VIETNAM NATIONAL UNIVERSITY

UNIVERSITY OF ECONOMICS AND LAW

FACULTY OF FINANCE AND BANKING

FINAL REPORT

SUBJECT: MACHINE LEARNING

APPLYING MACHINE LEARNING AND DEEP LEARNING TO PREDICT THE STOCK PRICE

LECTURER: Master Phan Huy Tam

STUDENT: Tran Bich Quan

STUDENT ID: K214140951

CLASS: K21414A

Ho Chi Minh City, June, 2024

TABLE OF CONTENTS

APPLYING MACHINE LEARNING AND DEEP LEARNING TO PREDICT THE STOCK PRICE

I. Introduction

1. Model

In the topic, the author will use and compare the effectiveness of many different models to forecast the Adjusted Closing Price of Military Commercial Joint Stock Bank’s Stock (MB). There are three models used in the research as the following:

- Linear Regression
- K-Nearest Neighbor (KNN)
- Long Short-Term Memory (LSTM)

These methods will be applied in turn and their results will be compared to determine the model that provides the most optimal forecasting performance. Using and comparing many different techniques, from traditional to modern, is a comprehensive approach to find the best model for the problem of predicting the stock price of Military Commercial Joint Stock Bank.

2. Data

The data in the assignment is the adjusted closing price data of Military Commercial Joint Stock Bank (MBBank), with the interval for each data point being 1 day, taken in the period from January 1, 2014 to June 1, 2024. Data is pulled directly from Yahoo Finance via yfinance's API. The models in the reseaech can be applied to the price data of any publicly traded stock.

3. MAE index

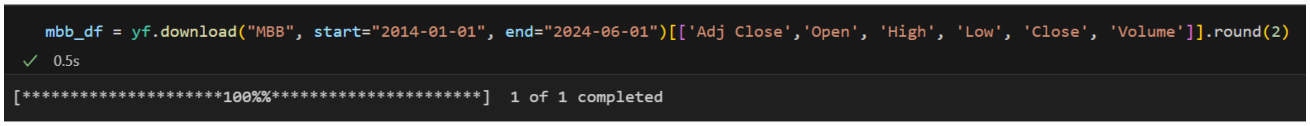
Mean Absolute Error (MAE) is a metric used to evaluate predictive model performance. It represents the average absolute difference between predicted and actual values, providing a straightforward interpretation of error magnitude. The mean absolute error can help measure the accuracy of a given machine learning model. The MAE can be a good complement or alternative to the mean squared error (MSE).

In any case, the closer the value of the MAE is to 0, the better. That said, the interpretation of the MAE is completely dependent on the data. In some cases, a MAE of 10 can be incredibly good, while in others it can mean that the model is a complete failure. The interpretation of the MAE depends on:

- The range of the values,
- The acceptability of error.

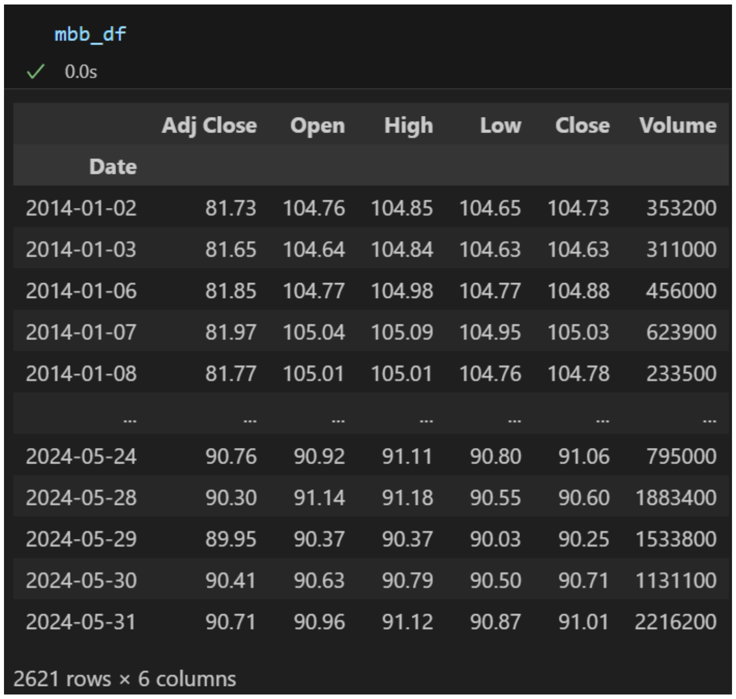
II. Preparing data

***Import data:** Reading MBBank data from Yahoo Finance using yfinance API into pandas DataFrame.



```
mbb_df = yf.download("MBB", start="2014-01-01", end="2024-06-01")[['Adj Close','Open', 'High', 'Low', 'Close', 'Volume']].round(2)
✓ 0.5s
[*********************100%%**********************]  1 of 1 completed
```

***Show data:**

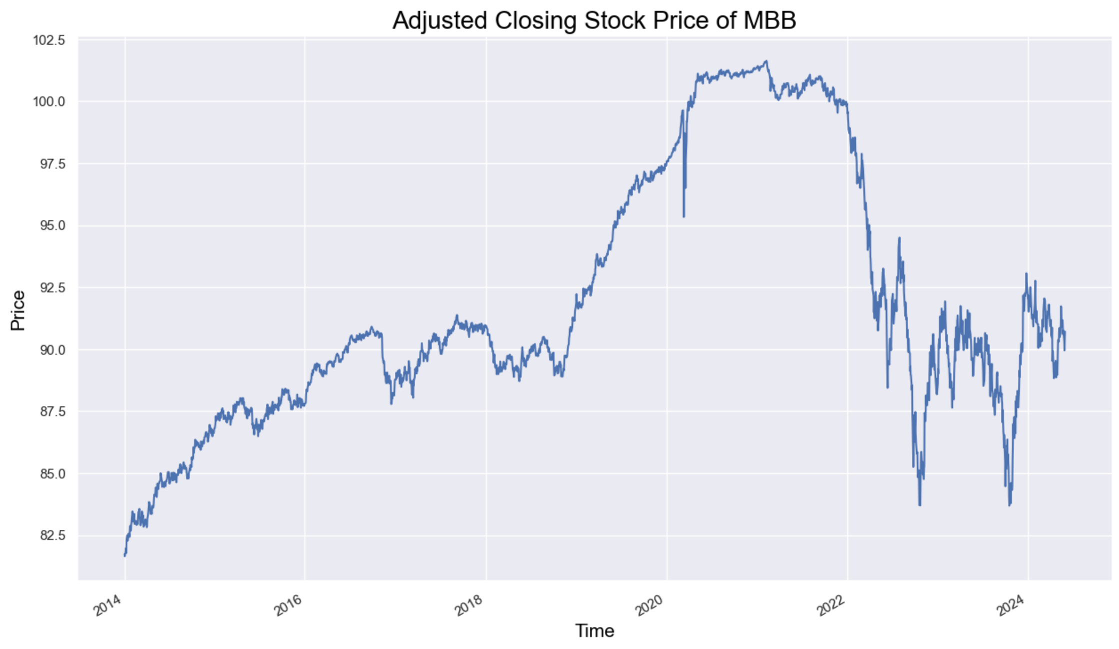


```
mbb_df
✓ 0.0s
```

	Adj Close	Open	High	Low	Close	Volume
Date						
2014-01-02	81.73	104.76	104.85	104.65	104.73	353200
2014-01-03	81.65	104.64	104.84	104.63	104.63	311000
2014-01-06	81.85	104.77	104.98	104.77	104.88	456000
2014-01-07	81.97	105.04	105.09	104.95	105.03	623900
2014-01-08	81.77	105.01	105.01	104.76	104.78	233500
...	...	...	...	...	...	...
2024-05-24	90.76	90.92	91.11	90.80	91.06	795000
2024-05-28	90.30	91.14	91.18	90.55	90.60	1883400
2024-05-29	89.95	90.37	90.37	90.03	90.25	1533800
2024-05-30	90.41	90.63	90.79	90.50	90.71	1131100
2024-05-31	90.71	90.96	91.12	90.87	91.01	2216200

2621 rows × 6 columns

1. Adjusted Closing Stock Price of MBB



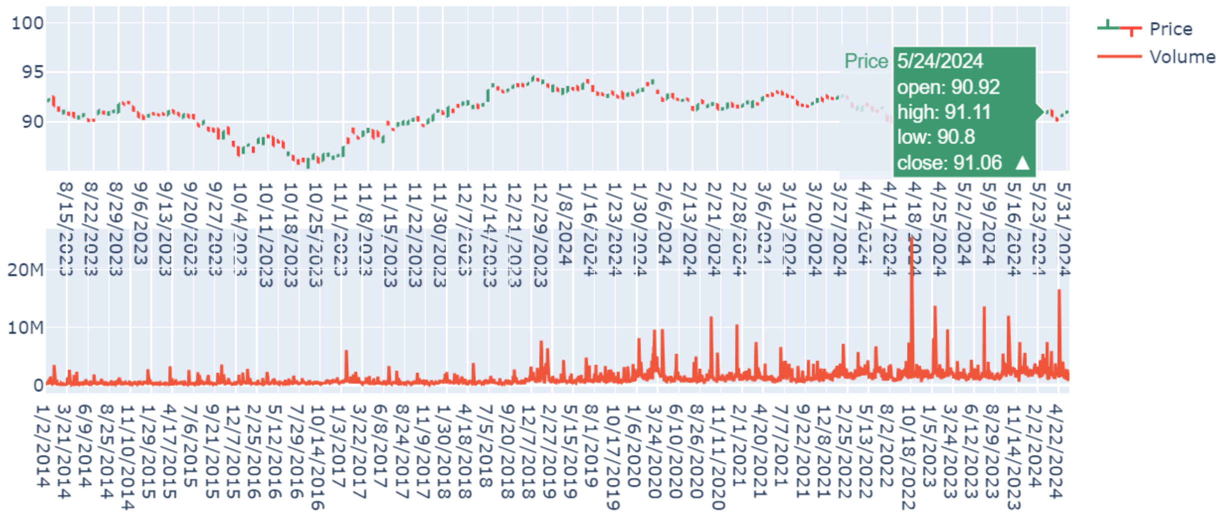
Displaying the adjusted closing stock price of MB Bank for the period 01/01/2014 – 01/06/2024.

2. OHLC Chart

I start with drawing an OHLC (open/high/low/close) chart to get a sense of historical prices. The vertical line represents the high and low for the period, while the line to the left marks the open price and the line to the right marks the closing price. This entire structure is called a bar. When the close is above the open, the bar is often colored black. When the close is below the open the bar is often colored red. The below OHLC I draw Volume chart which shows number of stocks traded each day.

Price Trend: The price of the asset has been fluctuating over the years, with periods of both upward and downward movements. The current price on 2024-06-16 is 91.06, which is within the recent price range.

Price Range: The price has ranged from a low of around 86 to a high of around 97 over the time shown in the chart.



Trading Volume: The trading volume, represented by the red bars, shows periods of high and low activity. There are several instances of spikes in trading volume, indicating increased market interest or significant events related to the asset.

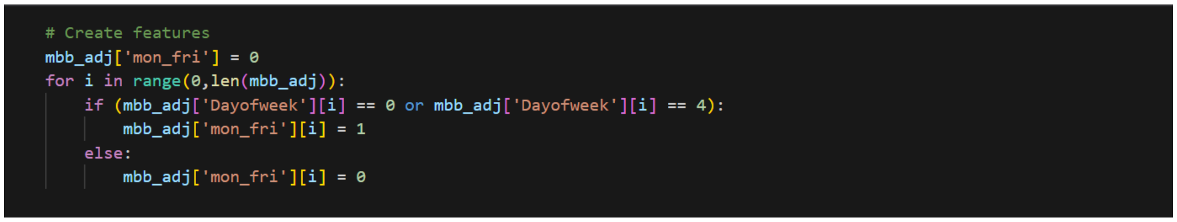
Recent Price and Volume: On the most recent trading day of 2024-06-16, the asset opened at 90.92, reached a high of 91.11, and closed at 91.06. The trading volume for this day appears to be average compared to the historical patterns.

III. Linear Regression

Linear regression is a statistical technique used to model the relationship between variables. The goal is to find the best-fitting straight line that describes their linear relationship, expressed as y = mx + b, where y is the dependent variable, x is the independent variable, m is the slope, and b is the y-intercept. The method of least squares is used to estimate the parameters that provide the best fit for the observed data. Linear regression is used for prediction and has applications in various fields.

1. Feature engineering

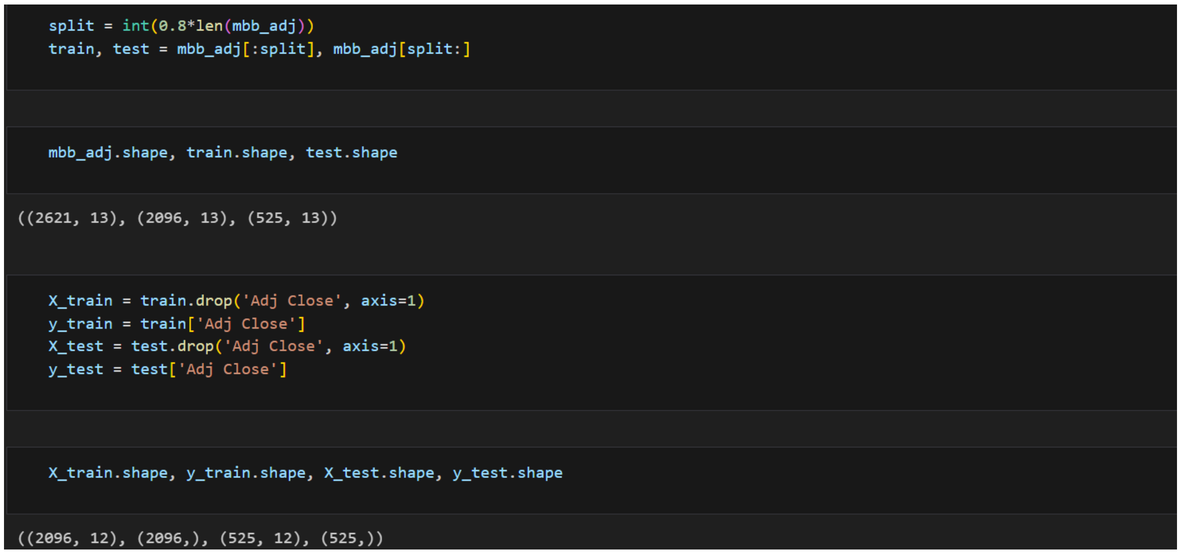
The hypothesis is that the first and last days of the week are likely to influence the adjusted closing price greatly. Therefore, I will add a feature to sort dates based on the day of the week. If the number of days is 0 (Monday) or 4 (Friday), the notes column will display 1, and vice versa, if the day falls on Tuesday, Wednesday, or Thursday, the displayed column will show 0.



```
# Create features
mbb_adj['mon_fri'] = 0
for i in range(0,len(mbb_adj)):
    if (mbb_adj['Dayofweek'][i] == 0 or mbb_adj['Dayofweek'][i] == 4):
        mbb_adj['mon_fri'][i] = 1
    else:
        mbb_adj['mon_fri'][i] = 0
```

2. Split data into train and test

I will divide the data into train set and test set to verify the prediction results. Because this is Time series data, it can not be divided randomly. So, I will divide the first 80% as the train set and the remaining 20% as the test set.



```
split = int(0.8*len(mbb_adj))
train, test = mbb_adj[:split], mbb_adj[split:]
```

```
mbb_adj.shape, train.shape, test.shape
```

```
((2621, 13), (2096, 13), (525, 13))
```

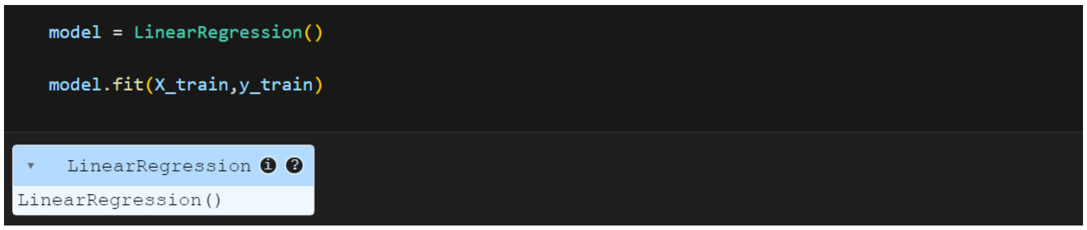
```
X_train = train.drop('Adj Close', axis=1)
y_train = train['Adj Close']
X_test = test.drop('Adj Close', axis=1)
y_test = test['Adj Close']
```

```
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((2096, 12), (2096,), (525, 12), (525,))
```

3. Create, train and test the Linear Regression model

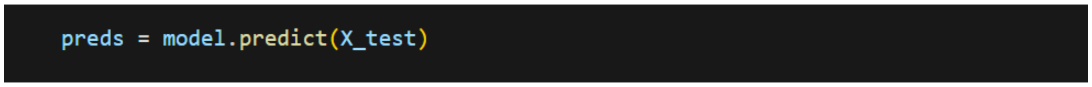
*Create and train model



```
model = LinearRegression()

model.fit(X_train,y_train)
```

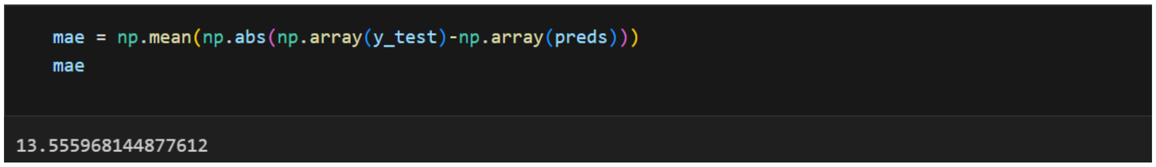
```
▾ LinearRegression
LinearRegression()
```

*Test model: create predictions



```
preds = model.predict(X_test)
```

4. Calculate MAE index

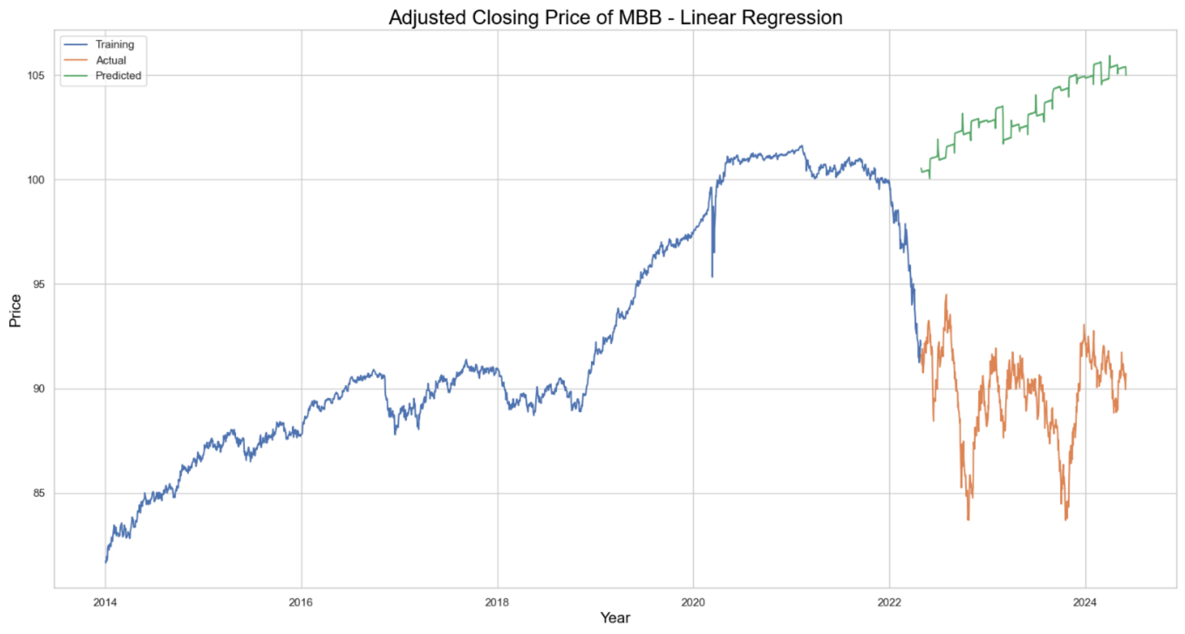


```
mae = np.mean(np.abs(np.array(y_test)-np.array(preds)))
mae
```

```
13.555968144877612
```

The MAE index of Linear Regression is **13.555947688098811**, which indicates the predictions deviate from true values by about 13.56 units on average. The lower the MAE score, the better the model's predictive accuracy, as it suggests smaller overall errors in the predictions.

5. Visualize actual and predicted data



The chart demonstrates the performance of a linear regression model in predicting the adjusted closing price of the MBB asset over time. The model generally captures the overall price trend well, with the predicted values closely tracking the actual prices during periods of market stability. However, the model struggles to accurately forecast sharp price swings and high volatility, particularly during market downturns. It tends to underestimate the magnitude of price drops and overestimate the pace of recoveries. While the linear regression approach provides a reasonable fit, it appears to have limitations in fully capturing the dynamics of this asset in turbulent market conditions.

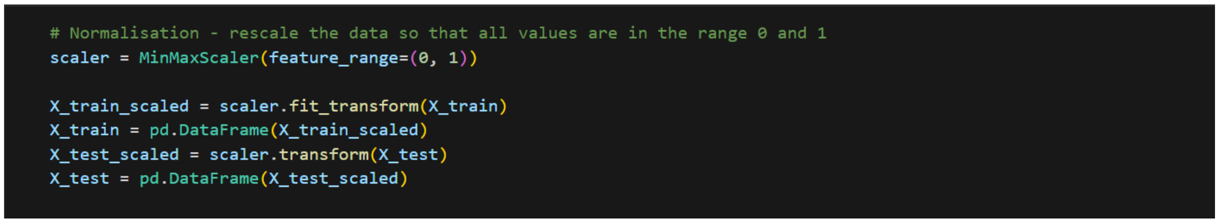
IV. K-Nearest Neighbours

The K-Nearest Neighbors (KNN) algorithm is a non-parametric machine learning technique that can be used to predict stock prices. Unlike linear regression, KNN identifies the K closest historical data points to current market conditions and uses their actual prices to make predictions.

KNN offers flexibility to capture complex, non-linear relationships, and can dynamically adjust predictions as new data becomes available. However, it requires careful tuning of the K parameter and may be sensitive to distance metrics and feature scaling. While KNN could be a useful complement to linear regression, especially during volatile periods, thorough testing and validation would be needed to assess its suitability for a given stock or asset.

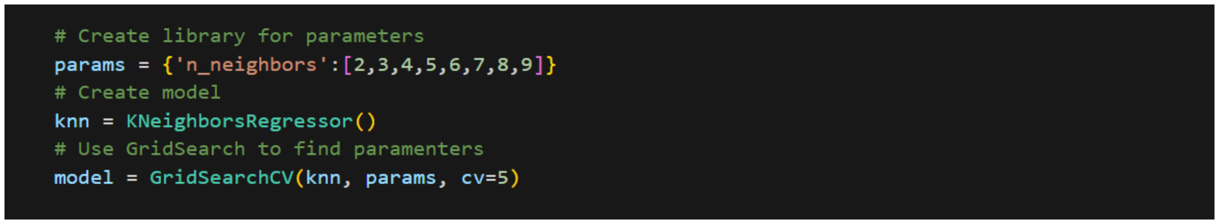
To be able to run this model, we must first Normalize the parameters of the dataset into a ratio so that all data is in the range of values from 0 to 1. Then execute GridSearch to find the optimal parameters for the model.

1. Feature Scaling



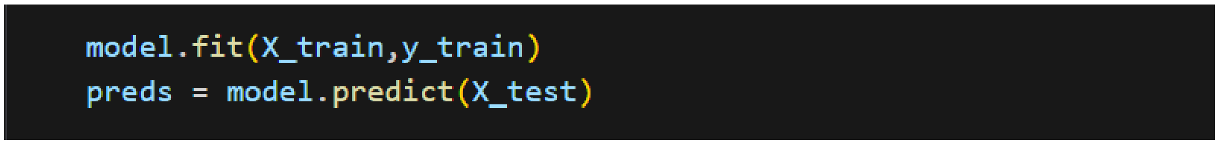
```
# Normalisation - rescale the data so that all values are in the range 0 and 1
scaler = MinMaxScaler(feature_range=(0, 1))

X_train_scaled = scaler.fit_transform(X_train)
X_train = pd.DataFrame(X_train_scaled)
X_test_scaled = scaler.transform(X_test)
X_test = pd.DataFrame(X_test_scaled)
```

2. Grid Search finds optimal parameters



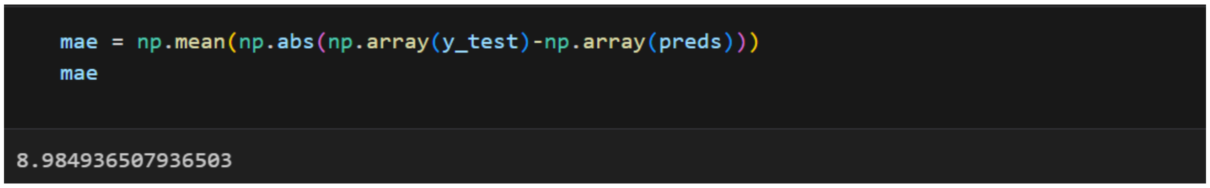
```
# Create library for parameters
params = {'n_neighbors':[2,3,4,5,6,7,8,9]}
# Create model
knn = KNeighborsRegressor()
# Use GridSearch to find paramenters
model = GridSearchCV(knn, params, cv=5)
```

3. Train and test model



```
model.fit(X_train,y_train)
preds = model.predict(X_test)
```

4. Calculate MAE index

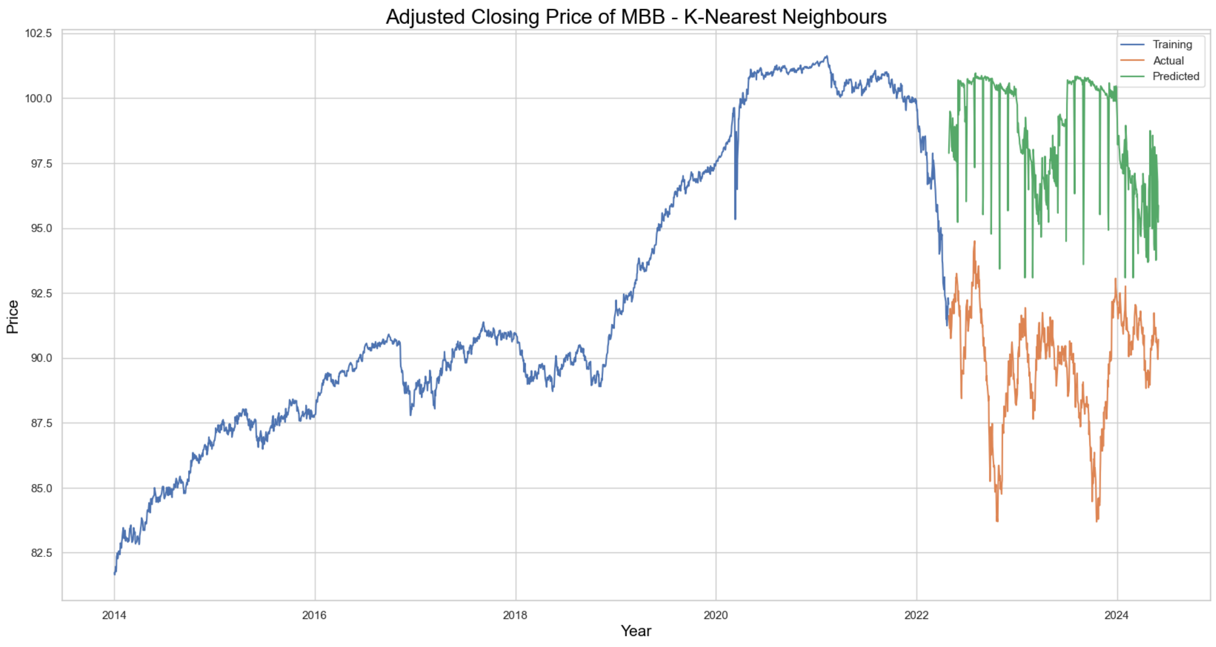


```
mae = np.mean(np.abs(np.array(y_test)-np.array(preds)))
mae
```

```
8.984936507936503
```

The KNN model has the MAE index of **8.984936507936503**, which is smaller than the MAE of Linear Regression.

5. Visualize actual and predicted data



The KNN model outperforms the linear regression approach in capturing price fluctuations, especially during volatile market periods. While the linear regression struggled with sharp swings, the KNN predictions align more tightly with actual prices throughout. However, the KNN exhibits more variation in stable markets, potentially indicating sensitivity to the chosen nearest neighbors. Overall, KNN's non-parametric flexibility makes it a valuable complement to linear regression for stock price forecasting, particularly in dynamic market conditions.

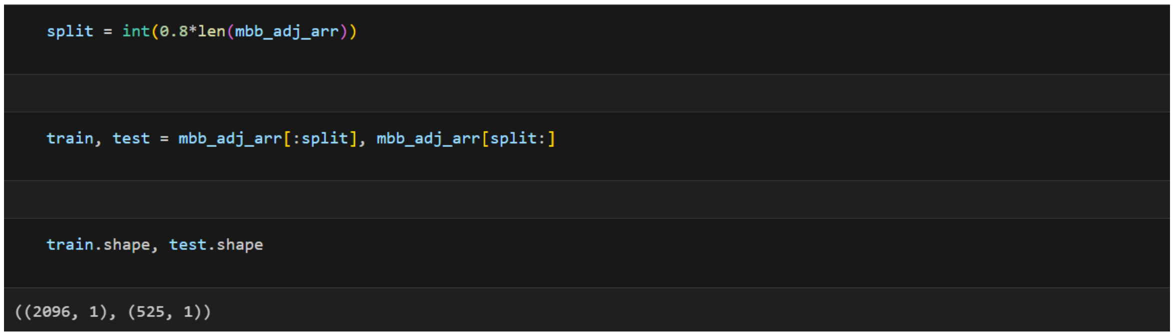
V. Long Short-Term Memory (LSTM)

LSTM (Long Short-Term Memory) is a recurrent neural network (RNN) architecture widely used in Deep Learning. It excels at capturing long-term dependencies, making it ideal for sequence prediction tasks.

Unlike traditional neural networks, LSTM incorporates feedback connections, allowing it to process entire sequences of data, not just individual data points. This makes it highly effective in understanding and predicting patterns in sequential data like time series, text, and speech.

1. Split data into train and test set

I still divide the model into train and test set with a ratio of 80%/20%.

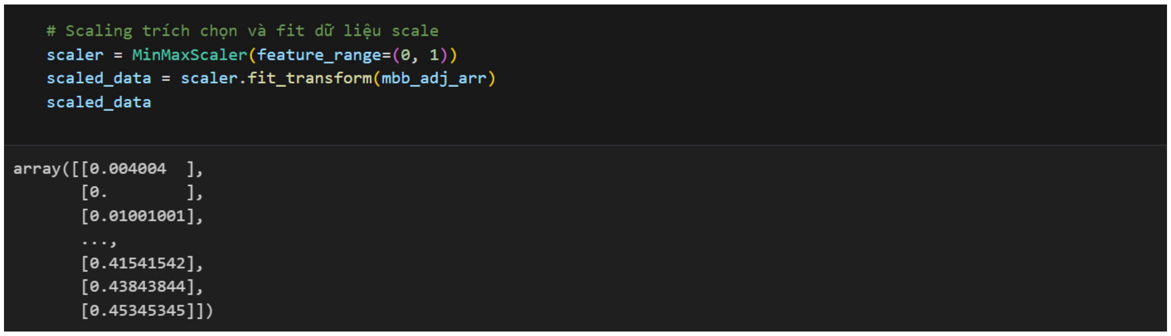


```
split = int(0.8*len(mbb_adj_arr))

train, test = mbb_adj_arr[:split], mbb_adj_arr[split:]

train.shape, test.shape
```

```
((2096, 1), (525, 1))
```

2. Feature scaling

Standardize data by scaling it into a 0 to 1 format to improve learning and model convergence.

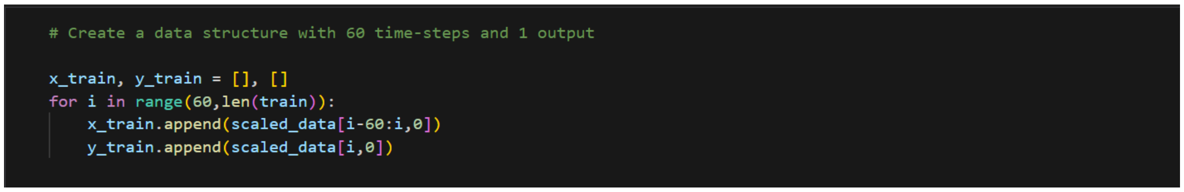


```
# Scaling trích chọn và fit dữ liệu scale
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(mbb_adj_arr)
scaled_data
```

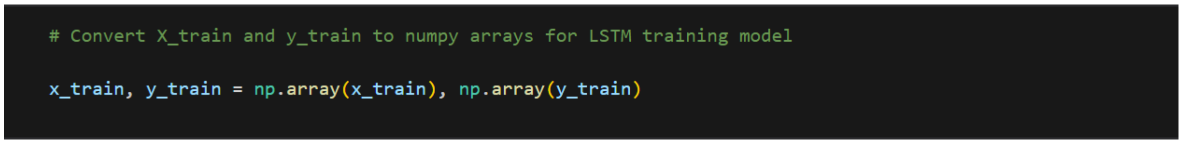
```
array([[0.004004  ],
       [0.        ],
       [0.01001001],
       ...,
       [0.41541542],
       [0.43843844],
       [0.45345345]])
```

3. Create training data

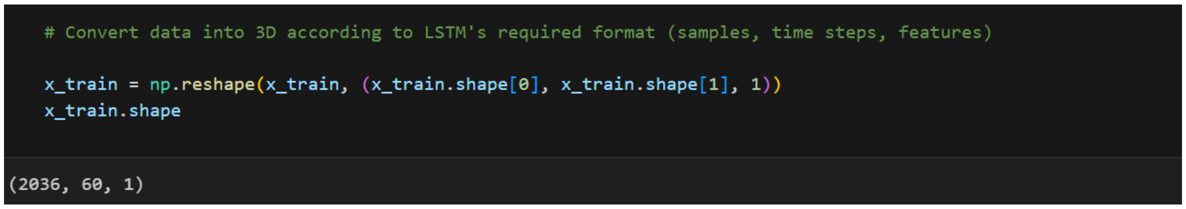
Before inputting data into the LSTM model, it's essential to process the data to meet the model's specific requirements.



```
# Create a data structure with 60 time-steps and 1 output

x_train, y_train = [], []
for i in range(60,len(train)):
    x_train.append(scaled_data[i-60:i,0])
    y_train.append(scaled_data[i,0])
```



```
# Convert X_train and y_train to numpy arrays for LSTM training model

x_train, y_train = np.array(x_train), np.array(y_train)
```

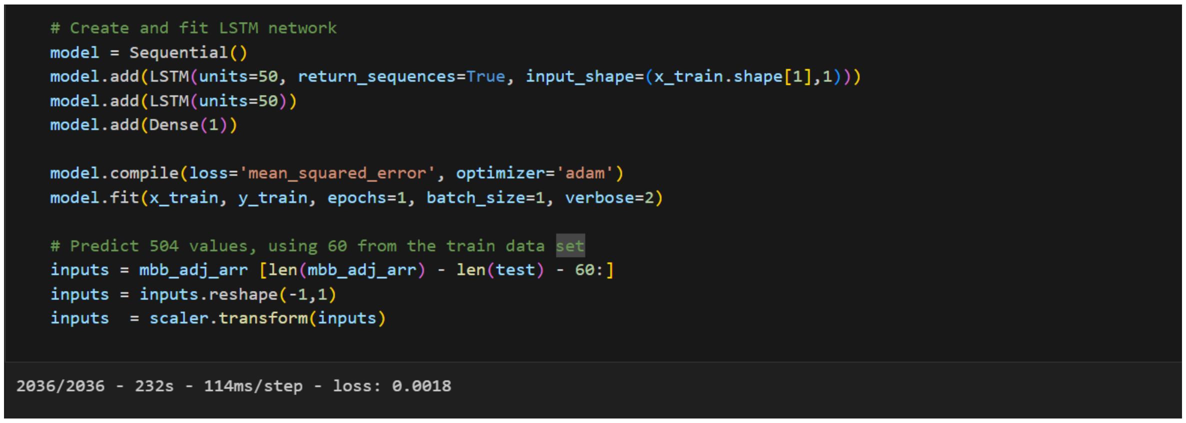


```
# Convert data into 3D according to LSTM's required format (samples, time steps, features)

x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))
x_train.shape
```

```
(2036, 60, 1)
```

4. Create and train model

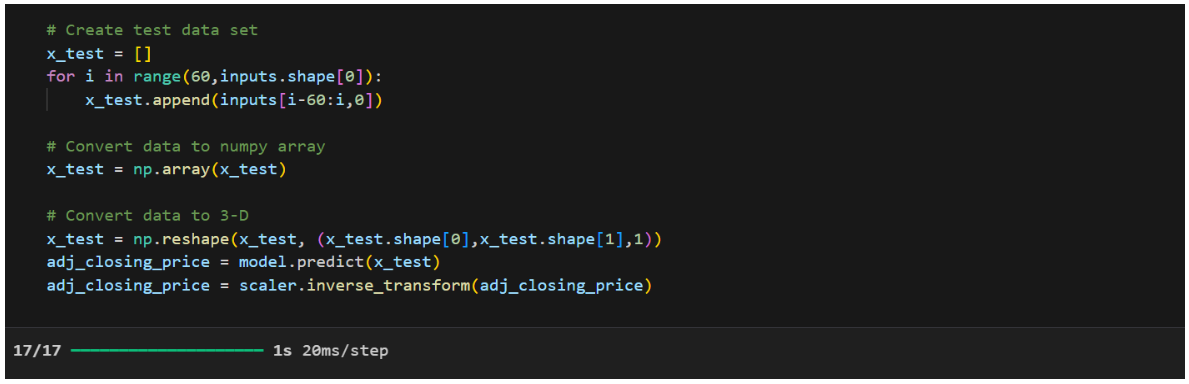
We will implement a simple model consisting of a hidden layer with 50 neurons, input data shape with number of time steps of (60) and dimensionality of (1) and an output layer with time step 1. The model will be compiled using Mean Absolute Error (MAE) and ADAM (Adaptive Moment Estimation) optimizer, then will be fit to the training set with one epoch and batch size is one.



```
# Create and fit LSTM network
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1],1)))
model.add(LSTM(units=50))
model.add(Dense(1))

model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(x_train, y_train, epochs=1, batch_size=1, verbose=2)

# Predict 504 values, using 60 from the train data set
inputs = mbb_adj_arr [len(mbb_adj_arr) - len(test) - 60:]
inputs = inputs.reshape(-1,1)
inputs  = scaler.transform(inputs)
```

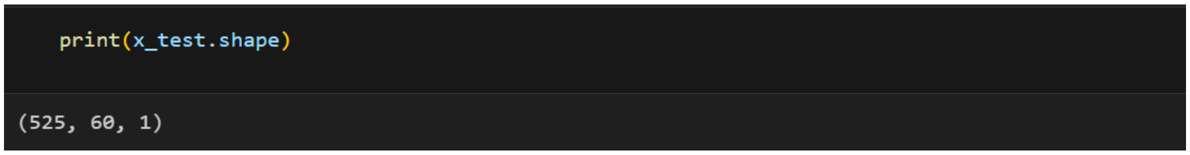
```
2036/2036 - 232s - 114ms/step - loss: 0.0018
```

5. Create test set and test



```
# Create test data set
x_test = []
for i in range(60,inputs.shape[0]):
    x_test.append(inputs[i-60:i,0])

# Convert data to numpy array
x_test = np.array(x_test)

# Convert data to 3-D
x_test = np.reshape(x_test, (x_test.shape[0],x_test.shape[1],1))
adj_closing_price = model.predict(x_test)
adj_closing_price = scaler.inverse_transform(adj_closing_price)
```

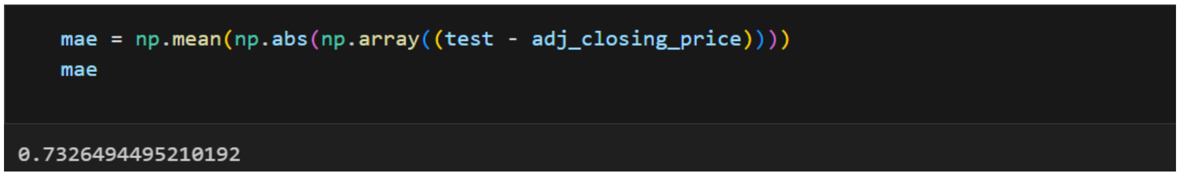
```
17/17 ━━━━━━━━━━━━━━━━━━━━ 1s 20ms/step
```



```
print(x_test.shape)
```

```
(525, 60, 1)
```

6. Calculate MAE index



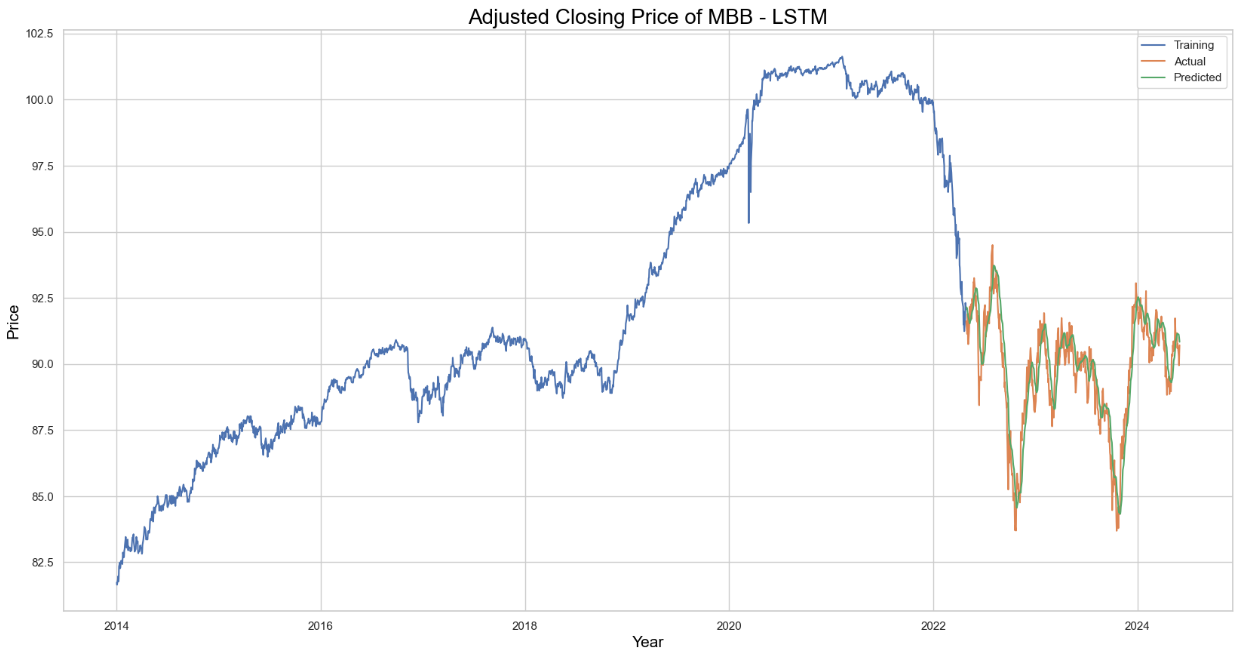
```
mae = np.mean(np.abs(np.array((test - adj_closing_price))))
mae
```

```
0.7326494495210192
```

With the MAE of **0.7326494495210192**, the LSTM model has shown better results compared to linear and basic models like Linear Regression and KNN. This proves that the LSTM model has a better ability to model non-linear relationships in the data, leading to higher predictive performance.

7. Visualize actual and predicted data

The LSTM model appears to have a superior performance compared to the traditional Linear Regression and KNN models. The LSTM predictions closely track the actual MBB prices, especially during the more volatile periods, suggesting that the LSTM model can effectively capture the non-linear dynamics and complex relationships present in the data. This advantage of the LSTM model over simpler techniques highlights its suitability for modeling and predicting financial time series data.



VI. Comparing the mean absolute error of the models

MODEL	MAE INDEX
Linear Regression	13.555947688098811
K-Nearest Neighbours	8.984936507936503
Long Short-Term Memory	0.7326494495210192

The MAE values clearly highlight the LSTM model's advantages in modeling the non-linear and time-dependent characteristics of the time series data, outperforming the more traditional Linear Regression and KNN models. This suggests that the LSTM's ability to learn and remember patterns over time makes it a more suitable choice for this particular application.

While the LSTM model has proven to be effective on its own, we believe there is further potential to enhance the stock price forecasting accuracy by exploring ensemble and hybrid modeling approaches. Specifically, we propose to combine the LSTM model with other techniques, such as integrating the LSTM model with ensemble algorithms like Random Forest and Gradient Boosting; or combining the LSTM with other advanced machine learning algorithms, including Support Vector Regression and XGBoost, may uncover complementary non-linear relationships that could further improve the overall forecasting performance.

References

1. (Long short term memory (LSTM), 2019)
2. (What is LSTM? Introduction to Long Short-Term Memory, 2024)
3. (How to Calculate Mean Absolute Error (MAE) in Python, 2022)
4. (Understanding an OHLC Chart and How to Interpret It, 2024)
5. (The k-Nearest Neighbors (kNN) Algorithm in Python, n.d.)
6. (Predicting Stock Prices with Python, 2018)