```
import yfinance as yf
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
import matplotlib.pyplot as plt

# 1. Data Acquisition and Preprocessing
ticker = "AAPL"  # Example: Apple stock
data = yf.download(ticker, start="2018-01-01",
end="2024-12-31")

# Data Cleaning and Feature Engineering
data = data[["Close"]]
data["Close_MA"] =
data["Close"].rolling(window=20).mean()
data.dropna(inplace=True)

# 2. Time Series Analysis (Example - Simple
Moving Average)
# 3. Model Selection and Training
scaler = MinMaxScaler()
data["Close_Scaled"] =
scaler.fit_transform(data["Close"])
data["Close_MA_Scaled"] = scaler.transf
orm(data["Close_MA"].values.reshape(-1,
1))

X = np.column_stack((data["Close_MA_Scaled"],
data["Close_Scaled"]))
y = data["Close_Scaled"].shift(-1)  # Predict the
next day's closing price
X = X[:-1]
y = y[:-1]

# Split data into training and testing
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:]

# Model Definition
model = Sequential()
model.add(LSTM(50, activation='relu',
input_shape=(X_train.shape[1], 1)))
model.add(Dense(1))
```

```
# 3. Model Selection and Training
scaler = MinMaxScaler()
data["Close_Scaled"] =
scaler.fit_transform(data["Close"])
data["Close_MA_Scaled"] = scaler.transf
orm(data["Close_MA"].values.reshape(-1,
1))

X = np.column_stack((data["Close_MA_Scaled"],
data["Close_Scaled"]))
y = data["Close_Scaled"].shift(-1)  # Predict the
next day's closing price
X = X[:-1]
y = y[:-1]

# Split data into training and testing
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:]

# Model Definition
model = Sequential()
model.add(LSTM(50, activation='relu',
input_shape=(X_train.shape[1], 1)))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
model.fit(X_train.reshape(-1, X_train.shape[1], 1),
y_train.values, epochs=50, batch_size=32)

# 4. Prediction
predictions = model.predict(X_test.reshape(-1,
X_test.shape[1], 1))
predictions =
scaler.inverse_transform(predictions)
y_test_actual = scaler.inverse_tra
nsform(y_test.values.reshape(-1,
1))

# 5. Visualization and Evaluation
plt.plot(y_test_actual, label="Actual")
plt.plot(predictions, label="Predicted")
plt.xlabel("Time")
plt.ylabel("Stock Price")
plt.title("Stock Price Prediction")
plt.legend()
plt.show()
```

8:05 pm