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:

	Date	Open	High	Low	Close	Adj Close	Volume	RSI_7	MACD_Line	MACD_Signal	MACD_Hist	Is_Higher_Than_Previous_Close
33	2020-08-05	11.24400	11.37175	11.16625	11.28675	11.253224	249924000	79.314298	0.345669	0.271446	0.074223	1
34	2020-08-06	11.34975	11.35800	11.17875	11.33550	11.301830	244316000	80.212551	0.371684	0.291493	0.080191	1
35	2020-08-07	11.31250	11.50475	11.03750	11.19950	11.166232	342516000	70.279787	0.376982	0.308591	0.068391	0
36	2020-08-10	11.33425	11.40825	10.85650	11.16500	11.131833	427796000	67.795203	0.374085	0.321690	0.052395	0
37	2020-08-11	11.07375	11.13675	10.79575	10.85000	10.817771	354512000	49.248877	0.342423	0.325836	0.016587	0
38	2020-08-12	10.99075	11.46700	10.95825	11.44025	11.406269	464412000	68.241669	0.360801	0.332829	0.027971	1
39	2020-08-13	11.54600	11.72175	11.35575	11.44300	11.409009	374460000	68.306140	0.371306	0.340525	0.030782	1
40	2020-08-14	11.53000	11.70475	11.44050	11.56400	11.529649	366436000	71.297217	0.384959	0.349411	0.035547	1
41	2020-08-17	11.85125	12.40975	11.81725	12.33700	12.300353	621300000	83.149566	0.452931	0.370115	0.082816	1
42	2020-08-18	12.45000	12.49600	12.08625	12.26075	12.224331	503448000	79.377453	0.494942	0.395081	0.099861	0
43	2020-08-19	12.29650	12.31500	12.09800	12.13850	12.102443	622624000	73.168680	0.512464	0.418558	0.093907	0

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	Float64	13.600 – 96.045

	on the data over the past 7 days		
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:

	Date	Open	High	Low	Close	Adj Close	Volume	RSI_7	MACD_Line	MACD_Signal	MACD_Hist	Is_Higher_Than_Previous_Close
33	2020-08-05	11.24400	11.37175	11.16625	11.28675	11.253224	249924000	79.314298	0.345669	0.271446	0.074223	1
34	2020-08-06	11.34975	11.35800	11.17875	11.33550	11.301830	244316000	80.212551	0.371684	0.291493	0.080191	1
35	2020-08-07	11.31250	11.50475	11.03750	11.19950	11.166232	342516000	70.279787	0.376982	0.308591	0.068391	0
36	2020-08-10	11.33425	11.40825	10.85650	11.16500	11.131833	427796000	67.795203	0.374085	0.321690	0.052395	0
37	2020-08-11	11.07375	11.13675	10.79575	10.85000	10.817771	354512000	49.248877	0.342423	0.325836	0.016587	0
38	2020-08-12	10.99075	11.46700	10.95825	11.44025	11.406269	464412000	68.241669	0.360801	0.332829	0.027971	1
39	2020-08-13	11.54600	11.72175	11.35575	11.44300	11.409009	374460000	68.306140	0.371306	0.340525	0.030782	1
40	2020-08-14	11.53000	11.70475	11.44050	11.56400	11.529649	366436000	71.297217	0.384959	0.349411	0.035547	1
41	2020-08-17	11.85125	12.40975	11.81725	12.33700	12.300353	621300000	83.149566	0.452931	0.370115	0.082816	1
42	2020-08-18	12.45000	12.49600	12.08625	12.26075	12.224331	503448000	79.377453	0.494942	0.395081	0.099861	0
43	2020-08-19	12.29650	12.31500	12.09800	12.13850	12.102443	622624000	73.168680	0.512464	0.418558	0.093907	0

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

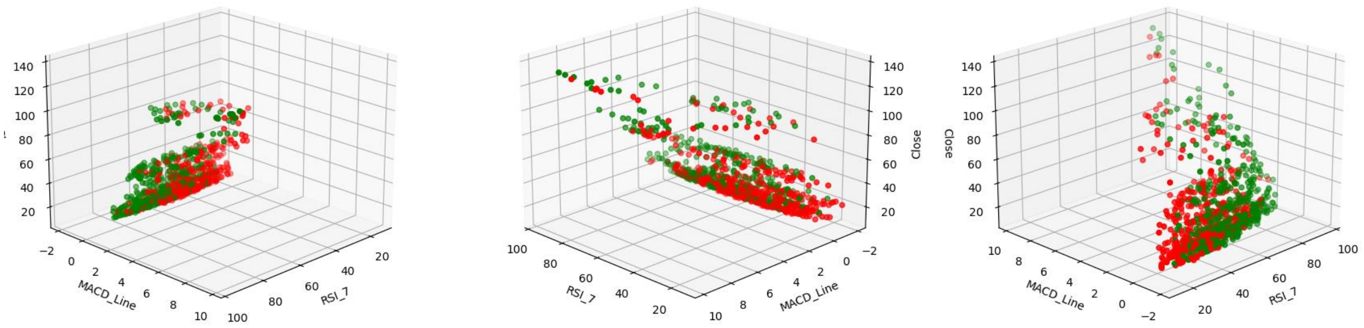
Data before dropping NaN values:

	Date	Open	High	Low	Close	Adj Close	Volume	RSI_7	MACD_Line	MACD_Signal	MACD_Hist	Is_Higher_Than_Previous_Close
0	2020-06-18	9.22700	9.28250	9.11450	9.21800	9.190620	254408000	NaN	NaN	NaN	NaN	0
1	2020-06-19	9.24250	9.44500	9.22725	9.26125	9.233741	524160000	NaN	NaN	NaN	NaN	1
2	2020-06-22	9.30000	9.53125	9.27325	9.52675	9.498451	398468000	NaN	NaN	NaN	NaN	1
3	2020-06-23	9.55100	9.64250	9.40750	9.45000	9.421929	375108000	NaN	NaN	NaN	NaN	0
4	2020-06-24	9.47625	9.55650	9.14450	9.23550	9.208067	449372000	NaN	NaN	NaN	NaN	0
5	2020-06-25	9.35575	9.50500	9.18225	9.49000	9.461810	376072000	NaN	NaN	NaN	NaN	1
6	2020-06-26	9.50000	9.50000	9.12500	9.15500	9.127805	592084000	41.356084	NaN	NaN	NaN	0
7	2020-06-29	9.16975	9.20450	8.90000	9.20000	9.172669	342248000	44.514197	NaN	NaN	NaN	1
8	2020-06-30	9.31400	9.52625	9.26650	9.49775	9.469539	367892000	60.807093	NaN	NaN	NaN	1
9	2020-07-01	9.52075	9.57575	9.41300	9.53000	9.501692	326648000	62.209344	NaN	NaN	NaN	1
10	2020-07-02	9.63900	9.73750	9.57825	9.61225	9.583698	364056000	65.845322	NaN	NaN	NaN	1

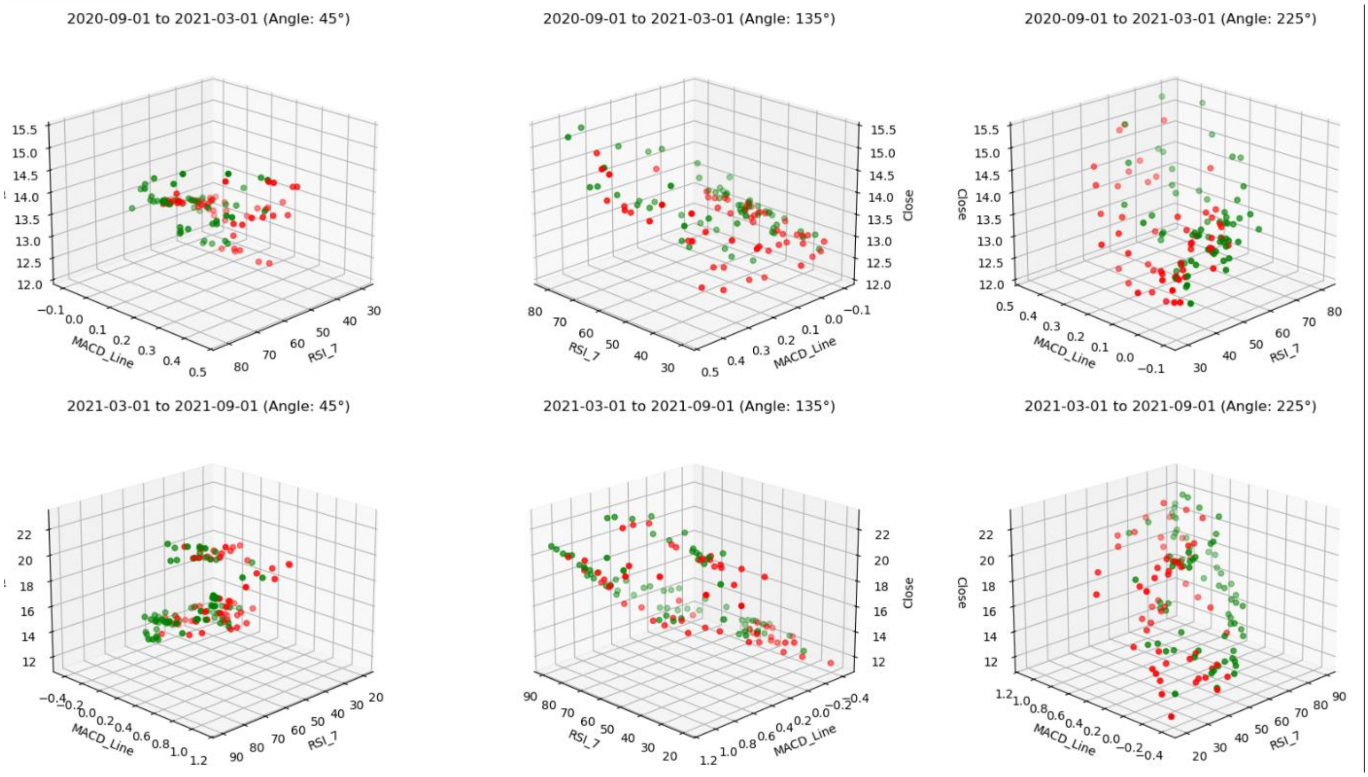
Visual and statistical representation of the dataset

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

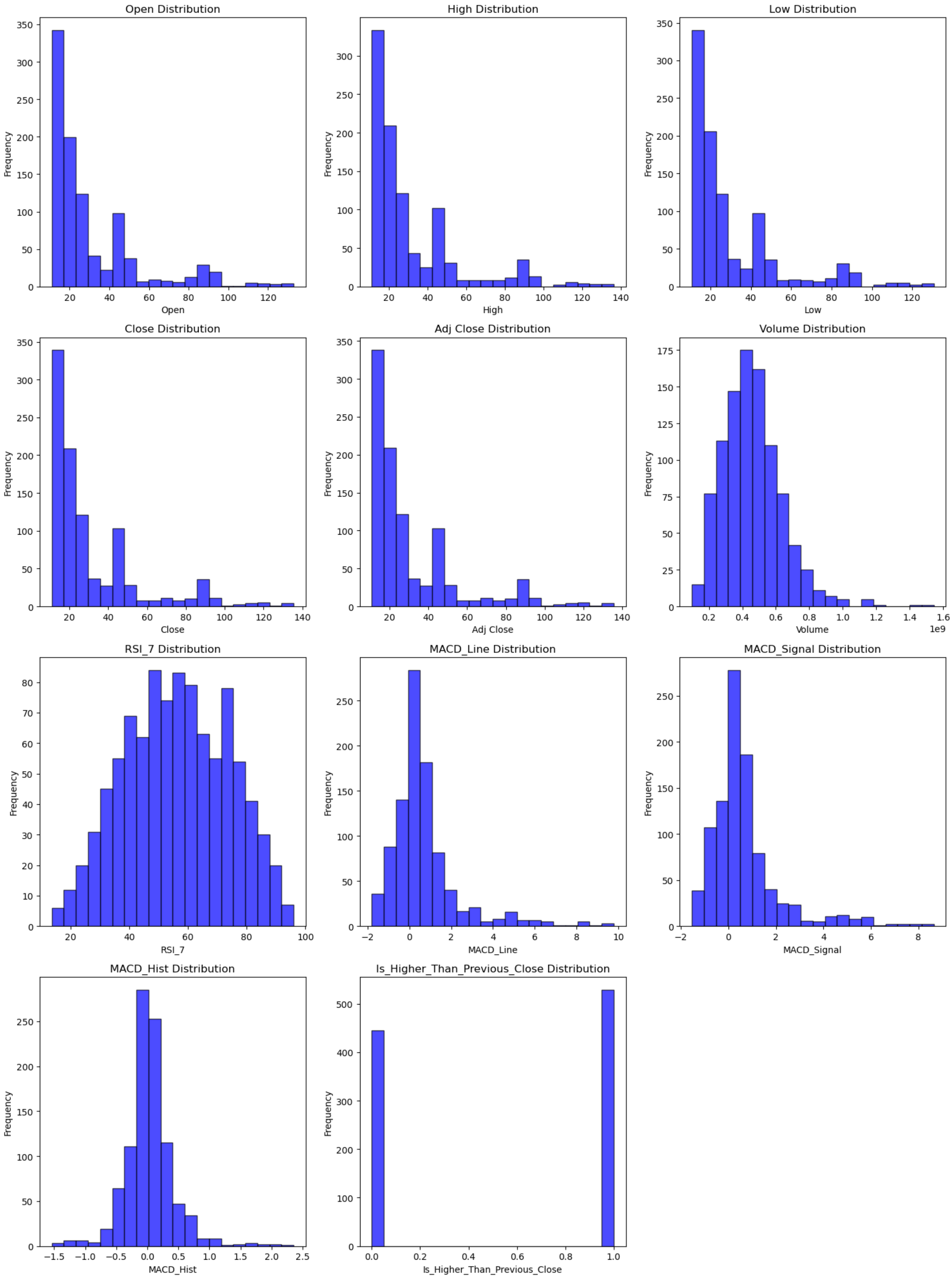
3D Scatterplot: x RSI_7, y MACD_Line, z Close Price, is higher, green, is not: red

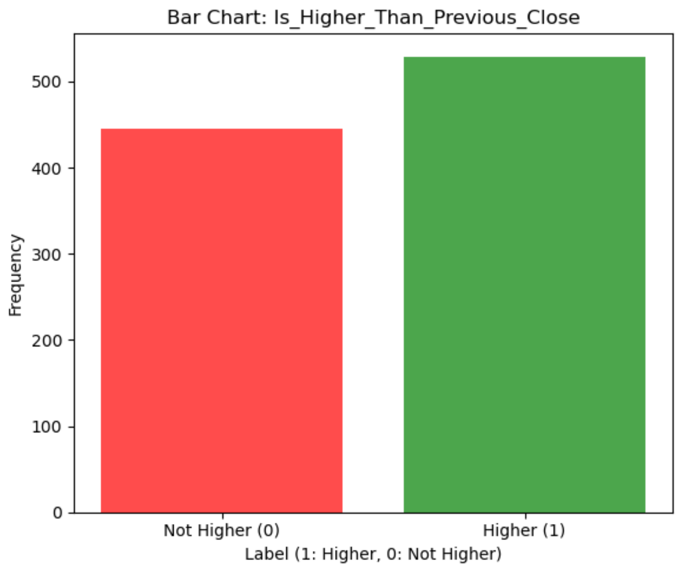


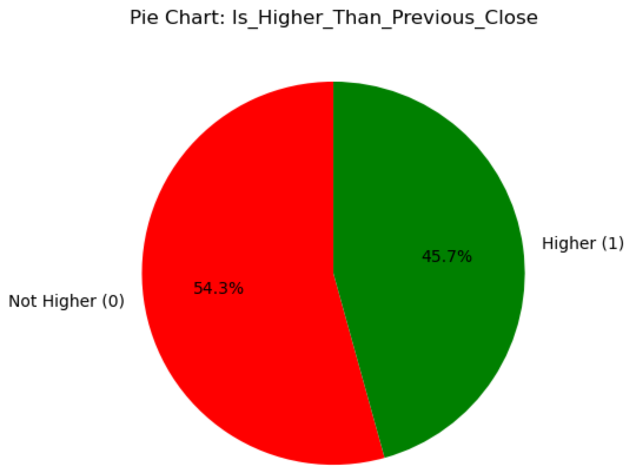
3D Scatterplot in 6-month Timeframe: x RSI_7, y MACD_Line, z Close Price, is higher, green, is not: red



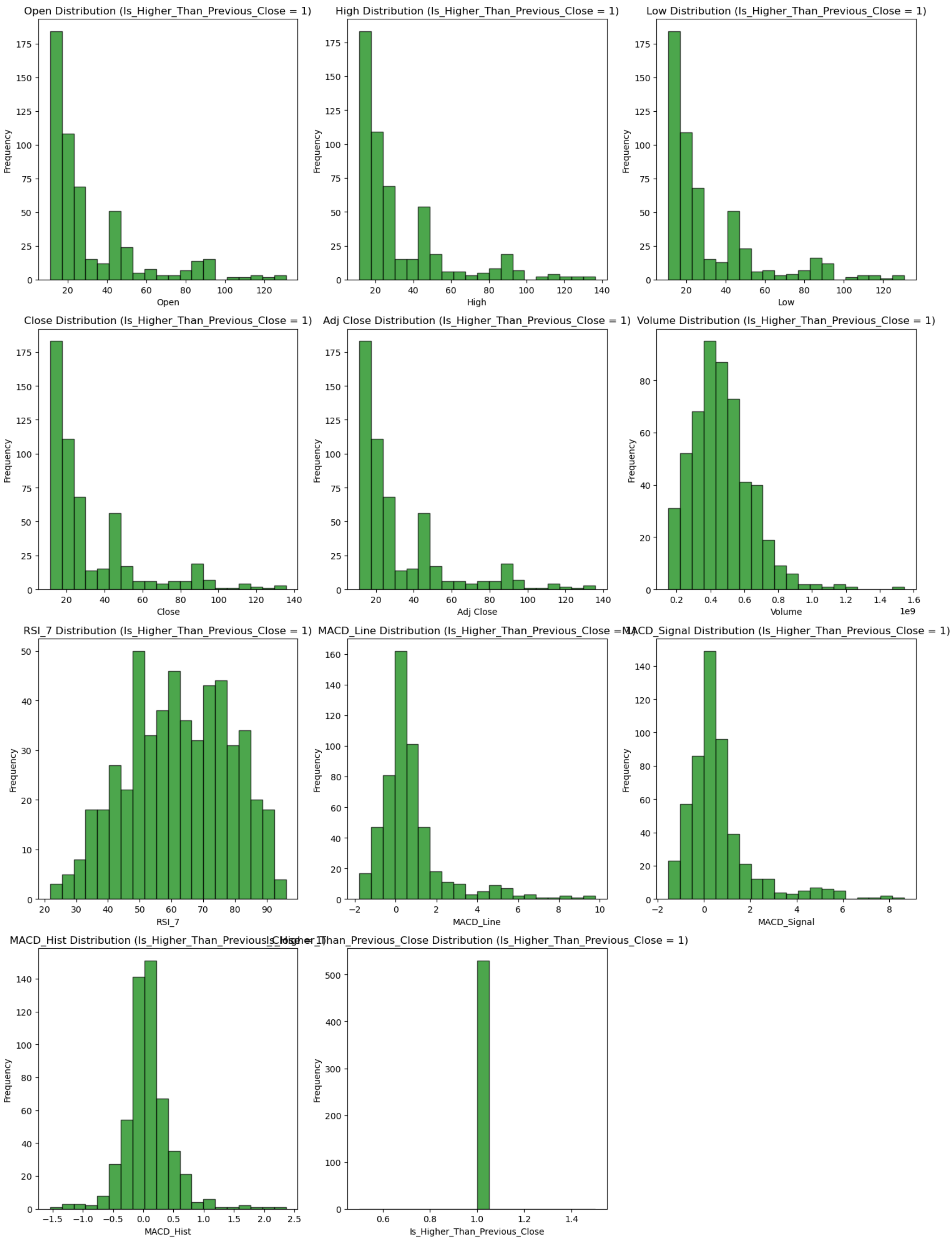
Data Summary (of all 974 rows used in this assignment):


Open Distribution
Frequency
Open
High Distribution
Frequency
High
Low Distribution
Frequency
Low
Close Distribution
Frequency
Close
Adj Close Distribution
Frequency
Adj Close
Volume Distribution
Frequency
Volume
1e9
RSI_7 Distribution
Frequency
RSI_7
MACD_Line Distribution
Frequency
MACD_Line
MACD_Signal Distribution
Frequency
MACD_Signal
MACD_Hist Distribution
Frequency
MACD_Hist
Is_Higher_Than_Previous_Close Distribution
Frequency
Is_Higher_Than_Previous_Close

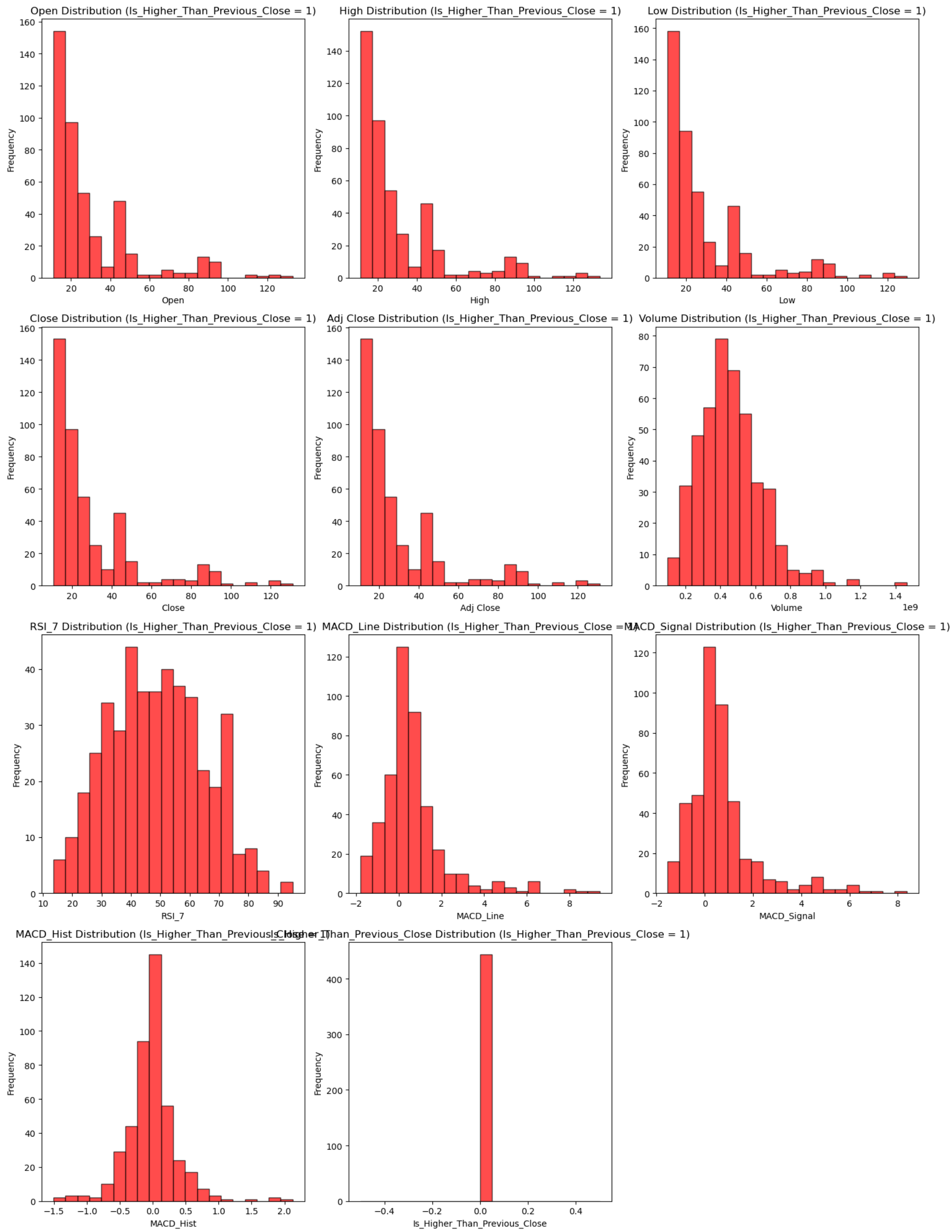




Data Summary of the rows with closing price higher than previous day:



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


Open Distribution (Is_Higher_Than_Previous_Close = 1)
High Distribution (Is_Higher_Than_Previous_Close = 1)
Low Distribution (Is_Higher_Than_Previous_Close = 1)
Close Distribution (Is_Higher_Than_Previous_Close = 1)
Adj Close Distribution (Is_Higher_Than_Previous_Close = 1)
Volume Distribution (Is_Higher_Than_Previous_Close = 1)
RSI_7 Distribution (Is_Higher_Than_Previous_Close = 1)
MACD_Line Distribution (Is_Higher_Than_Previous_Close = 1)
MACD_Signal Distribution (Is_Higher_Than_Previous_Close = 1)
MACD_Hist Distribution (Is_Higher_Than_Previous_Close = 1)
Is_Higher_Than_Previous_Close Distribution (Is_Higher_Than_Previous_Close = 1)
Frequency
Open
High
Low
Close
Adj Close
Volume
1e9
RSI_7
MACD_Line
MACD_Signal
MACD_Hist
Is_Higher_Than_Previous_Close

Statistics:

```
                                    Mean        Median          Mode  \
Open                        3.076130e+01  2.152800e+01  2.100000e+01
High                        3.133059e+01  2.206300e+01  1.383800e+01
Low                         3.016739e+01  2.102700e+01  1.210500e+01
Close                       3.079294e+01  2.158700e+01  1.317600e+01
Adj Close                   3.077063e+01  2.155499e+01  1.316509e+01
Volume                      4.599243e+08  4.375680e+08  9.788400e+07
RSI_7                       5.617434e+01  5.608098e+01  6.148952e+01
MACD_Line                   6.988460e-01  3.366265e-01 -1.794038e+00
MACD_Signal                 6.617937e-01  3.605097e-01 -1.539338e+00
MACD_Hist                   3.448317e-02  1.424944e-02 -1.539200e+00
Is_Higher_Than_Previous_Close  5.431211e-01  1.000000e+00  1.000000e+00

                            Standard Deviation      Variance           IQR
Open                              2.350892e+01  5.526691e+02  2.654937e+01
High                              2.391817e+01  5.720790e+02  2.705156e+01
Low                               2.305852e+01  5.316952e+02  2.603062e+01
Close                             2.355887e+01  5.550203e+02  2.674056e+01
Adj Close                         2.356542e+01  5.553292e+02  2.676200e+01
Volume                            1.790604e+08  3.206263e+16  2.160118e+08
RSI_7                             1.755226e+01  3.080819e+02  2.722349e+01
MACD_Line                         1.642133e+00  2.696602e+00  1.200702e+00
MACD_Signal                       1.473676e+00  2.171721e+00  1.030659e+00
MACD_Hist                         3.930967e-01  1.545250e-01  3.389424e-01
Is_Higher_Than_Previous_Close     4.983930e-01  2.483956e-01  1.000000e+00
```

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>

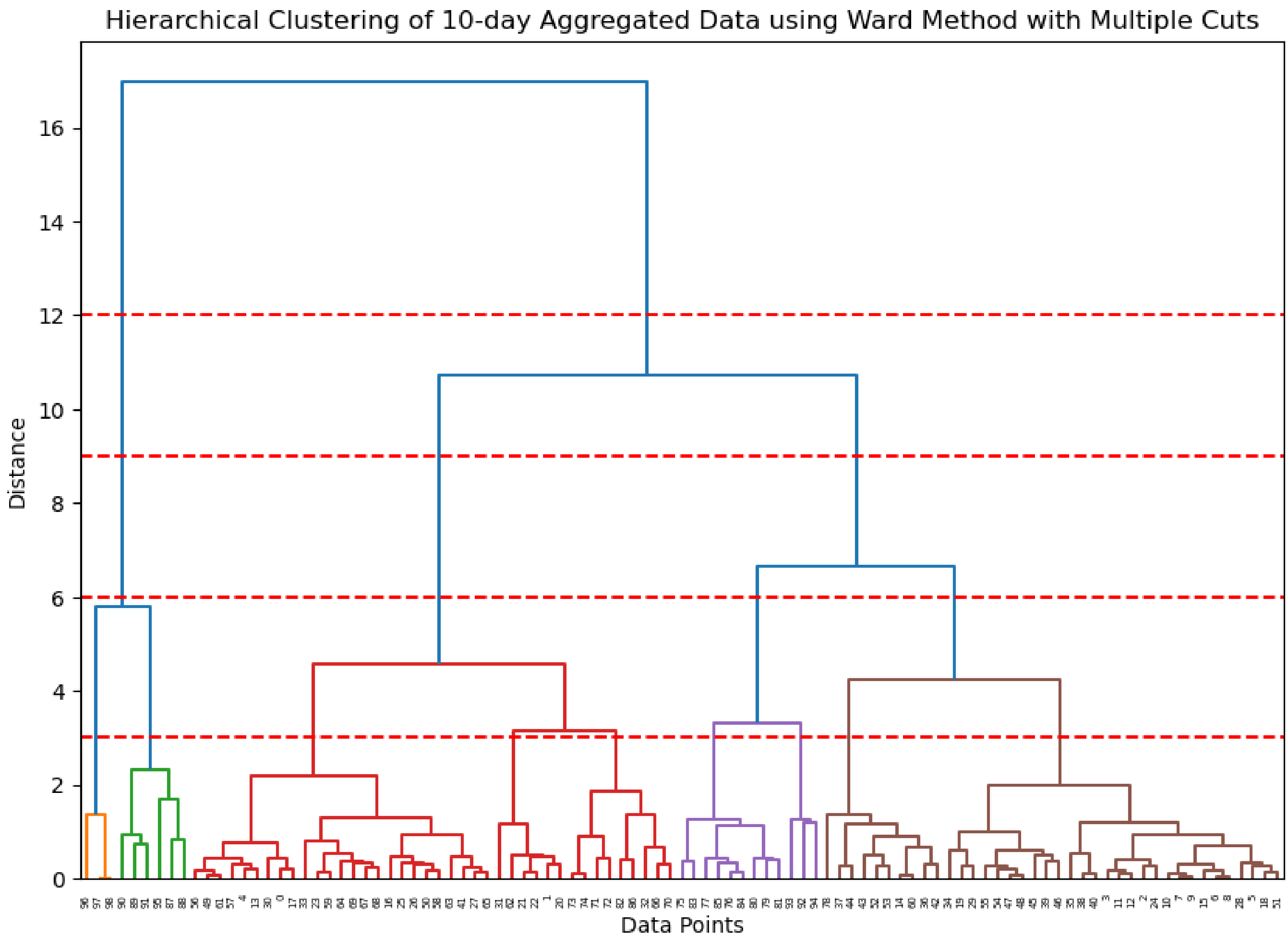
Hyperparameter	Description
Distance Threshold	A way of controlling thee number of clusters by determining when to stop merging clusters
Linkage Method	Linkage method determines how the distances between clusters are calculated.

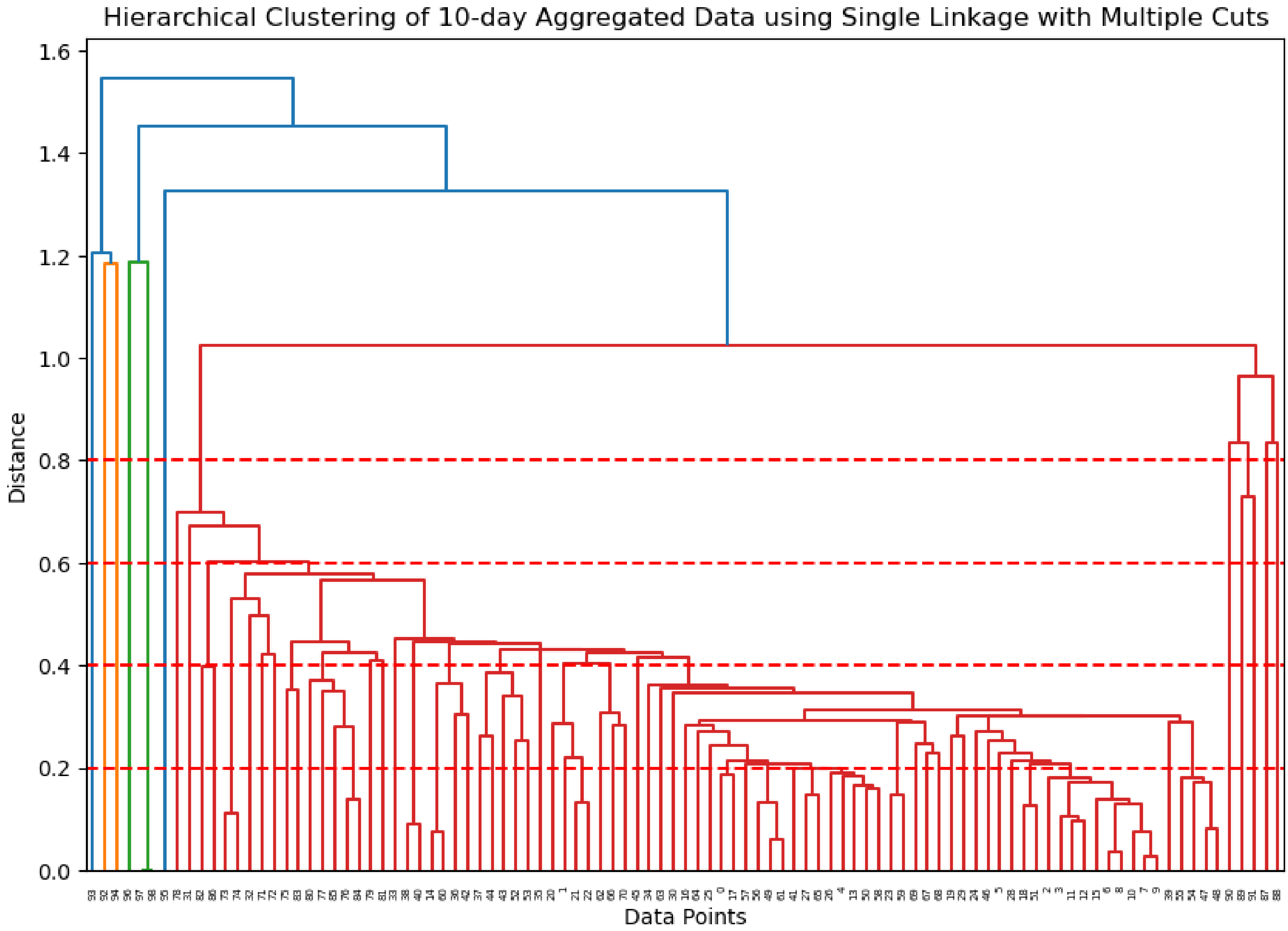
<a screenshot with hyperparameter values set for the algorithm>

```
# Perform hierarchical clustering using Simple Linkage method
Z = linkage(data_scaled, method='single')

# Create the dendrogram to visualize the hierarchical clustering
plt.figure(figsize=(10, 7))
dendrogram(Z, color_threshold=Z[-4, 2])  # Initially set the color threshold (we'll adjust it)

# Add three cutting lines at different threshold values
thresholds = [0.2, 0.4, 0.6, 0.8]  # You can adjust these threshold values based on the dendrogram's scale

for threshold in thresholds:
    plt.axhline(y=threshold, color='r', linestyle='--')  # Add red dashed lines for each threshold

# Title and labels
plt.title('Hierarchical Clustering of 10-day Aggregated Data using Single Linkage with Multiple Cuts')
plt.xlabel('Data Points')
plt.ylabel('Distance')

plt.show()
```

```python
# Perform hierarchical clustering using Ward's method
Z = linkage(data_scaled, method='ward')

# Create the dendrogram to visualize the hierarchical clustering
plt.figure(figsize=(10, 7))
dendrogram(Z, color_threshold=Z[-4, 2])  # Initially set the color threshold (we'll adjust it)

# Add three cutting lines at different threshold values
thresholds = [3, 6, 9, 12]  # You can adjust these threshold values based on the dendrogram's scale

for threshold in thresholds:
    plt.axhline(y=threshold, color='r', linestyle='--')  # Add red dashed lines for each threshold

# Title and labels
plt.title('Hierarchical Clustering of 10-day Aggregated Data using Ward Method with Multiple Cuts')
plt.xlabel('Data Points')
plt.ylabel('Distance')

plt.show()
```

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

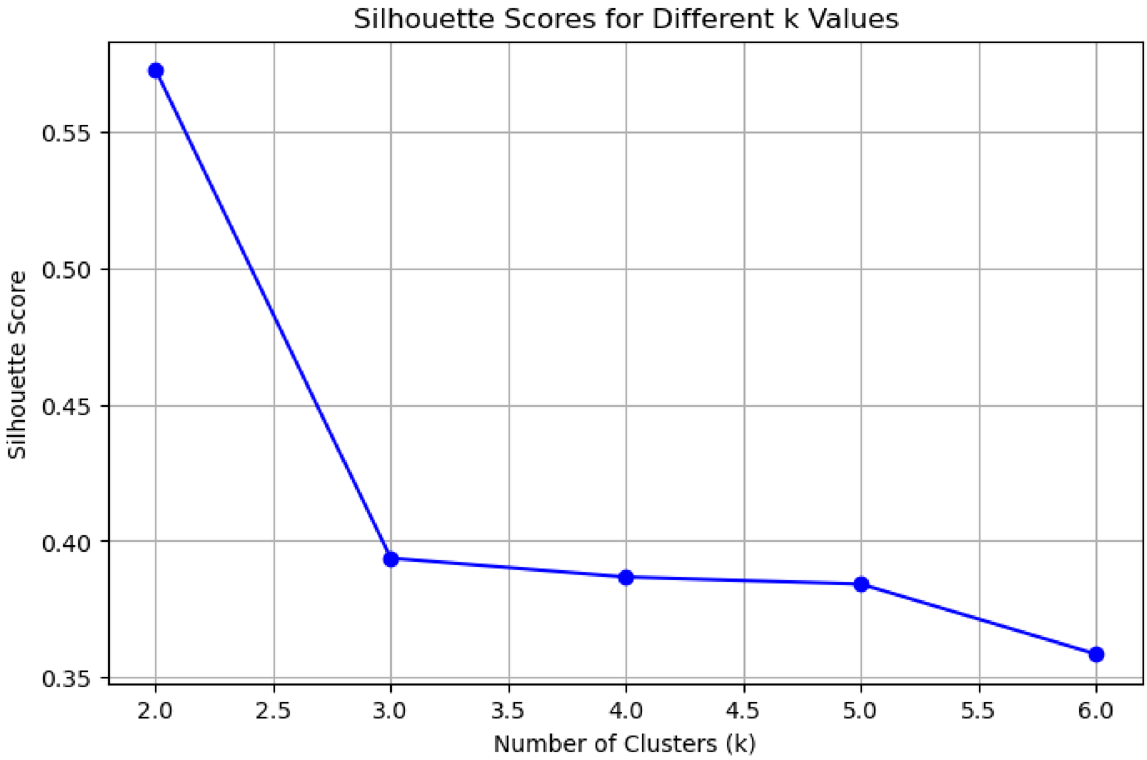
Hyperparameters available in the Orange tool:

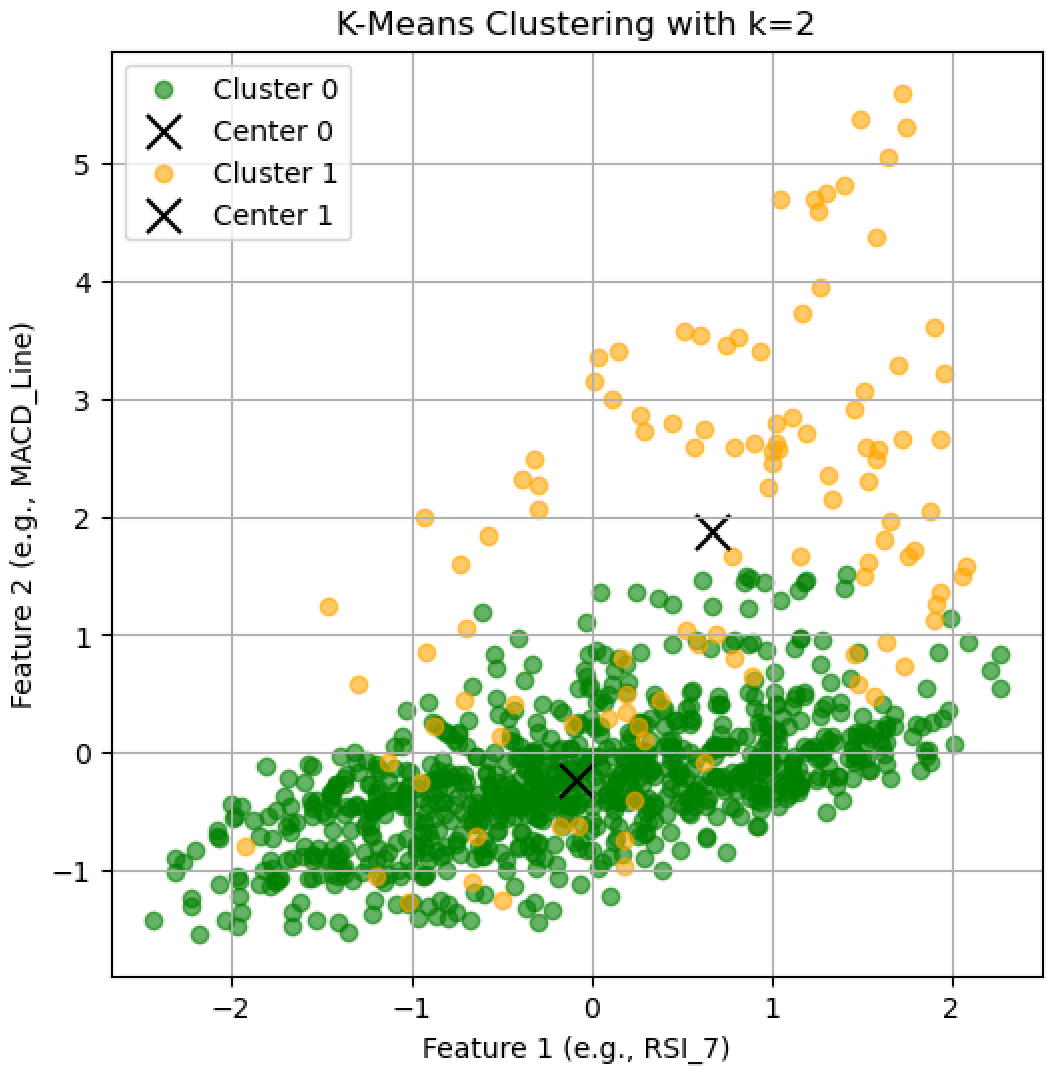
<add rows to table as needed>

Hyperparameter	Description
K for n_clusters	Different k will affect the number of clusters in the clustering process

```
# Standardize the data
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data_filtered[['RSI_7', 'MACD_Line', 'Close', 'High', 'Low']])

# Define the range of k values to try
k_values = [2, 3, 4, 5, 6]

# Iterate through each k value
for k in k_values:
    # Create and fit the KMeans model
    kmeans = KMeans(n_clusters=k, random_state=42)
    clusters = kmeans.fit_predict(data_scaled)
    cluster_centers = kmeans.cluster_centers_
```

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

```python
# Split the data into training and testing based on time sequence
train_size = int(len(data_filtered) * 0.75)

# Training data (first 75%)
train_data = data_filtered[:train_size]

# Testing data (last 25%)
test_data = data_filtered[train_size:]

# Print the size of train and test data
print(f"Training data size: {train_data.shape}")
print(f"Testing data size: {test_data.shape}")
```

✓ 0.0s Python

```
         Date      Open      High       Low     Close  Adj Close     Volume  \
0  2020-08-05  11.24400  11.37175  11.16625  11.28675  11.253224  249924000
1  2020-08-06  11.34975  11.35800  11.17875  11.33550  11.301830  244316000
2  2020-08-07  11.31250  11.50475  11.03750  11.19950  11.166232  342516000
3  2020-08-10  11.33425  11.40825  10.85650  11.16500  11.131833  427796000
4  2020-08-11  11.07375  11.13675  10.79575  10.85000  10.817771  354512000

       RSI_7  MACD_Line  MACD_Signal  MACD_Hist  Is_Higher_Than_Previous_Close
0  79.314298   0.345669     0.271446   0.074223                              1
1  80.212551   0.371684     0.291493   0.080191                              1
2  70.279787   0.376982     0.308591   0.068391                              0
3  67.795203   0.374085     0.321690   0.052395                              0
4  49.248877   0.342423     0.325836   0.016587                              0
Training data size: (730, 12)
Testing data size: (244, 12)
```

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

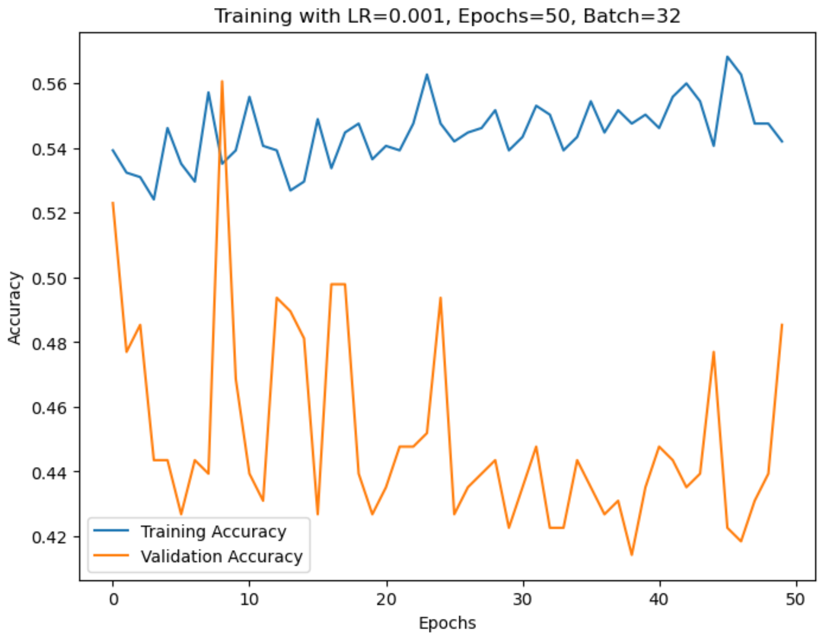
Here, I used the data from my second LSTM model with the following structure:

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_18 (LSTM)              (None, 5, 64)             18176

 dropout_18 (Dropout)        (None, 5, 64)             0

 lstm_19 (LSTM)              (None, 5, 128)            98816

 dropout_19 (Dropout)        (None, 5, 128)            0

 lstm_20 (LSTM)              (None, 64)                49408

 dropout_20 (Dropout)        (None, 64)                0

 dense_18 (Dense)            (None, 64)                4160

 dense_19 (Dense)            (None, 1)                 65

=================================================================
Total params: 170,625
Trainable params: 170,625
Non-trainable params: 0
```

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

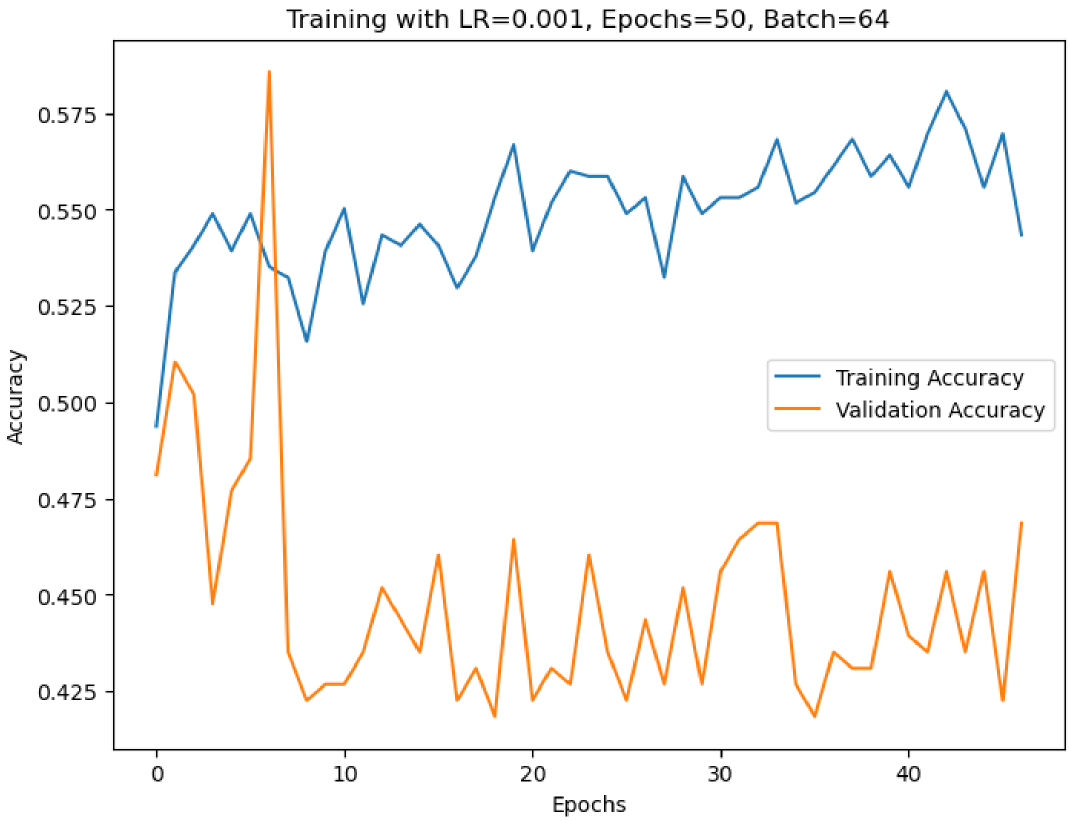
Experiment	Hyperparameter values
Experiment 1	LR = 0.001, Epochs = 50, Batch = 32
Experiment 2	LR = 0.001, Epochs = 50, Batch = 64
Experiment 3	LR = 0.002, Epochs = 50, Batch = 32

Experiment 1:



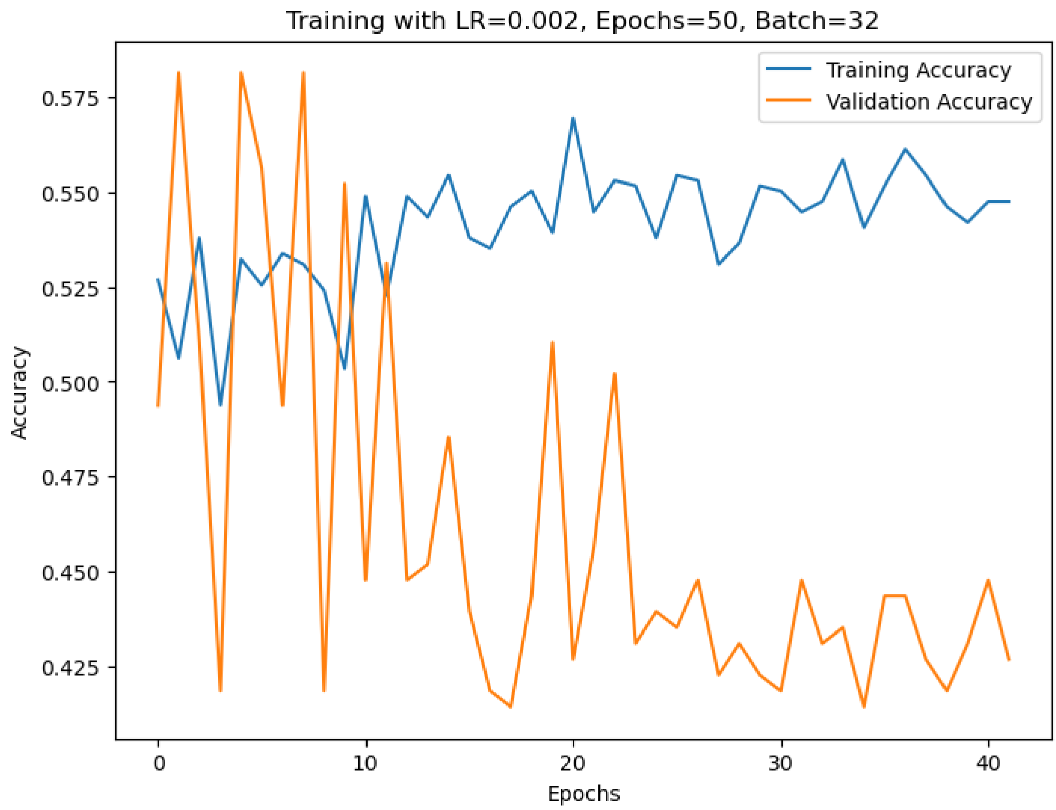
```
Accuracy (Experiment): 0.49
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.42      1.00      0.59       100
           1       0.00      0.00      0.00       139

    accuracy                           0.42       239
   macro avg       0.21      0.50      0.29       239
weighted avg       0.18      0.42      0.25       239

ROC-AUC Score: 0.51
```

Experiment 2:




```
Accuracy (Experiment): 0.47
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.42      1.00      0.59       100
           1       0.00      0.00      0.00       139

    accuracy                           0.42       239
   macro avg       0.21      0.50      0.29       239
weighted avg       0.18      0.42      0.25       239

ROC-AUC Score: 0.51
```

Experiment 3:




```
Accuracy (Experiment): 0.43
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.42      1.00      0.59       100
           1       0.00      0.00      0.00       139

    accuracy                           0.42       239
   macro avg       0.21      0.50      0.29       239
weighted avg       0.18      0.42      0.25       239

ROC-AUC Score: 0.51
```

<u>Conclusions from experiments:</u>

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

<u>Model selected for testing:</u>

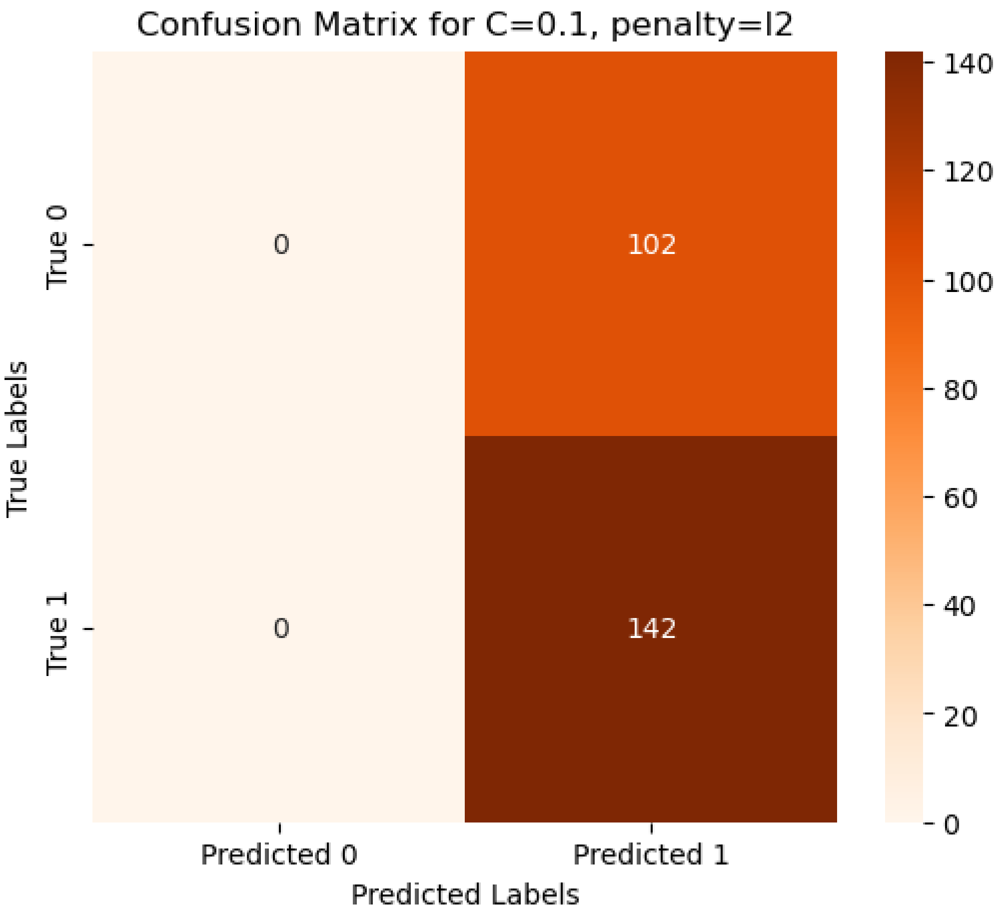
Experiment 1

Experiments with Logistic Regression

<add rows to table as needed>

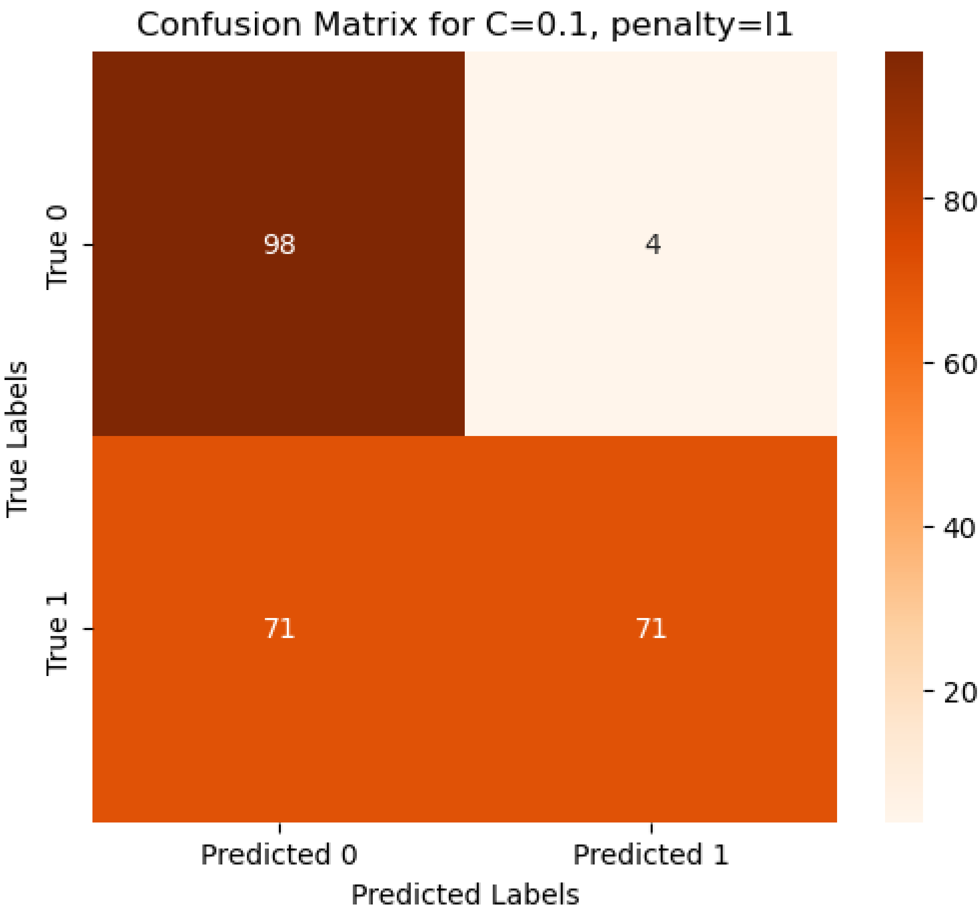
Experiment	Hyperparameter values
Experiment 1	C=0.1, penalty=l2
Experiment 2	C=0.1, penalty=l1
Experiment 3	C=10, penalty=l2
Experiment 4	C=10, penalty=l1

Experiment 1:



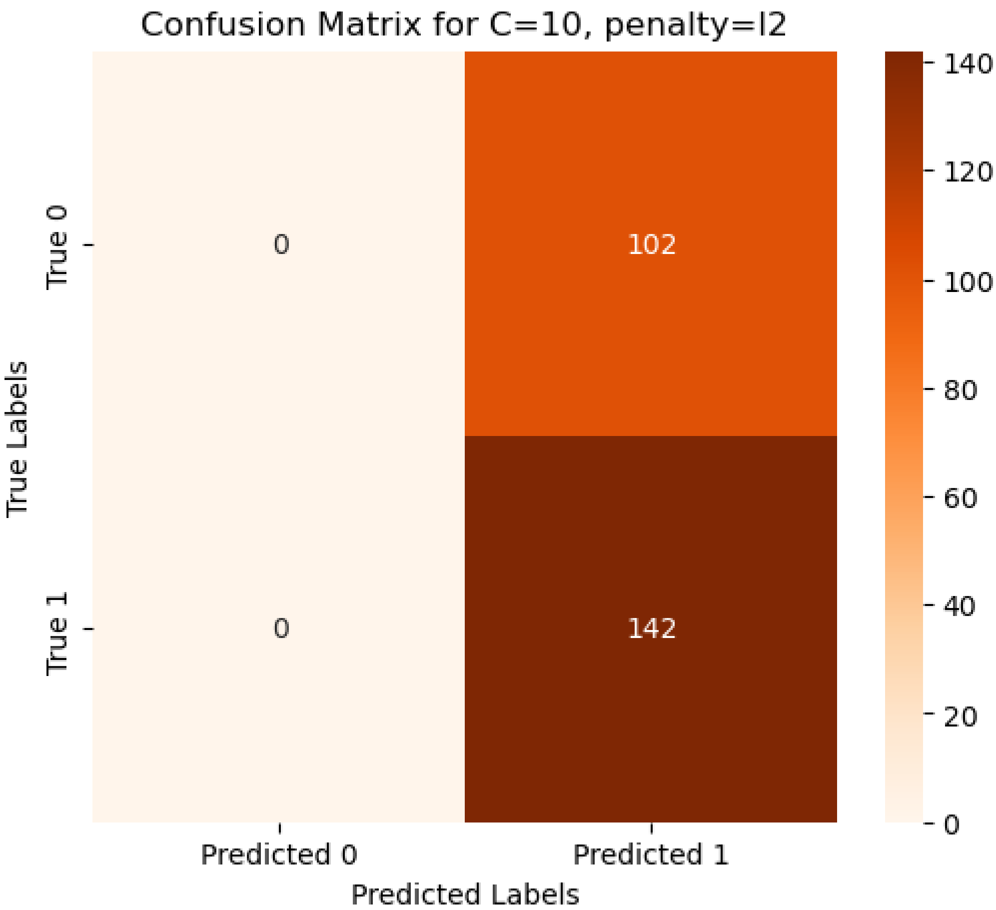

```
Training with C=0.1, penalty=l2
Accuracy: 0.5819672131147541
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       102
           1       0.58      1.00      0.74       142

    accuracy                           0.58       244
   macro avg       0.29      0.50      0.37       244
weighted avg       0.34      0.58      0.43       244

ROC-AUC Score: 0.45
Confusion Matrix (Experiment):
[[  0 102]
 [  0 142]]
```

Experiment 2:



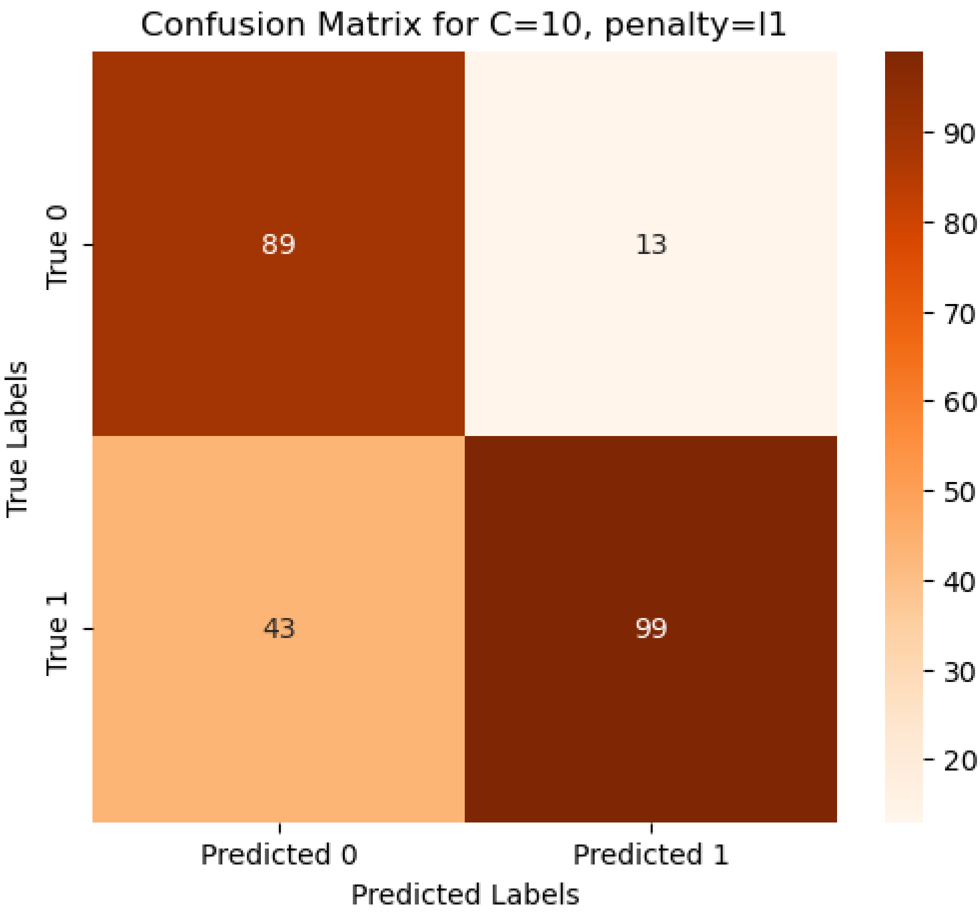

```
Training with C=0.1, penalty=l1
Accuracy: 0.6926229508196722
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.58      0.96      0.72       102
           1       0.95      0.50      0.65       142

    accuracy                           0.69       244
   macro avg       0.76      0.73      0.69       244
weighted avg       0.79      0.69      0.68       244

ROC-AUC Score: 0.88
Confusion Matrix (Experiment):
[[98  4]
 [71 71]]
```

Experiment 3:




```
Training with C=10, penalty=l2
Accuracy: 0.5819672131147541
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       102
           1       0.58      1.00      0.74       142

    accuracy                           0.58       244
   macro avg       0.29      0.50      0.37       244
weighted avg       0.34      0.58      0.43       244

ROC-AUC Score: 0.45
Confusion Matrix (Experiment):
[[  0 102]
 [  0 142]]
```

Experiment 4:




```
Accuracy: 0.7704918032786885
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.67      0.87      0.76       102
           1       0.88      0.70      0.78       142

    accuracy                           0.77       244
   macro avg       0.78      0.78      0.77       244
weighted avg       0.80      0.77      0.77       244

ROC-AUC Score: 0.85
Confusion Matrix (Experiment):
[[89 13]
 [43 99]]
```

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:

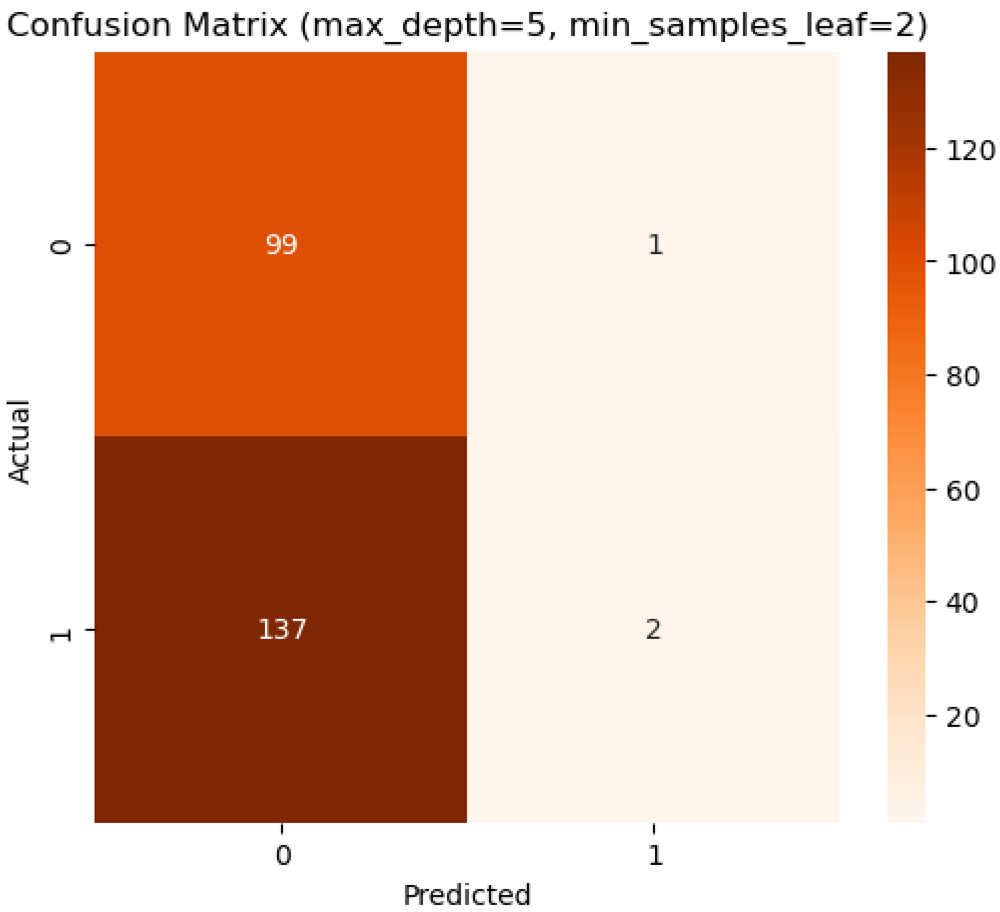
Experiment 4

Experiments with Random Forest

<add rows to table as needed>

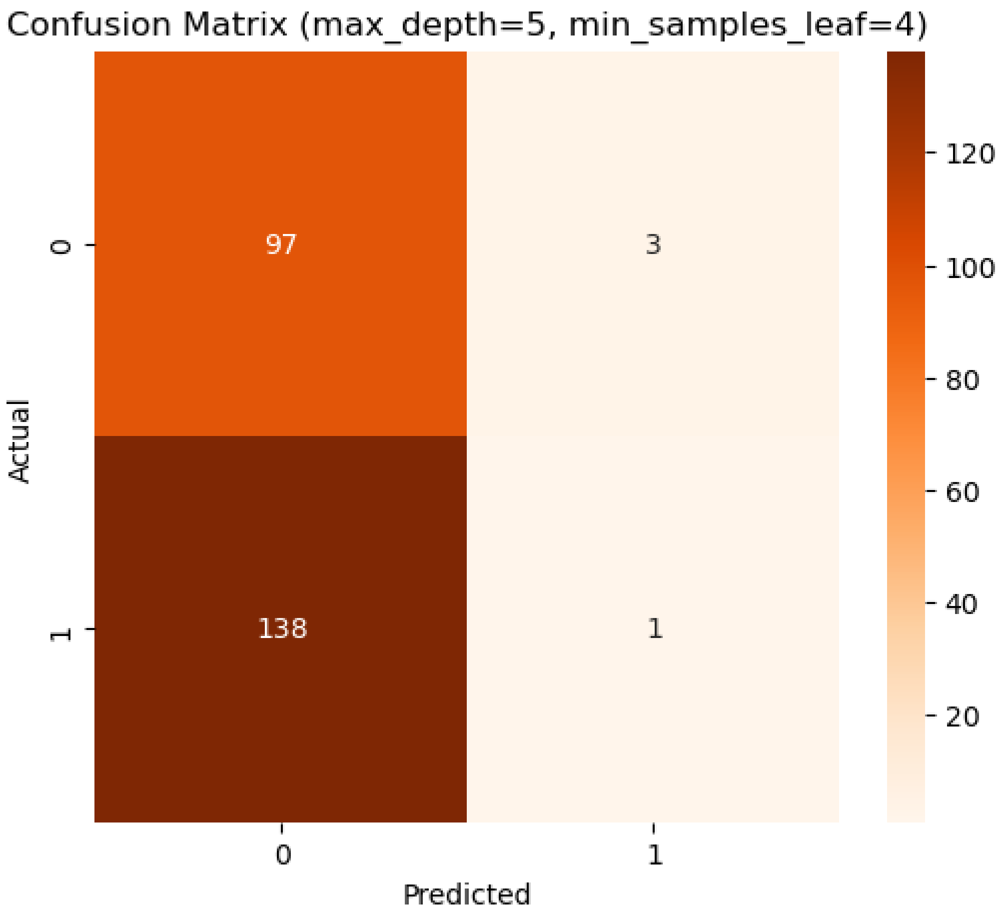
Experiment	Hyperparameter values
Experiment 1	max_depth=5, min_samples_leaf=2
Experiment 2	max_depth=5, min_samples_leaf=4
Experiment 3	max_depth=10, min_samples_leaf=2
Experiment 4	max_depth=10, min_samples_leaf=4

Experiment 1:



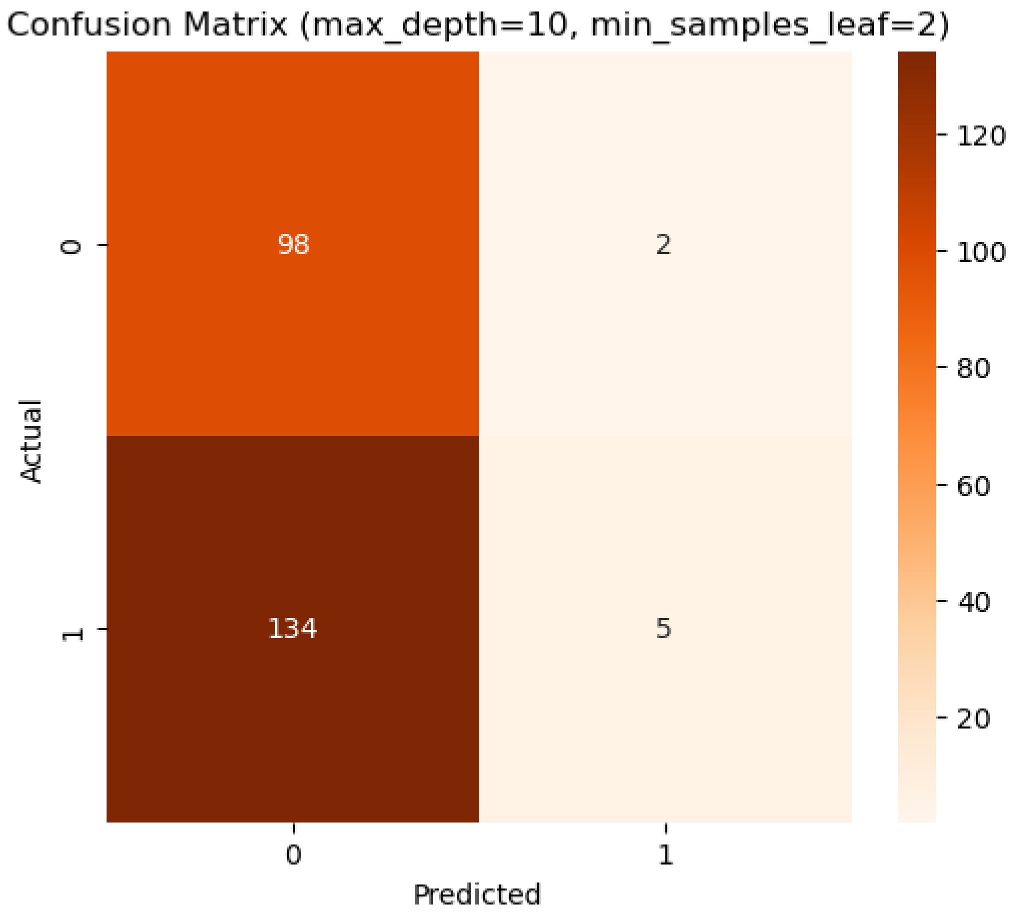

```
Training with max_depth=5, min_samples_leaf=2
Accuracy: 0.4225941422594142
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.42      0.99      0.59       100
           1       0.67      0.01      0.03       139

    accuracy                           0.42       239
   macro avg       0.54      0.50      0.31       239
weighted avg       0.56      0.42      0.26       239

ROC-AUC Score: 0.50
Model saved to RF_Model/model_max_depth5_min_samples_leaf2.pkl
```

Experiment 2:



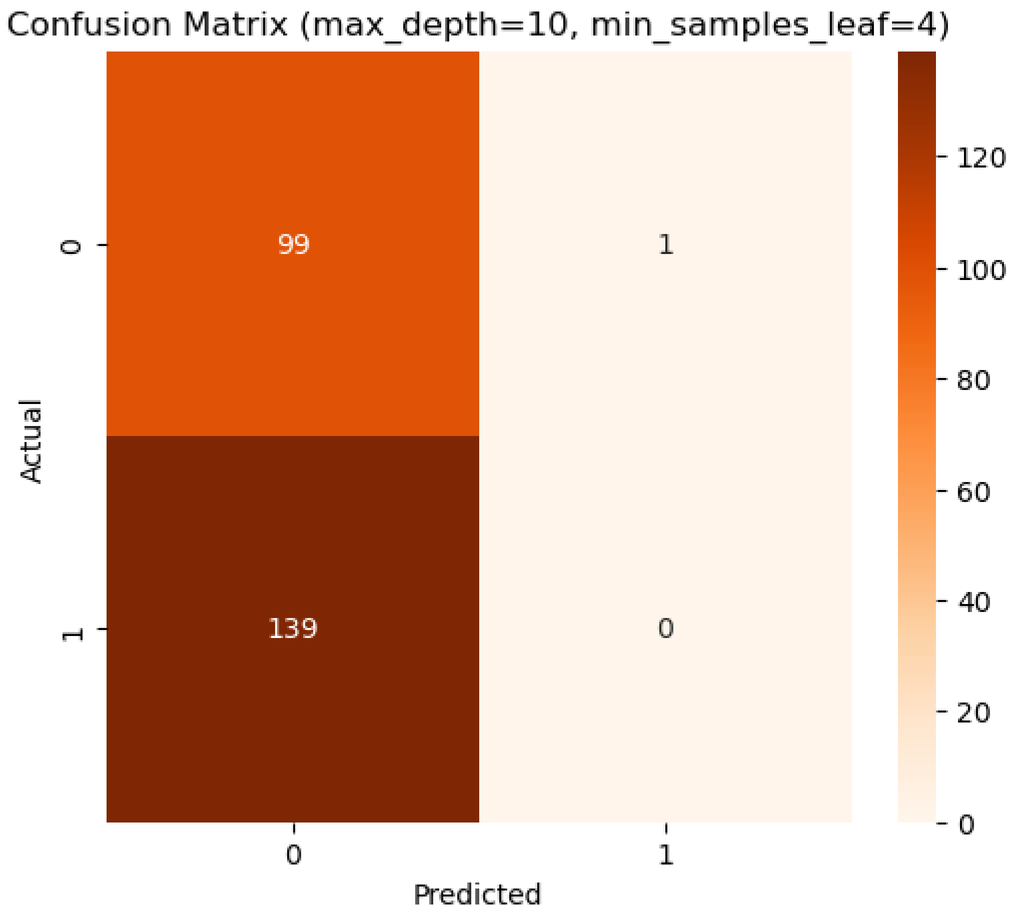

```
Training with max_depth=5, min_samples_leaf=4
Accuracy: 0.4100418410041841
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.41      0.97      0.58       100
           1       0.25      0.01      0.01       139

    accuracy                           0.41       239
   macro avg       0.33      0.49      0.30       239
weighted avg       0.32      0.41      0.25       239

ROC-AUC Score: 0.51
Model saved to RF_Model/model_max_depth5_min_samples_leaf4.pkl
```

Experiment 3:




```
Training with max_depth=10, min_samples_leaf=2
Accuracy: 0.4309623430962343
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.42      0.98      0.59       100
           1       0.71      0.04      0.07       139

    accuracy                           0.43       239
   macro avg       0.57      0.51      0.33       239
weighted avg       0.59      0.43      0.29       239

ROC-AUC Score: 0.51
Model saved to RF_Model/model_max_depth10_min_samples_leaf2.pkl
```

Experiment 4




```
Training with max_depth=10, min_samples_leaf=4
Accuracy: 0.41422594142259417
Classification Report (Experiment):
              precision    recall  f1-score   support

           0       0.42      0.99      0.59       100
           1       0.00      0.00      0.00       139

    accuracy                           0.41       239
   macro avg       0.21      0.49      0.29       239
weighted avg       0.17      0.41      0.25       239

ROC-AUC Score: 0.49
Model saved to RF_Model/model_max_depth10_min_samples_leaf4.pkl
```

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)

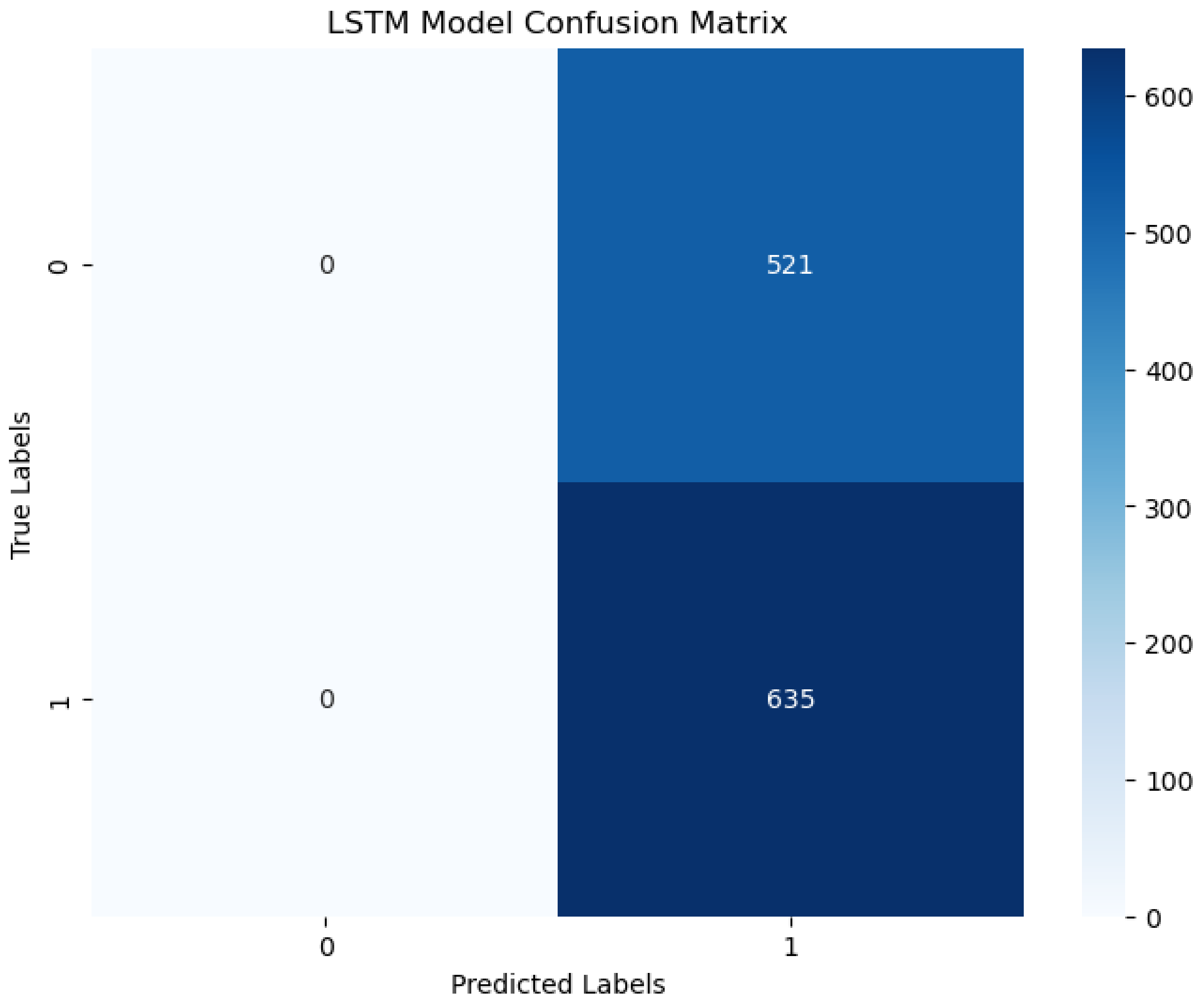
   Perhaps the data is slightly imbalanced (as we can see from data summary) and this has affected the Random Forest.

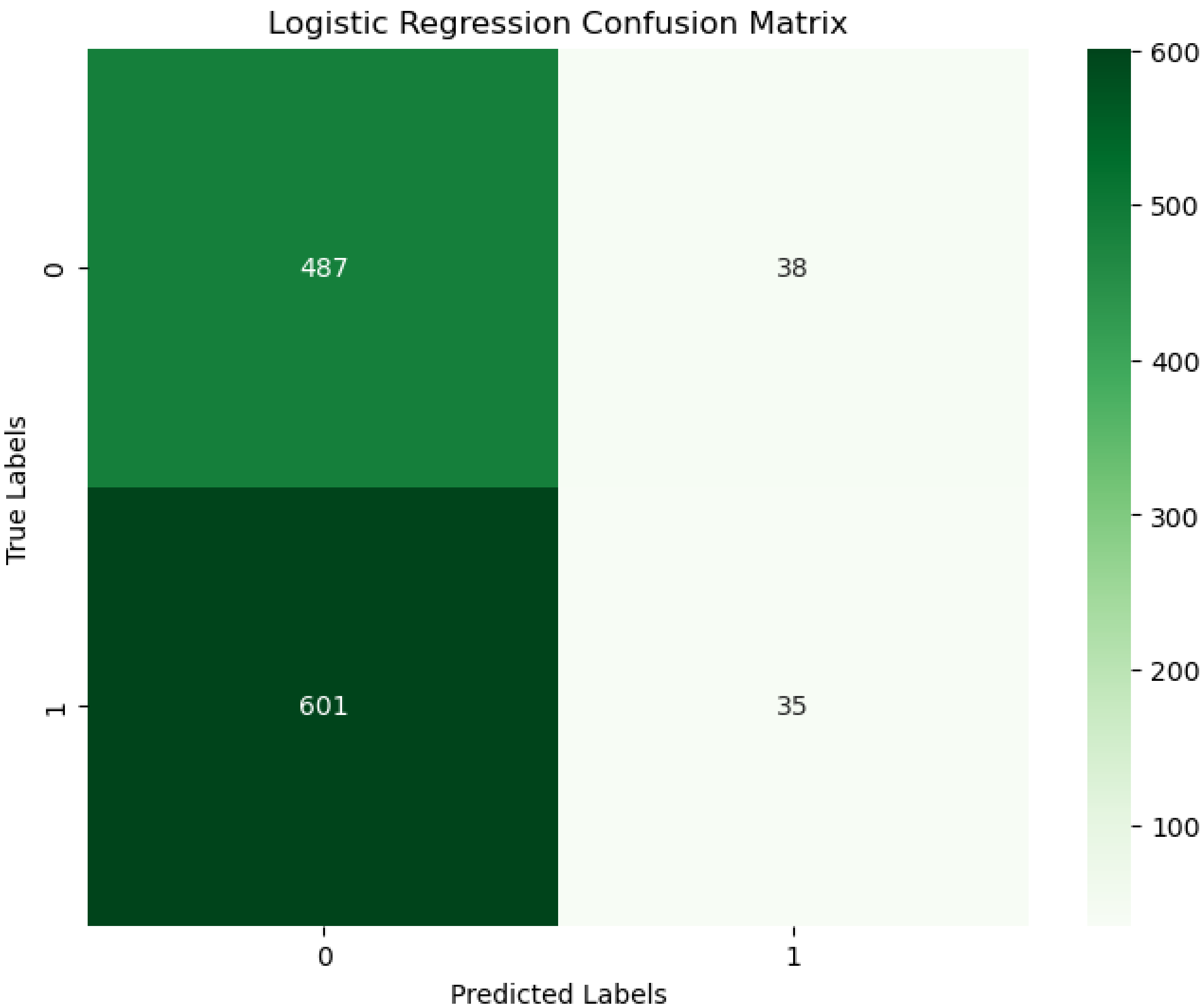
Model selected for testing:

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

```
37/37 [==============================] - 1s 3ms/step
=== LSTM Model ===
Accuracy: 0.5493079584775087
Classification Report:
               precision    recall  f1-score   support

           0       0.00      0.00      0.00       521
           1       0.55      1.00      0.71       635

    accuracy                           0.55      1156
   macro avg       0.27      0.50      0.35      1156
weighted avg       0.30      0.55      0.39      1156

ROC-AUC Score: 0.5223570662112533
Confusion Matrix:
 [[  0 521]
 [  0 635]]

=== Logistic Regression Model ===
Accuracy: 0.4496124031007752
Classification Report:
               precision    recall  f1-score   support

           0       0.45      0.93      0.60       525
           1       0.48      0.06      0.10       636
...
ROC-AUC Score: 0.45777777777777784
Confusion Matrix:
 [[487  38]
 [601  35]]
```





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.