**Riga Technical University**

Faculty of Computer Science, Information Technology and Energy

**Report on the second practical assignment**

Study course "Artificial Intelligence in Digital Humanities"

Student: Fengdi Huang 231AHG003

Teaching staff: Alla Anohina-Naumeca

Project link: <https://github.com/JesseLau24/Fengdi_Huang_Second_PA>

Link to the dataset: <https://www.kaggle.com/datasets/programmerrdai/nvidia-stock-historical-data>

2024/2025 academic year

**For the record:**

I have used Python for this instead of Orange tool because I cannot install it properly on my laptop (might have something to do with Nvidia driver and toolkit).

I have used ChatGPT for automation and debugging, but only for the purpose of enhancing efficiency.

**The idea for this assignment is mine, and the whole design is done by me. AI tools merely helps with code generation for repeated workflow and debugging.**

Here is the ChatGPT conversation, in case you want to check it:

<https://chatgpt.com/share/674c9b04-8cc8-8001-9107-e2169ad8793d>

And also, I have referred to the book *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* by Géron, Aurélien for some fundamental knowledge for different algorithms that are not covered by our lecture (LSTM and Random Forest mostly)

# Orange tool workflow

< a screenshot of the workflow created in the Orange tool>

# Part I

<this subsection should provide a general description of the dataset, accompanied by screenshots and references to the information sources used>

## Description of the dataset

### Dataset title: NVIDA Stock Historical Data

### Dataset source:

### Kaggle (<https://www.kaggle.com/datasets/programmerrdai/nvidia-stock-historical-data>)

### Creator and/or owner of the dataset: HRTERHRTER

Description of the dataset problem domain:

This dataset is the historical data of the stock NVIDA’s price from 01/21/1999 to 06/18/2024 (but only the data from around the past 4 years are used in this assignment).

It has 7 columns: Date, Open, High, Lowm Close, Adj Close and Volume.

Later, 5 more columns are added, which are: RSI\_7, MACD\_Line, MACD\_Signal, MACD\_Hist and Is\_Higher\_Than\_Previous\_Close.

They are calculated with figures from the previous 7 columns with Ta Library

It contains 6595 rows, of which the last 974 rows are used in this assignment

Dataset licensing conditions: Apache 2.0

Information about the method or procedure for collecting the dataset:

Downloaded from Kaggle directly in the format of CSV files.

## Description of the dataset content

Number of data objects in the dataset:

Totally: 6595 (Only the last 974 are used)

Representation of features (attributes) of the dataset together with their roles in the Orange tool:



### Number of classes in the dataset: 2 Classes.

Description of classes:

Under the column Is\_higher\_Than\_Precious\_Close:

1. if the closing price of this row is higher than that of the previous day, it would be labelled 1,
2. otherwise, if the closing price is not higher than that of the previous day, it would be labelled 0.

Number of data objects belonging to each class:

<add rows to table as needed>

|  |  |
| --- | --- |
| **Class label** | **Number of data objects** |
| 1 (Higher) | 530 |
| 0 (Not Higher) | 444 |
|  |  |

### Description of features:

<add rows to table as needed>

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature title** | **Explanation of the feature** | **Value type** | **Range of values** |
| Open | The open price of that trading day | Float64 | 10.971 - 132.990 |
| High | The highest price of that trading day | Float64 | 11.136 -136.330 |
| Low | The lowest price of that trading day | Float64 | 10.795 – 130.690 |
| Close | The close price of that trading day | Float64 | 10.850 – 135.580 |
| Adj Close | Adjusted closing price (based on dividends, split shares and etc) | Float64 | 10.817 – 135.580 |
| Volume | Trading volume of that trading day | Int64 | 97884000 - 1543911000 |
| RSI\_7 | Indicator that measures the relative strength of that stock based on the data over the past 7 days | Float64 | 13.600 – 96.045 |
| MACD\_Line | The difference between 12-day EMA and 26-day EMA, which indicates trends of the price movement | Float64 | -1.794 – 9.774 |
| MACD\_Signal | The 9-day EMA, for generate trading signals | Float64 | -1.539 – 8.649 |
| MACD\_Hist | Difference between MACD\_Line and MACD\_Signal | Float64 | -1.539 – 2.358 |

### Data file structure:

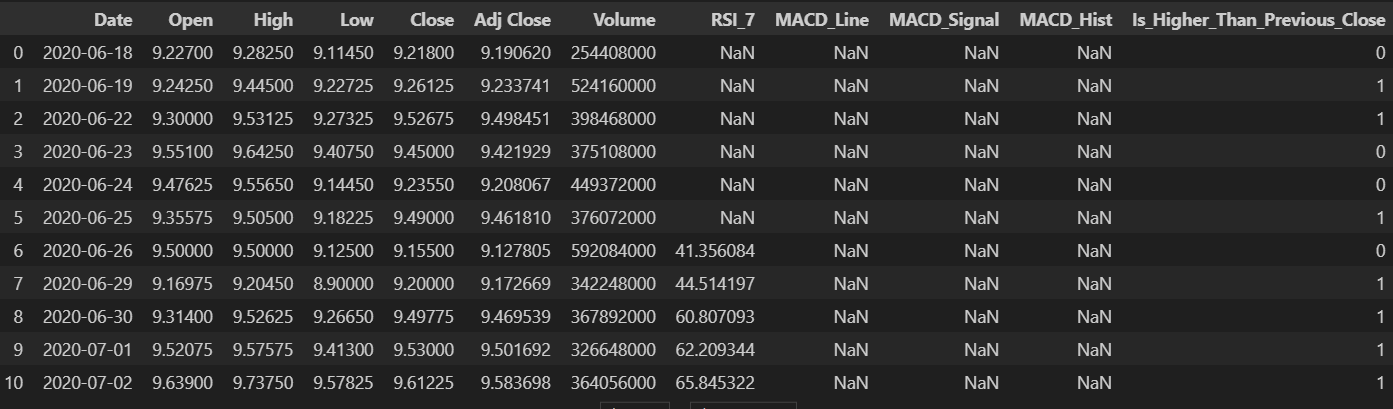


### Information about missing values or outliers:

Description: The original data doesn’t have any missing values. However, when generating additional features (RSI\_7, MACD\_Line, MACD\_Signal, MACD\_Hist), since all the figures are calculated based on data from the previous days, making the first few rows containing NaN values.

I decided to drop those lines since I still have many rows left, and it would not affect the rsult

Data before dropping NaN values:



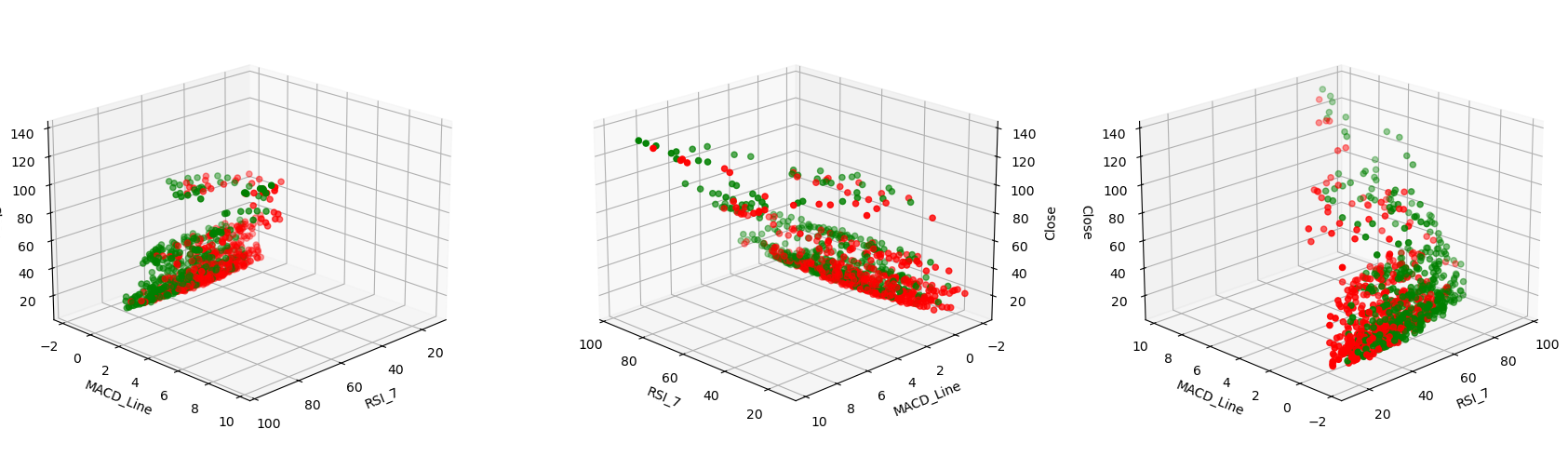
## Visual and statistical representation of the dataset

I have tried more combinations and here is just part of the total visualization.

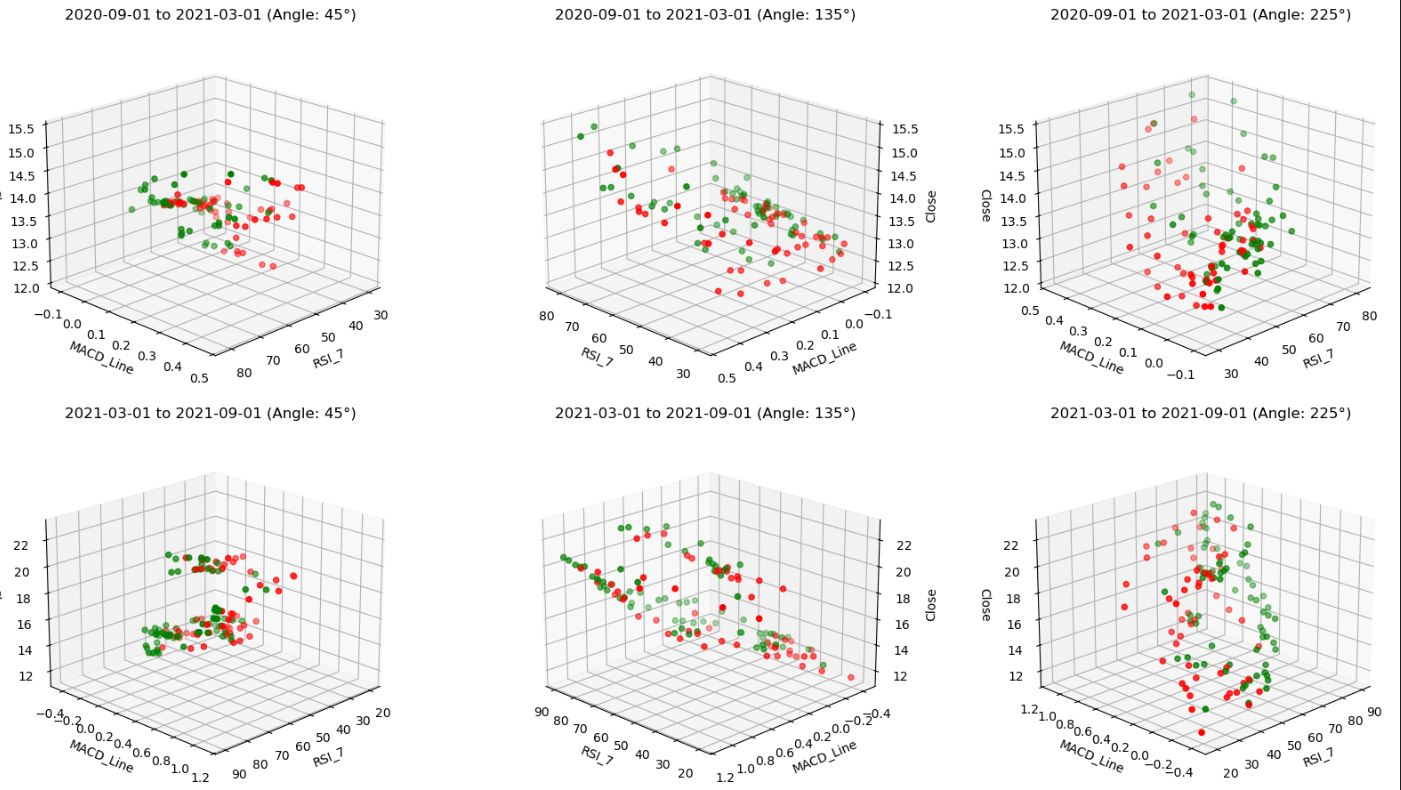
More Visualization can be found in the Jupyter Notebook File here:

<https://github.com/JesseLau24/Fengdi_Huang_Second_PA/blob/main/NVDA_Stock_Prediction_Fengdi_Huang_231AHG003_Second_Practical_Task.ipynb>

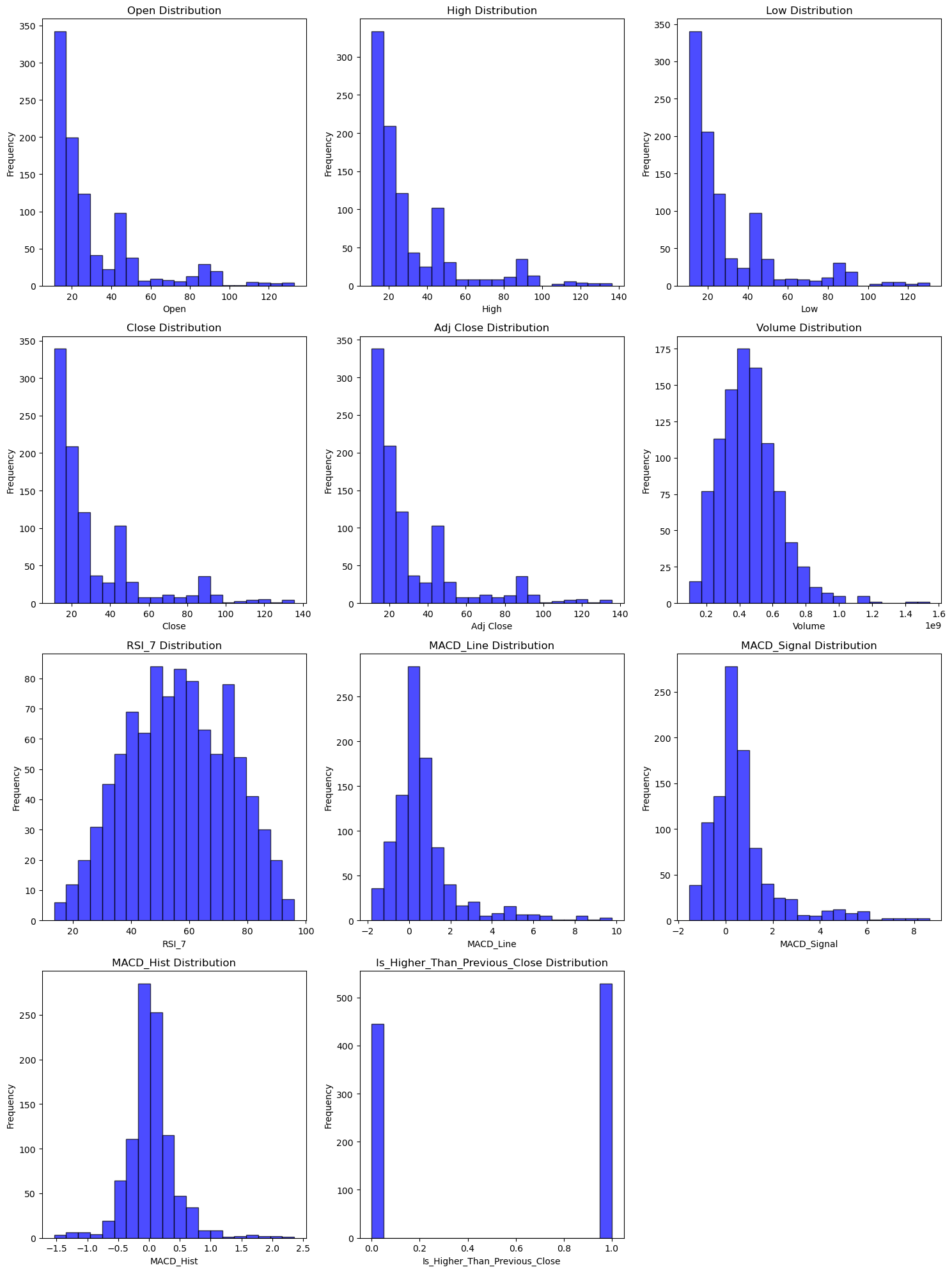
**3D Scatterplot:** x RSI\_7, y MACD\_Line, z Close Price, is higher, green, is not: red

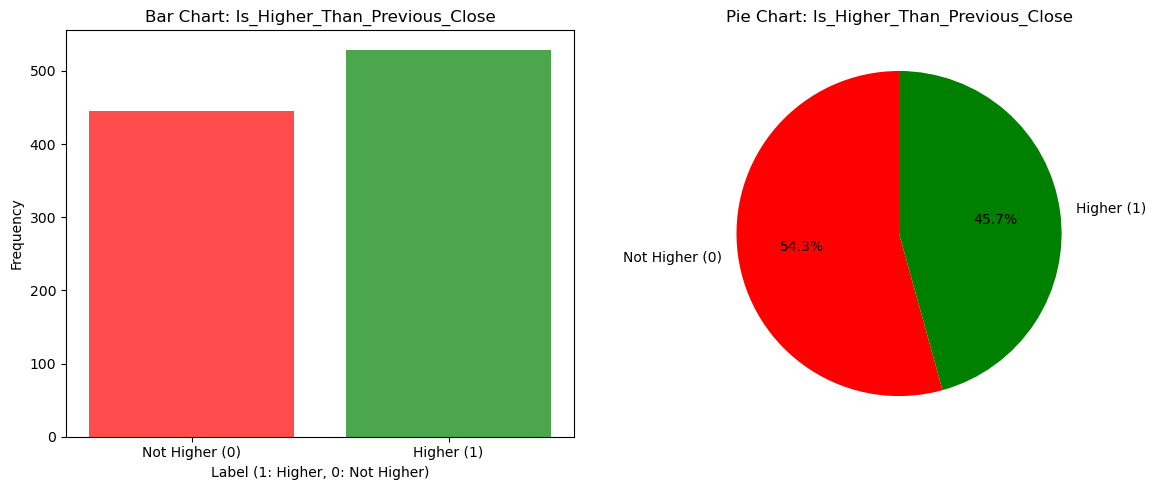


**3D Scatterplot in 6-month Timeframe**: x RSI\_7, y MACD\_Line, z Close Price, is higher, green, is not: red

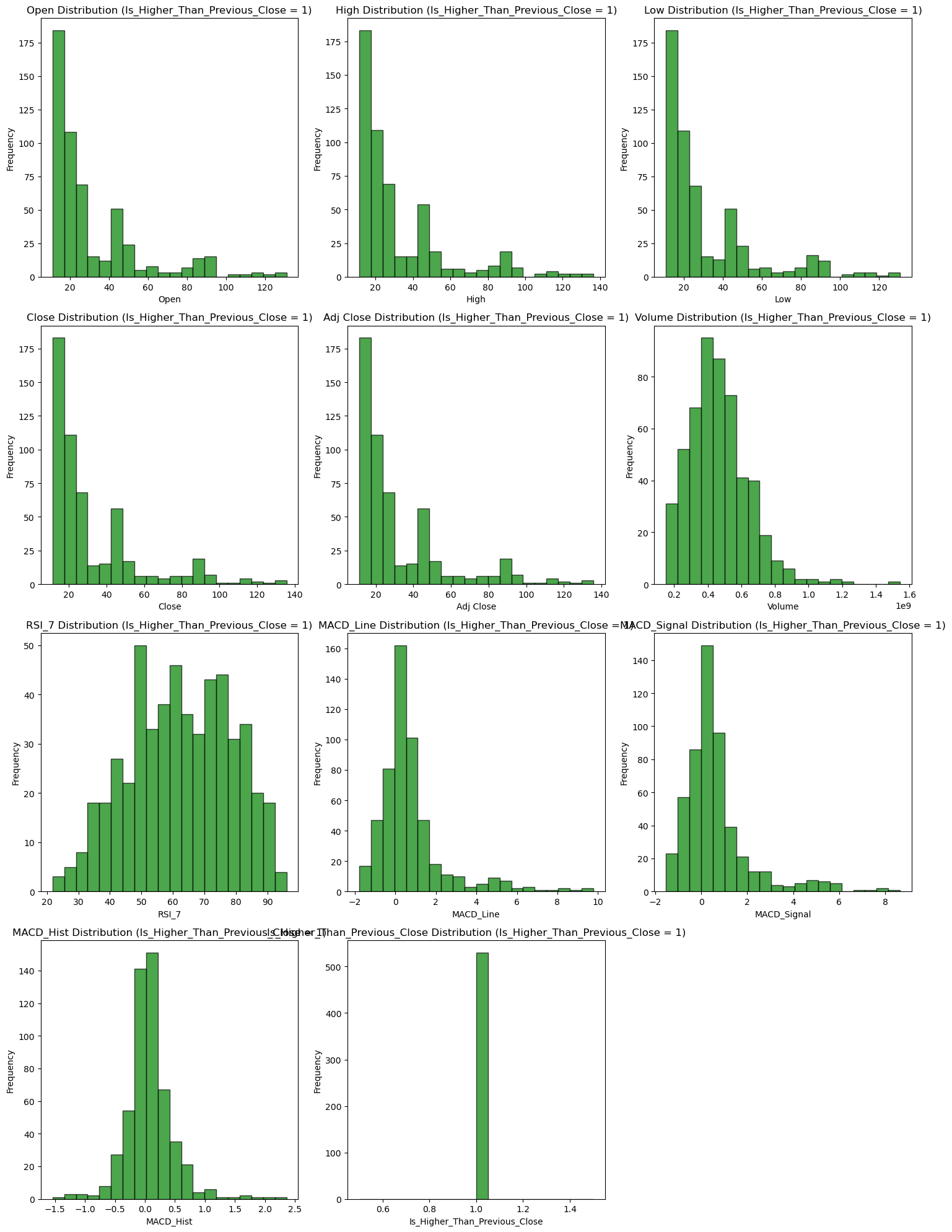


**Data Summary (of all 974 rows used in this assignment):**

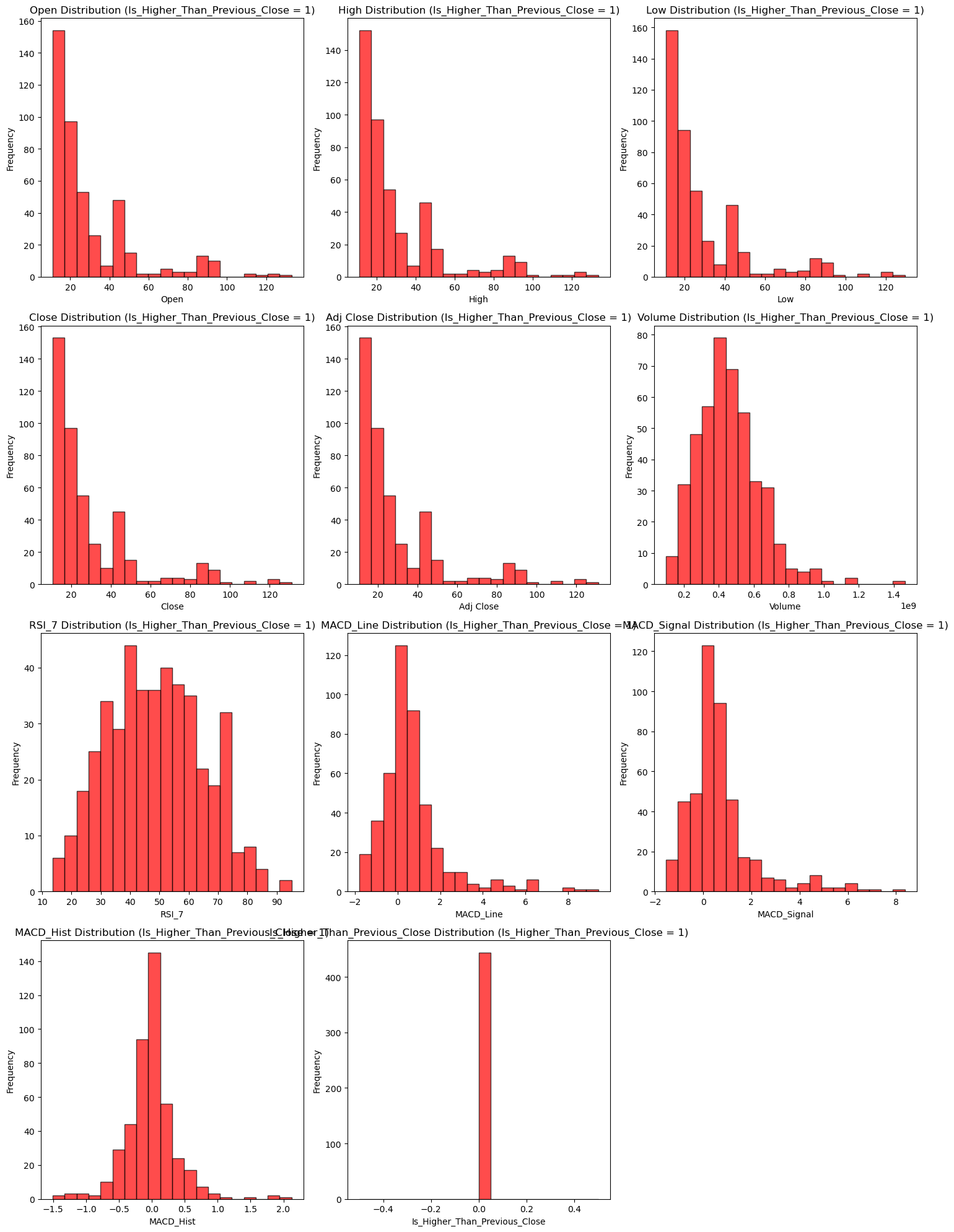




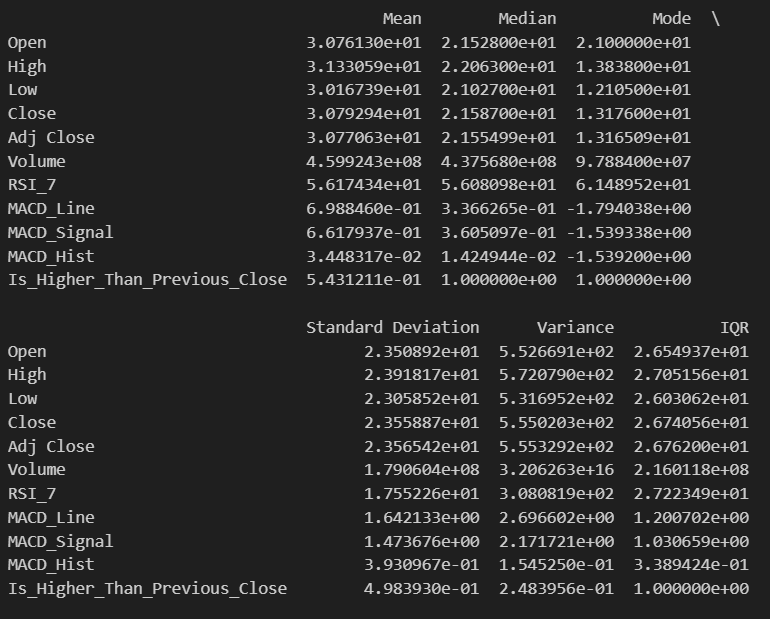
**Data Summary of the rows with closing price higher than previous day:**



**Data Summary of the rows with closing price not higher than previous day:**



**Statistics:**



## Answers to questions

<answers the questions below, referring to the screenshots above and providing an analysis of the results>

### Are the classes in the dataset balanced, or does one class (or several classes) prevail?

### For both the over distribution and the separate distributions of each class:

### 1. Price distributions (Open, High, Low, Close, Adj Close): all skewed right, with most of the values are lower values, between (0, 40),

### 2. Volume distribution: skewed right, with most of the values between (0.2, 0.6)

### 3. RSI distribution: relatively normal distribution

### 4. MACD (Line, Signal, Hist): skewed right

### 5. MACD Histogram: normal distribution

PS: The distribution of price related columns are skewed right; it has something to do with the Nvidia stock pricing going insanely up recently.

### 6. Is\_Higher\_Than\_Previous\_Close: relatively even distribution

### Does the visual representation of the data allow you to see the structure of the data?

### I have tried to test out different combinations, but the datapoints are interconnceted. There seems to be no apparent classes seperated.

### I guess this is the nature of financial data such as stock price data.

### The hidden patterns would be recognized with Neuron Networks, and since it is time-serial data, LSTM or GRU would be good.

### How many data groupings can be identified by studying the visual representation of the data?

So far, I have found no clear boundaries between the two classes. It may have some hidden patterns, but not apparently recognizable.

### Are the identified data groupings close to each other or far from each other?

No.

## Conclusions arising from the analysis of statistical indicators

Central Tendency (Mean, Median, Mode):

For Price related columns (Open, High, Low, Close, Adj Close):

1. The mean values for all price related columns are relatively close to each other, indicating that the stock price is relatively stable in shorter time frame.
2. The median and the mode are both slightly lower than the mean, suggesting that the data is right skewed, which can also be found in the distribution graph.

For Volume:

1. Mean is significantly higher than mode, indicating that there are some outliers with very high volume

For Technical Indicators:

1. RSI\_7: mean and median are close (around 56), which says the stock price is balanced between overbought and oversold. (RSI over 70 is overbought and under 30 is oversold)
2. MACD Related: MACD\_Hist, the mean, median and mode are far from each other, indicating that the stock is relatively volatile.

For Is\_Higher\_Than\_Previous\_Close:

1. Mean is 0.543. indicating that the stock has a slightly higher probability for the close price being higher than the previous day

Dispersion (Standard Deviation, Variance, IQR):

For Price related columns (Open, High, Low, Close, Adj Close):

1. Standard deviation is relatively high, meaning the daily values are very scattered.
2. Variance is quite large for all these features, which is in line with standard deviation.
3. The IQR values are similar in price related columns.

For volume:

1. Standard deviation and variance are high, indicating there are some days with considerable higher trading volume.

Overall Conclusion:

1. The price volatility is relatively balanced, most of the days, the stock is neither overbought nor oversold.
2. There are certain days the volume is considerably higher than other days
3. Over half of the time, the closing price is higher than that of the previous day.

# Part II

<this subsection should describe the use of unsupervised machine learning algorithms, accompanied by screenshots and references to the information sources used>

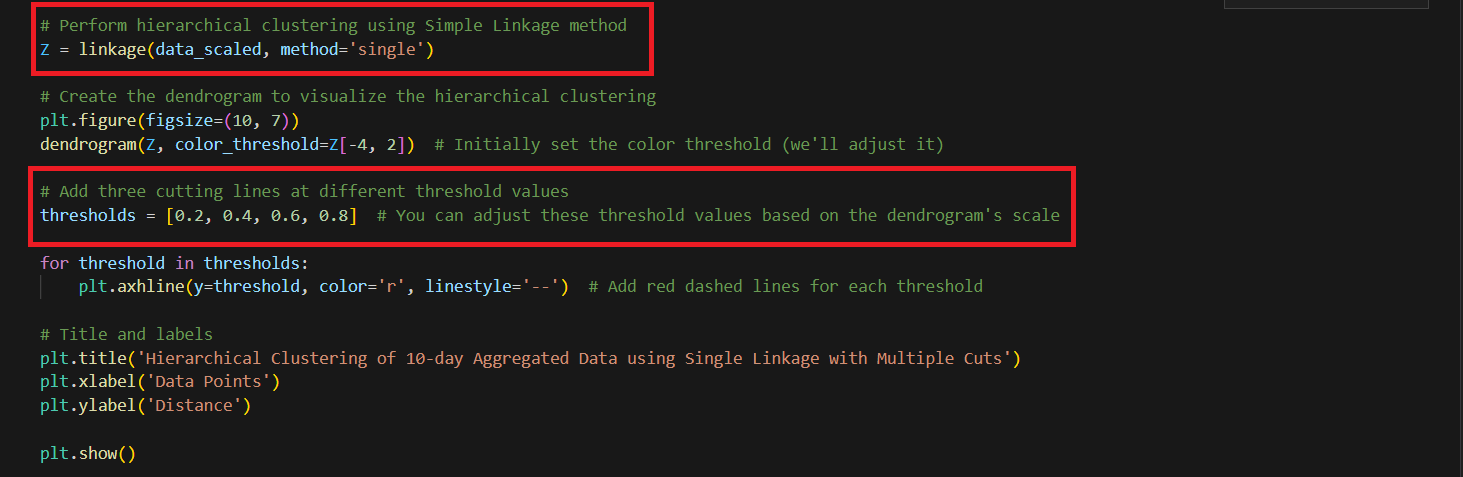
## Hierarchical clustering

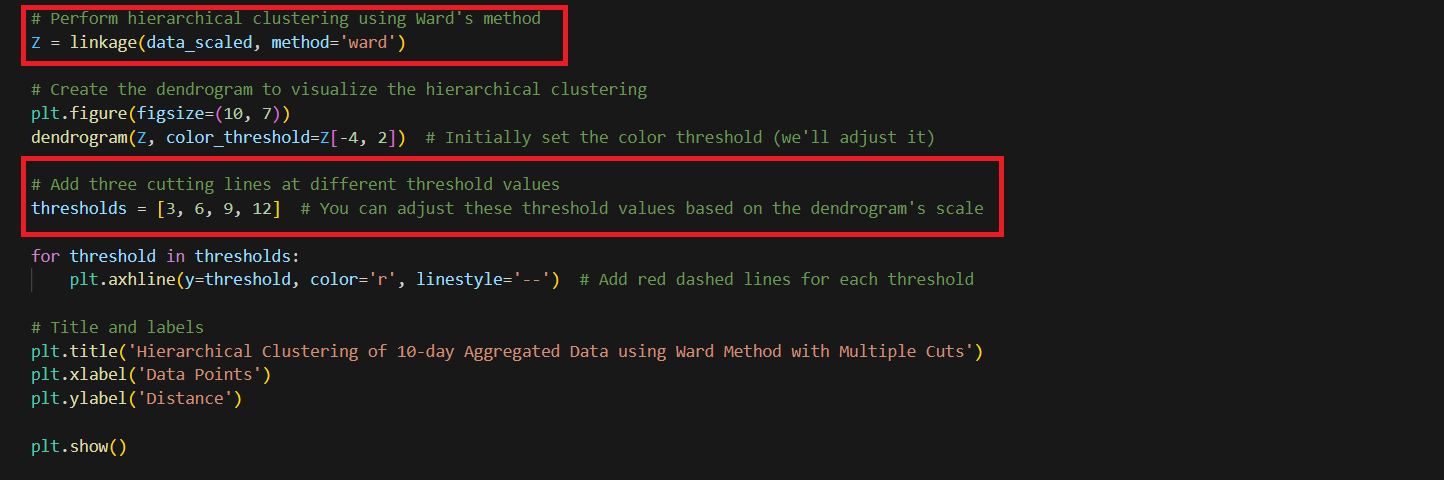
### Hyperparameters available in the Orange tool:

<add rows to table as needed>

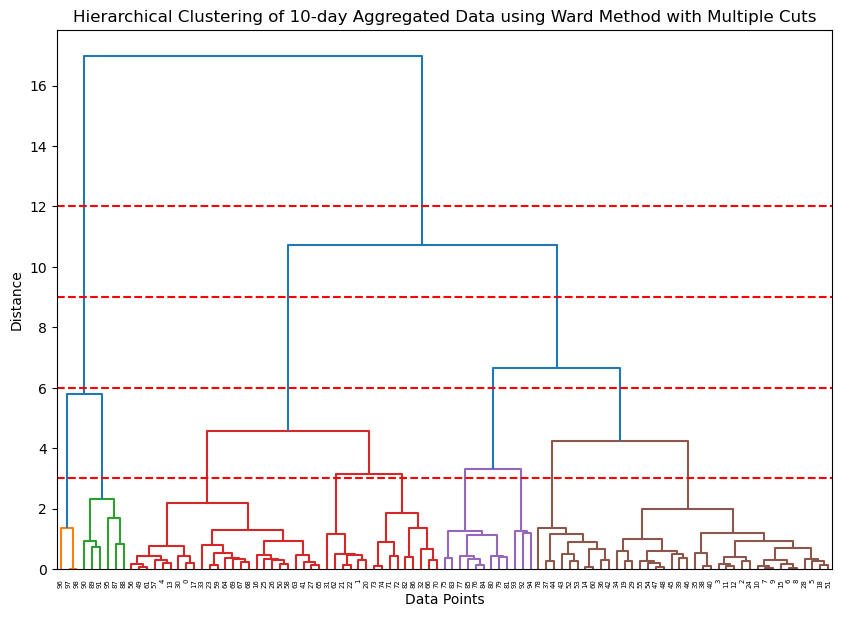
|  |  |
| --- | --- |
| **Hyperparameter** | **Description** |
| Distance Threshold | A way of controlling thee number of clusters by determining when to stop merging clusters |
| Linkage Method | Linkage method determines how the distances between clusters are calculated. |

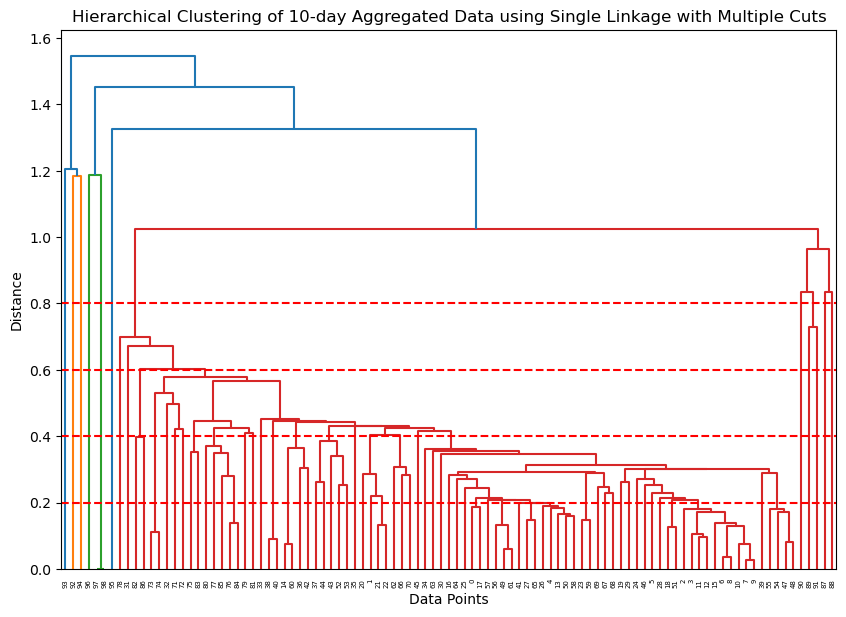
<a screenshot with hyperparameter values set for the algorithm>





### Description of experiments





### Conclusions from experiments:

Different Linkage Methods may affect the shapes of clusters. Even with the same dataset, the result are different.

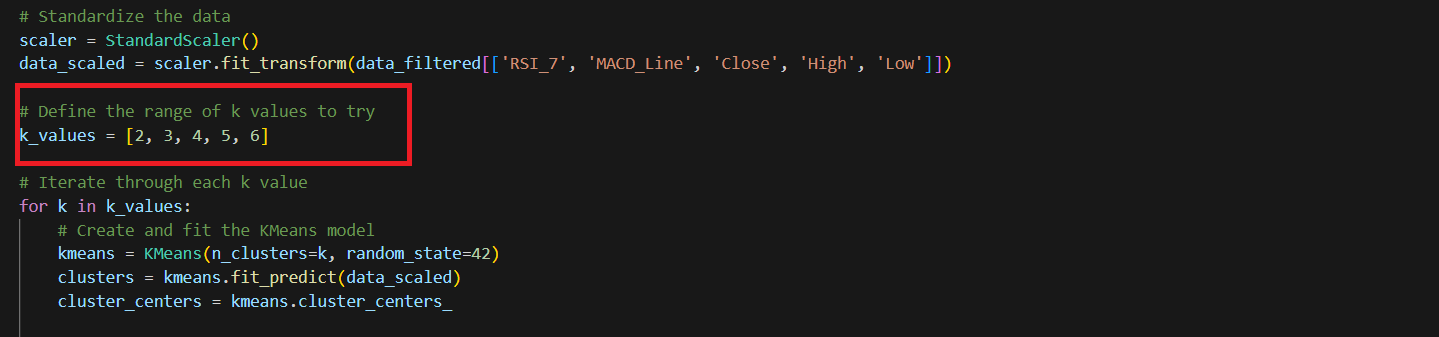
With Higher cut-off line, the less clusters

## K-means algorithm

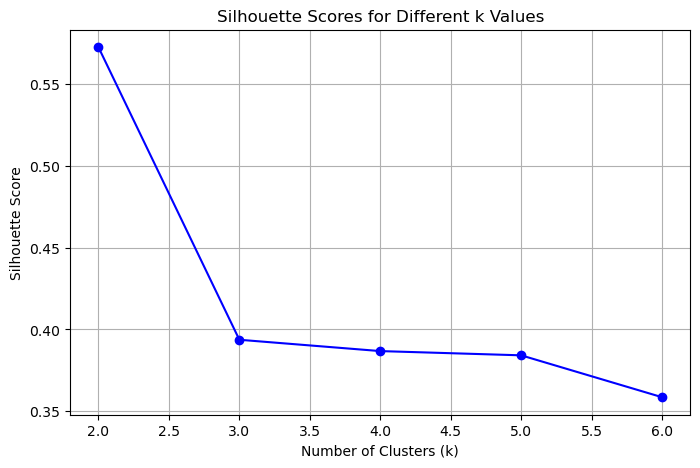
### Hyperparameters available in the Orange tool:

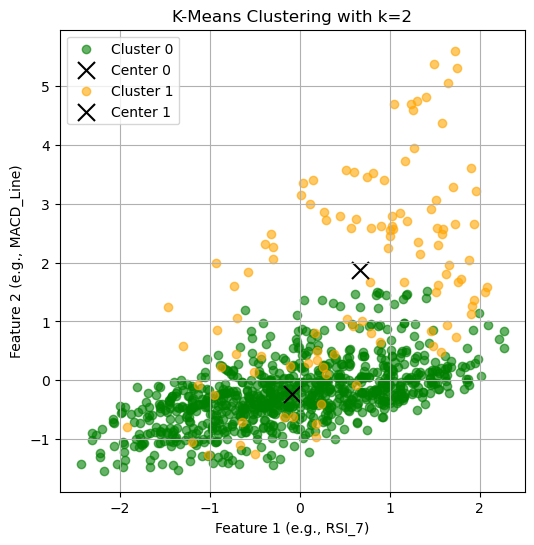
<add rows to table as needed>

|  |  |
| --- | --- |
| **Hyperparameter** | **Description** |
| K for n\_clusters | Different k will affect the number of clusters in the clustering process |
|  |  |



### Description of experiments





### Conclusions from experiments:

The K-mean clustering with 2 clusters looks fine to me. It seems that the centroid of cluster 0 and 1 are not close to each other, but two clusters are intersected.

It seems that the data are interconnected and using simple clustering methods like Hierarchical clustering and K-mean are not going to solve the issue. But this may have something to do with the nature of financial data. Therefore, more complicated methods like neuron networks are required.

## Final conclusions

The classes are not properly separated based on the previous attempts.

This may be because of the nature of financial data. Price movements is random and can be affected by many factors, therefore, it may not have apparent patterns.

But methods like LSTM neuron networks may solve the issue because it is good for processing time serial data and with more neurons, it may be able to capture the underlying patterns hidden behind the dataset.

# Part III

<this subsection should describe the use of supervised machine learning algorithms, accompanied by screenshots and references to the information sources used>

## Description of the selected algorithms

<a description of freely chosen algorithms and the rationale for their choice (except artificial neural network)>

### Title of the first algorithm: Logistic Regression

### Description of the first algorithm:

### For this algorithm, I used all features as input features, tested with different hyper parameters:

### c\_values= [0.1, 1, 10]

### penalties = [ ‘l2’, ‘l1’]

All the models trained are exported and stored in LR\_Model folder

### Title of the second algorithm: Random Forest

### Description of the second algorithm:

For this algorithm, I used all features as input features, and tested with different hyperparameters:

max\_depth\_values = [5, 19, 15]

min\_samples\_leaf\_values = [2, 4]

## Description of hyperparameters

<a description of the hyperparameters available in the Orange tool should be given for each of the algorithms, adding rows to the table as necessary>​​​​

|  |  |
| --- | --- |
| **Hyperparameter** | **Description and values** |
| Artificial Neural Networks - LSTM | |
| Learning rate | Learning Rate controls how much the algorithm updates its weights when in backpropagation  Values: [0.001, 0.002] |
| Epochs | Epoch is when the algorithm processes all datapoints in the training set for one time.  The number of epochs controls how many times the model trains.  Value: [50, 100]  PS: Since I implemented Early Stopping to avoid overfitting, the epoch is really not that important here. |
| Batch size | Barch size is the number of samples passed to the algorithm. After processing this amount of data, the model would update the weights.  Larger batch size is better than smaller size because it allows the model to process more data for weight updating  However, smaller batch size can reduce the VRam requirements if you train on low performance GPU.  Value: [32, 64] |
| I also tried two different structures, one with less layers and each layer with less neurons. But I don’t want to make this report complicated, so please refer to this link:  <https://github.com/JesseLau24/Fengdi_Huang_Second_PA/blob/main/NVDA_Stock_Prediction_Fengdi_Huang_231AHG003_Second_Practical_Task.ipynb> | |
| Logistic Regression | |
| C (Inverse of Regularization Strength) | C balances the regularization and the possibility of overfitting.  Larger c allows the model to prioritize minimizing the training error at the cost of higher risks of overfitting.  Smaller c helps prevent overfitting but may lead to larger training errors and affect the performance of the model  Value: [0.1, 1. 10] |
| Penalities | Determines the types (l1, l2, elasticnet or none) of regularization for Logistic regression algorithm.  Value: [‘l2’, ‘l1’] |
| Random Forest | |
| Max depth values | It determines the total number of splits a tree has from root to the leaf.  Smaller max depth can give us a simpler model, but the performance may not be ideal.  Larger max depth can give us a more complex model, but at the risk of overfitting.  Value: [5, 10, 16] |
| Min sample leaf values | It determines the minimum number of samples in a leaf node.  Larger values would prevent the trees from growing when the samples in a node starts becoming smaller than the value of min sample leaf, therefore, making the tree structure simpler.  Smaller values, on the other hand, would allow the tree to grow deeper and can have better performance, though may be at risk of overfitting.  Value: [2, 4] |

## Information about test and training datasets

### 

### **PS: Here, since the data is time serial data, the split would be to use the first 75% dates for training and use the last 25% dates as testing set.**

### Number of data objects in the training dataset:

Training data size: 730, the first 75%

Testing data size: 244, the last 25%

### % proportion of data objects in the training dataset:

Training data size: 730, the first 75%

Testing data size: 244, the last 25%

|  |  |  |
| --- | --- | --- |
| **Class label** | **Number of data objects in the training dataset** | **% proportion of data objects in the training dataset** |
| 1 (is higher than previous close) | 388 | 53.15% |
| 0 (is not higher than previous close) | 342 | 46.85% |

### Number of data objects in the test dataset:

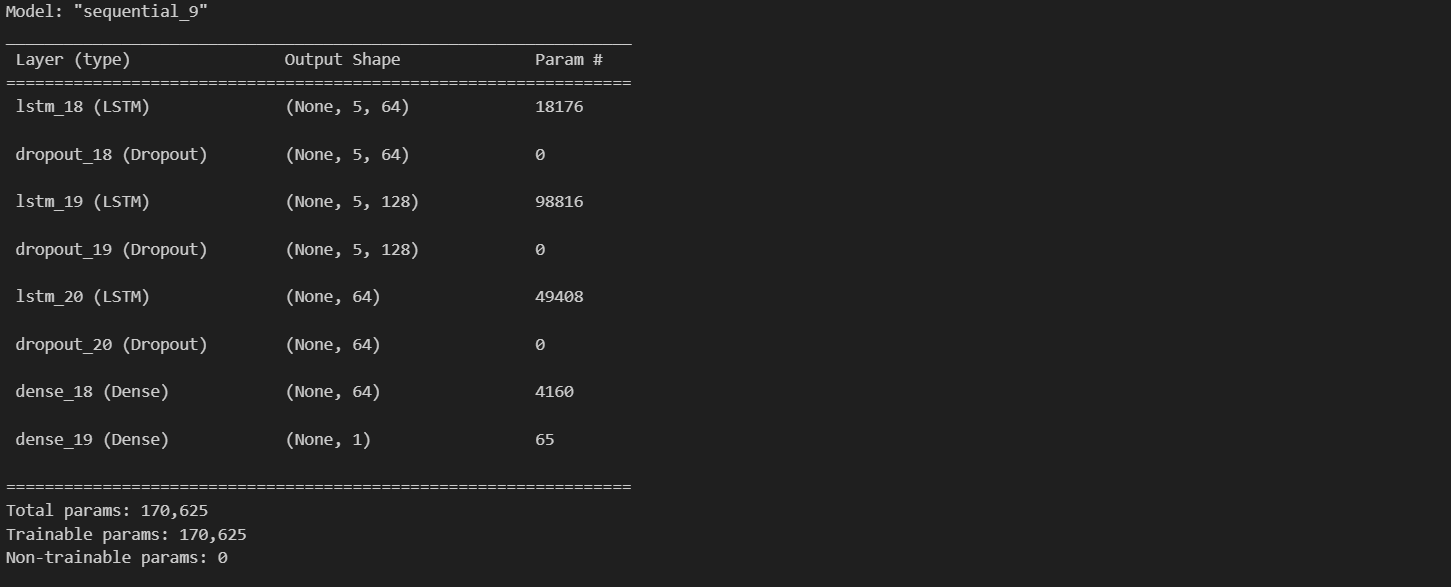
### % proportion of data objects in the test dataset:

<add rows to table as needed>

|  |  |  |
| --- | --- | --- |
| **Class label** | **Number of data objects in the test dataset** | **% proportion of data objects in the test dataset** |
| 1 (is higher than previous close) | 142 | 58.20% |
| 0 (is not higher than previous close) | 102 | 41.80% |

## Experiments with artificial neural network - LSTM

Here, I used the data from my second LSTM model with the following structure:



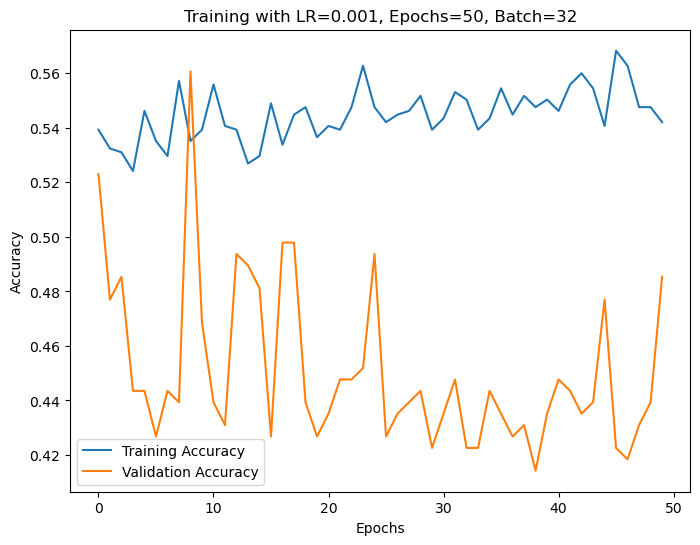
There is also a simpler model with just 2 LSTM layers with less neurons each layer.

Check out to see more results:

<https://github.com/JesseLau24/Fengdi_Huang_Second_PA/blob/main/NVDA_Stock_Prediction_Fengdi_Huang_231AHG003_Second_Practical_Task.ipynb>

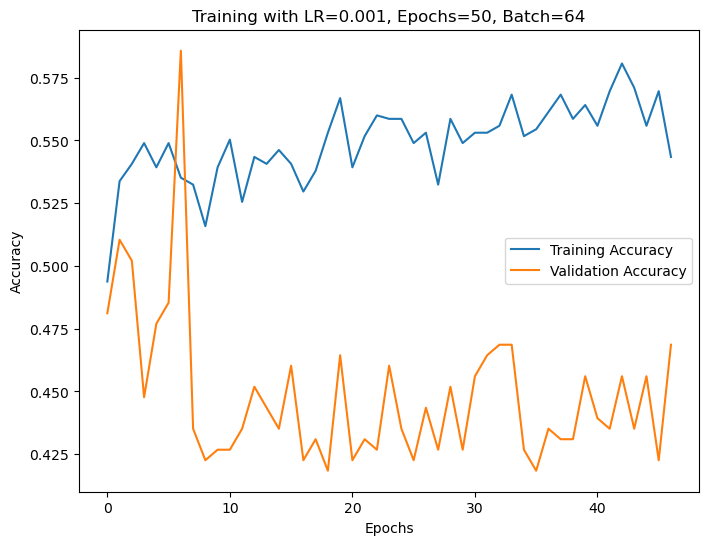
|  |  |
| --- | --- |
| **Experiment** | **Hyperparameter values** |
| Experiment 1 | LR = 0.001, Epochs = 50, Batch = 32 |
| Experiment 2 | LR = 0.001, Epochs = 50, Batch = 64 |
| Experiment 3 | LR = 0.002, Epochs = 50, Batch = 32 |

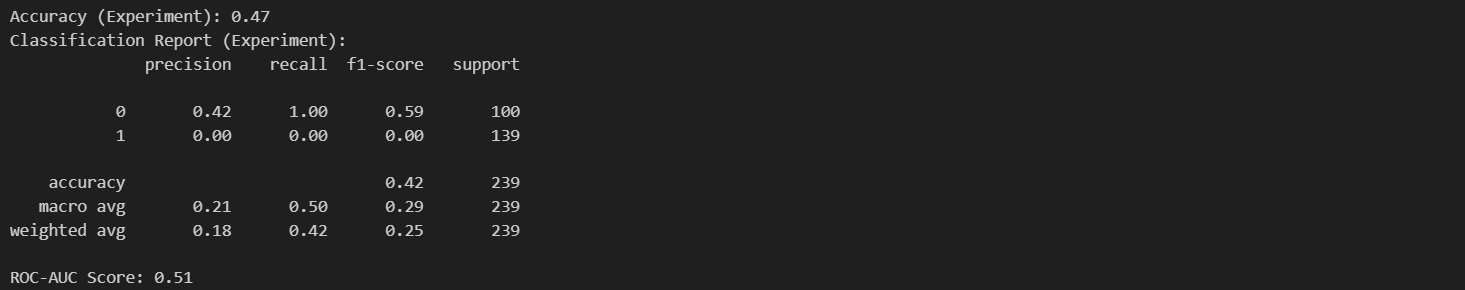
Experiment 1:



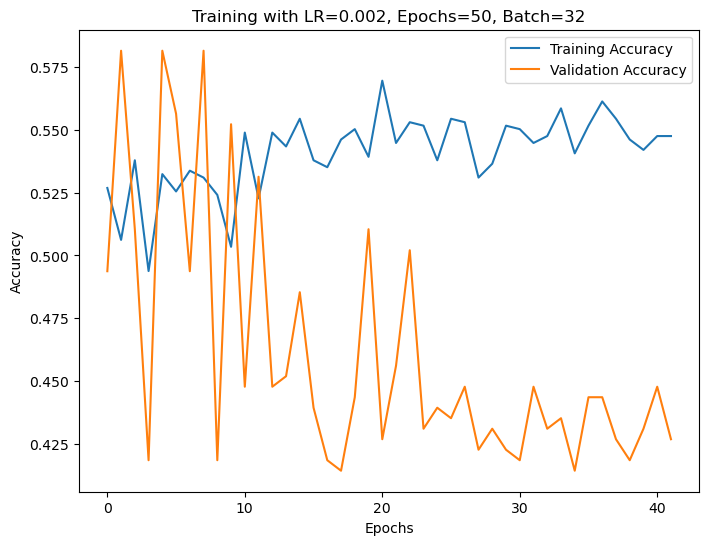


Experiment 2:





Experiment 3:





Conclusions from experiments:

1. If we don’t implement measures like Step Decay or Cyclic Learning Rates, and only consider this simple model, lower learning rates gives better results
2. Batch size doesn’t matter that much, or perhaps, I should not double the batch size, but to increase by 10 times to see the results
3. Adding more layers and increase the complexity of the model did help with better results

### Model selected for testing:

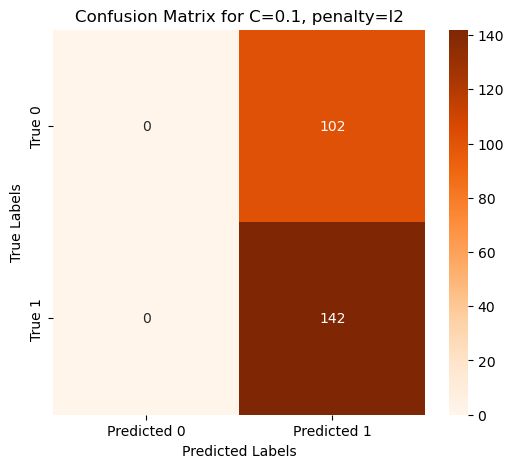
Experiment 1

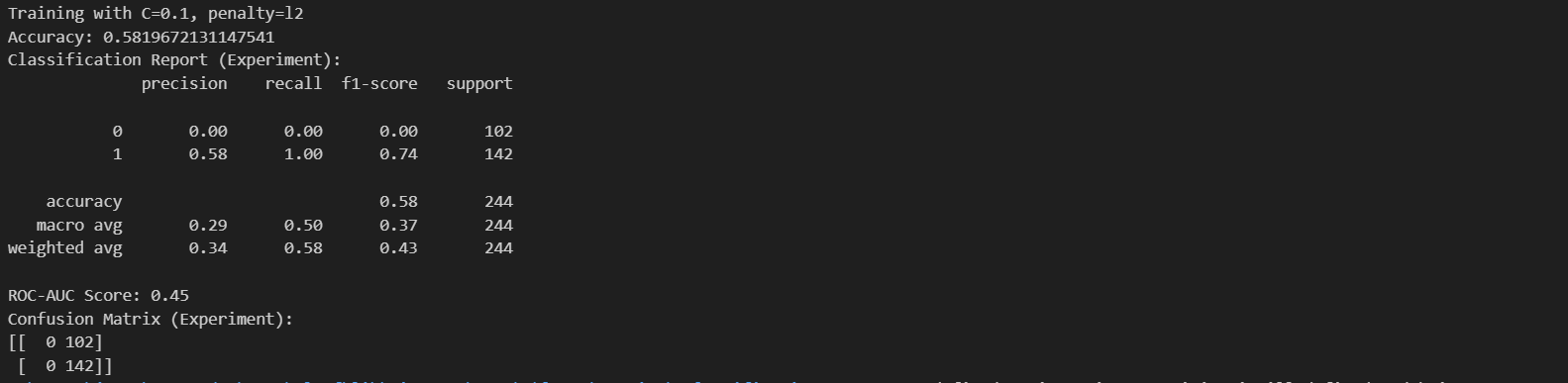
## Experiments with Logistic Regression

<add rows to table as needed>

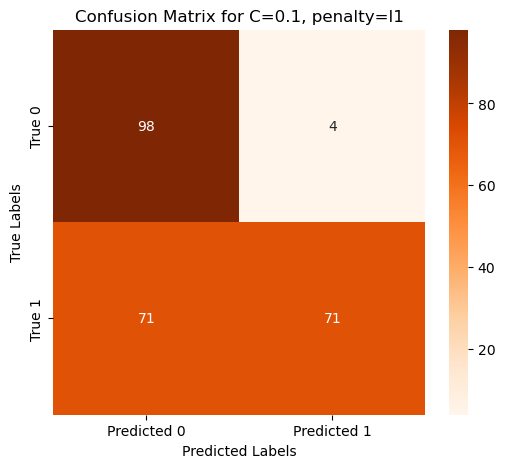
|  |  |
| --- | --- |
| **Experiment** | **Hyperparameter values** |
| Experiment 1 | C=0.1, penalty=l2 |
| Experiment 2 | C=0.1, penalty=l1 |
| Experiment 3 | C=10, penalty=l2 |
| Experiment 4 | C=10, penalty=l1 |

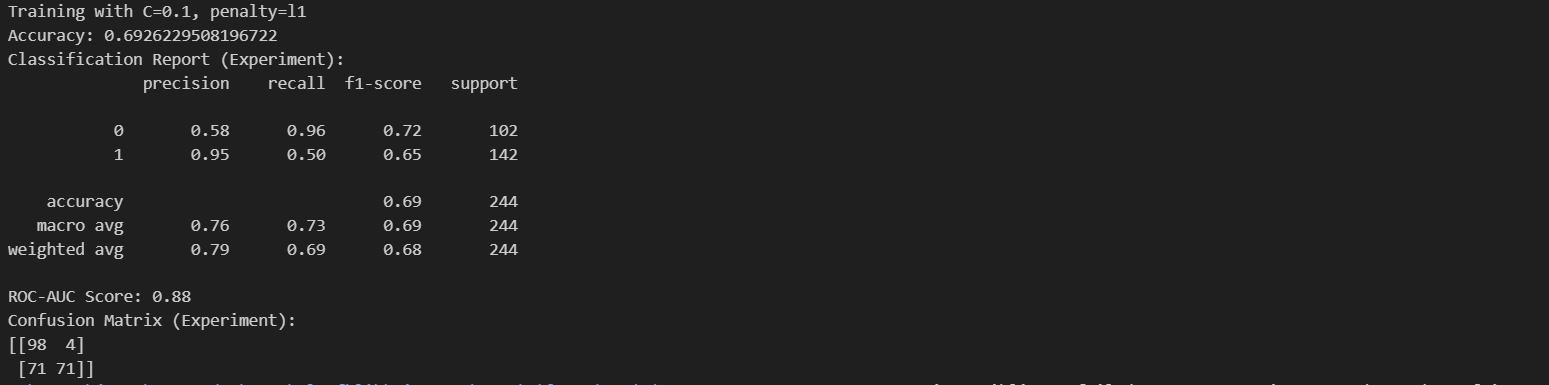
Experiment 1:



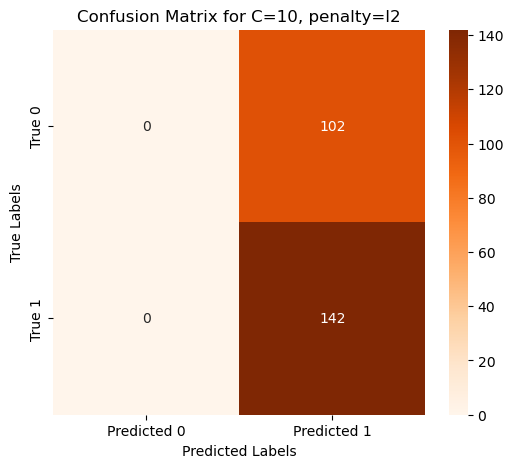


Experiment 2:



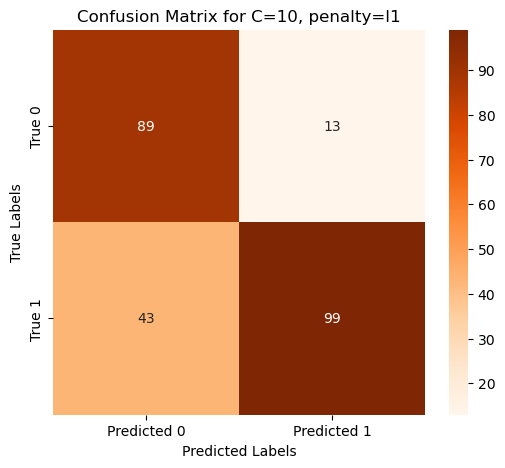


Experiment 3:





Experiment 4:





Conclusions from experiments:

From the 4 different experiments we can see that:

1. For C value: Increasing c value would improve accuracy on testing data set when using L2 as penalty, but this is not true for using L1 as penalty.
2. For Penalty: L1 performs better overall, at least in this testing data set.

I might need to experiment with more

### Model selected for testing:

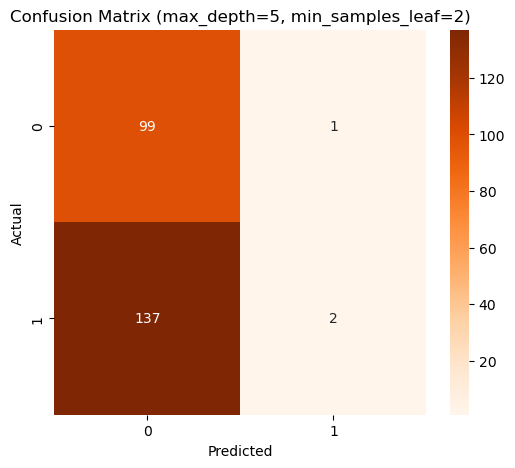
Experiment 4

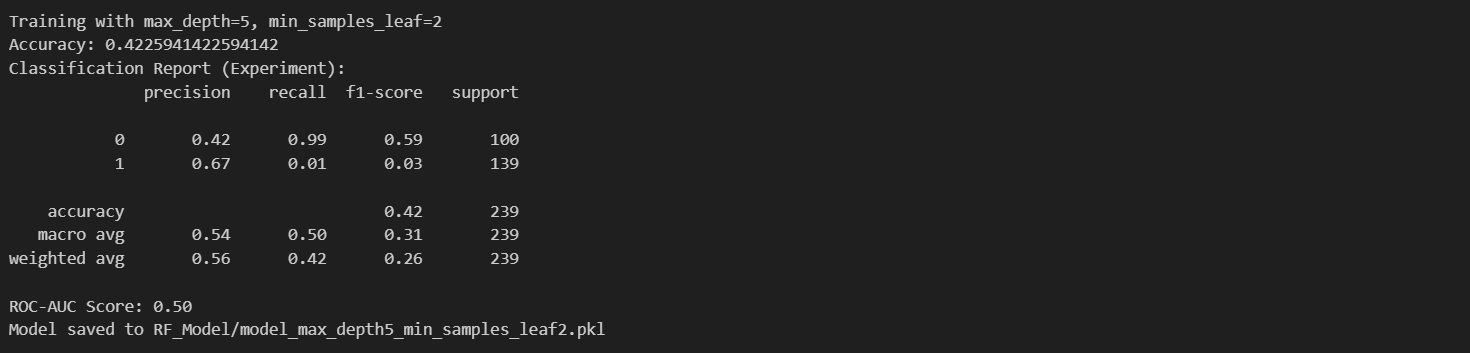
## Experiments with Random Forest

<add rows to table as needed>

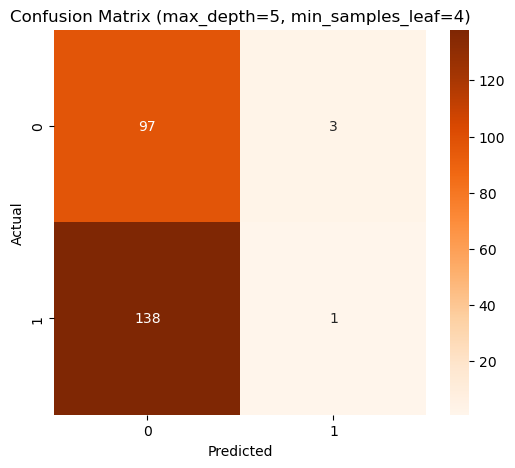
|  |  |
| --- | --- |
| **Experiment** | **Hyperparameter values** |
| Experiment 1 | max\_depth=5, min\_samples\_leaf=2 |
| Experiment 2 | max\_depth=5, min\_samples\_leaf=4 |
| Experiment 3 | max\_depth=10, min\_samples\_leaf=2 |
| Experiment 4 | max\_depth=10, min\_samples\_leaf=4 |

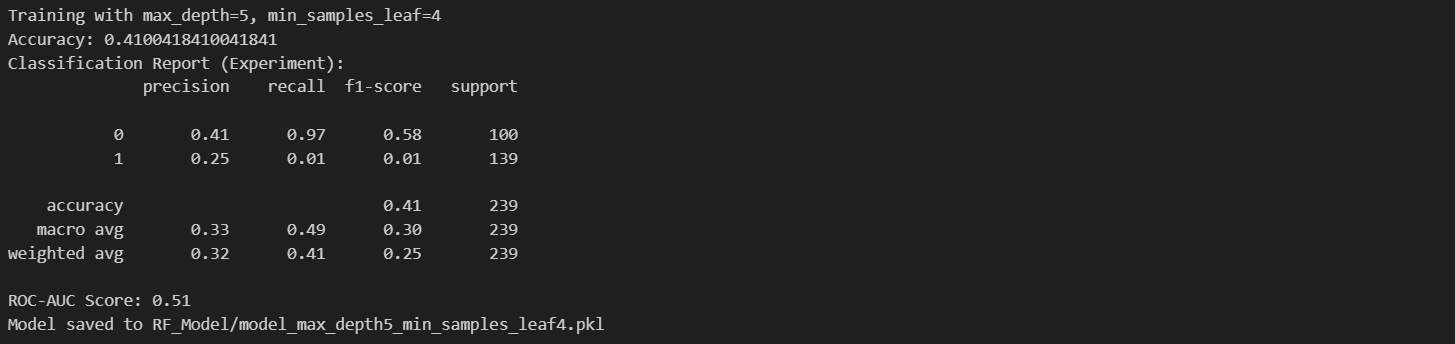
Experiment 1:



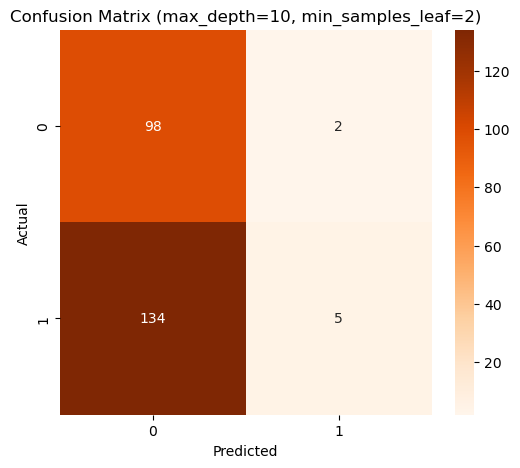


Experiment 2:



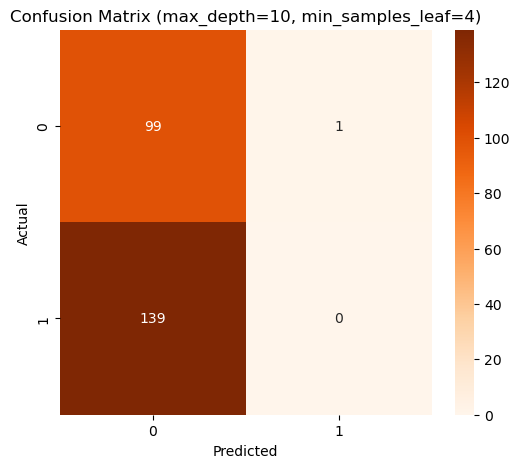


Experiment 3:





Experiment 4





Conclusions from experiments:

From above we can find that:

1. All four experiments have lower accuracy (around 41% to 43%), which means that the model is performing poorly despite we tried different hyperparameters.
2. From the confusion matrix, we can see that the model tend to have higher accuracy when predicting 0 (not higher than pervious close)

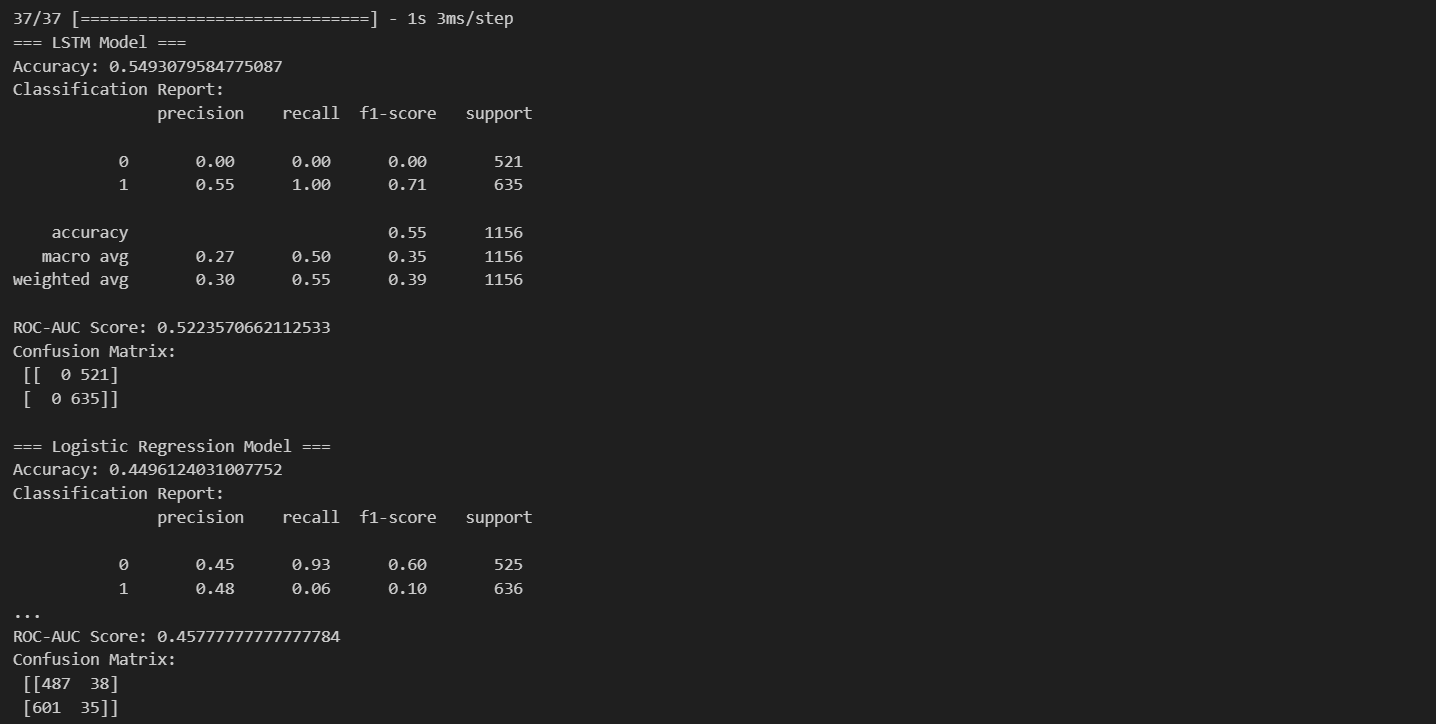
Perhaps the data is slightly imbalanced (as we can see from data summary) and this has affected the Random Forest.

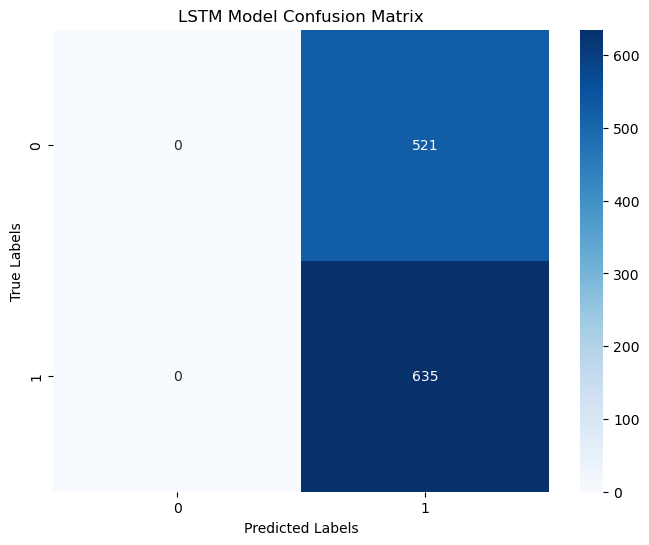
### Model selected for testing:

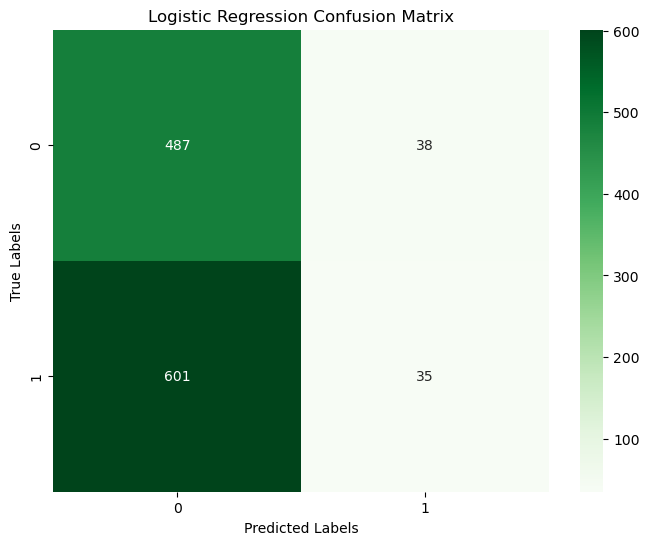
I am not selecting this model for testing because I can pretty much be sure of the results.

And the models have some issues when testing, still trying to fix it.

## Testing results of the trained models







Conclusions after testing:

For the LSTM model:

1. The model Classification is Biased. The confusion matrix shows that the LSTM model predicts all samples as class 1 (positive)
2. From the recall and precision for class 0 are both 0. This is not good
3. The ROC-AUC score is 0.522, which is close to o,6, which means the using this model is just slightly better than tossing a coin.

For Logistic Regression:

1. The model has some issues with predicting classification 1 (positive), with por recall and F1 score for classification 1.
2. ROC-AUC score is even lower than LSTM.

Conclusion: Both models are poor. But that could be due to the lack of available data.

# Information sources

Database:

<https://www.kaggle.com/datasets/programmerrdai/nvidia-stock-historical-data>

Github Repo:

<https://github.com/JesseLau24/Fengdi_Huang_Second_PA>

Géron, Aurélien. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.* 2nd ed. Sebastopol, CA: O'Reilly Media, Inc., 2019.