_scaled = scaler.fit_transform(ts)

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  # number of past days to use
X, y = create_sequences(ts_scaled, sequence_length)

split = int(0.8 * len(X))
X_train, X_test = X[:split], X[split:]
y_train, y_test = y[:split], y[split:]

model = Sequential([
```

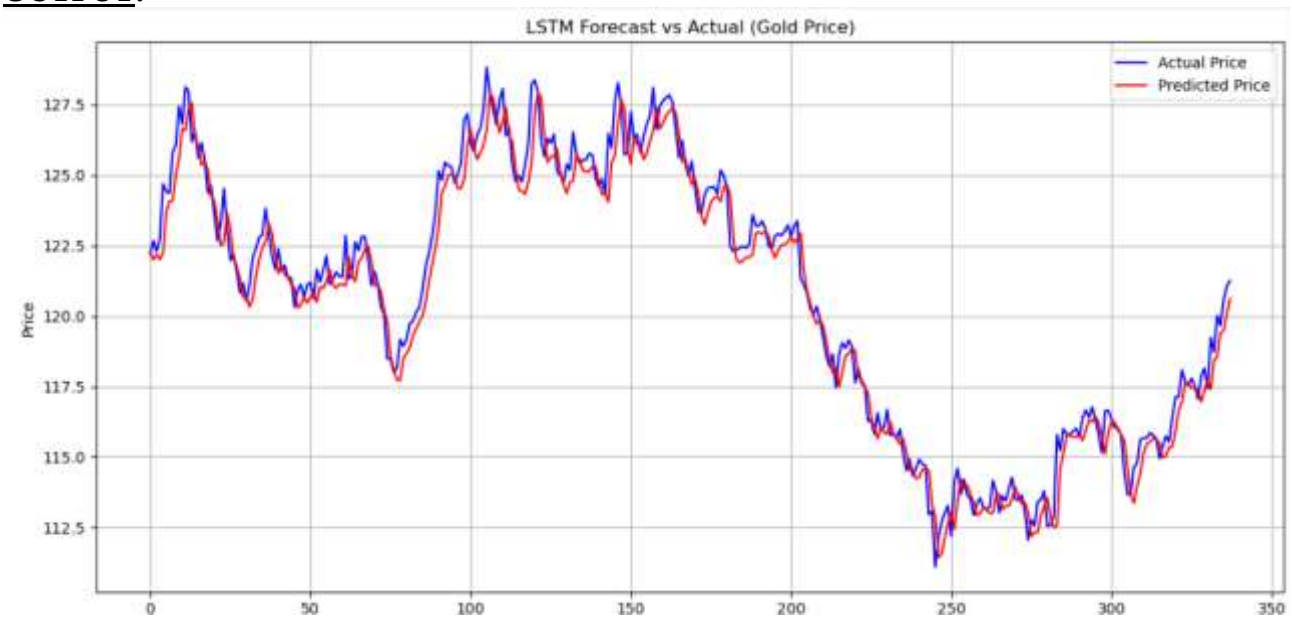
```
    LSTM(50, activation='relu', input_shape=(sequence_length, 1)),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')

early_stop = EarlyStopping(monitor='val_loss', patience=10)
history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=100,
    batch_size=32,
    callbacks=[early_stop],
    verbose=1
)

y_pred = model.predict(X_test)
y_pred_inv = scaler.inverse_transform(y_pred)
y_test_inv = scaler.inverse_transform(y_test)

plt.figure(figsize=(12, 6))
plt.plot(y_test_inv, label='Actual Price', color='blue')
plt.plot(y_pred_inv, label='Predicted Price', color='red')
plt.title("LSTM Forecast vs Actual (Gold Price)")
plt.xlabel("Time Step")
plt.ylabel("Price")
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.