Can you predict the Stock Market reliably?

Machine Learning Final
Professor: Li
Salvatore Cirisano

Task and Purpose

Task : Based on previous market data, can we be confident in whether or not the current day will be a positive or negative change in stock price?

Purpose : The majority of stock market analytics is done by Hedge Funds who trade on a daily basis, as well as long term investing. Their goal is to make money.

The Data

```
import yfinance as yf
```

Attributes

Close	High	Low	Open	Volume
0.872768	0.912946	0.845982	0.910714	505064000

Close

Numerical

Where the price closed the day at

Open

Numerical

Where the price opened the day at

Volume

Numerical

The total volume of transactions for that day

Hypothesis: If previous days had extremely high/low volume or high/low change in price, The following days will react similarly

Preprocessing

Change	pos_neg	vol_change
0.037946	1	-1.0
0.028460	1	-1.0
0.069754	1	-1.0
-0.020229	-1	-1.0
-0.003906	-1	-1.0
-0.026227	-1	-1.0
-0.008371	-1	-1.0

Our Target (Y) value is pos_neg because it determines if the day was + or -.

However if we have all the data for the day, we would know the change with 100% certainty.

You need to take the previous days data and shift it to the day after because that is what's determining its outcome

NEW ATTRIBUTES:

Change = Open - Close

Pos_neg = 1 else -1 if Change >0

Vol_change 1 else -1 if Vol > Mean

```
for data in stocks:
  open = data['Open']
  close = data['Close']
  data['Change'] = data['Open'] - data['Close']
  print(data['Change'])
  data['pos_neg'] = 0
  data.loc[data['Change'] > 0, 'pos_neg'] = 1
  data.loc[data['Change'] < 0, 'pos_neg'] = -1
  data.loc[data['Change'] == 0, 'pos_neg'] = 0
```

```
for data in stocks:
  data['Volume'].fillna(0)
  mean = data['Volume'].mean()
  data['vol_change']= mean-data['Volume']
  data.loc[data['vol_change'] > 0, 'vol_change'] = 1
  data.loc[data['vol_change'] < 0, 'vol_change'] = -1
  data['vol_change'] = data['vol_change'].shift(1)
```

Final Attributes

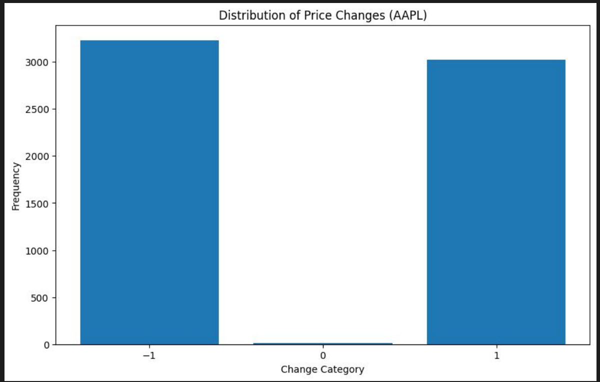
Change	pos_neg	day_before	vol_change	day_before1	day_before2	day_before3	day_before4
0.037946	1	-0.026786	-1.0	0.099331	-0.002232	0.051339	-0.063058
0.028460	1	0.037946	-1.0	-0.026786	0.099331	-0.002232	0.051339
0.069754	1	0.028460	-1.0	0.037946	-0.026786	0.099331	-0.002232
-0.020229	-1	0.069754	-1.0	0.028460	0.037946	-0.026786	0.099331
-0.003906	-1	-0.020229	-1.0	0.069754	0.028460	0.037946	-0.026786
-0.026227	-1	-0.003906	-1.0	-0.020229	0.069754	0.028460	0.037946
-0.008371	-1	-0.026227	-1.0	-0.003906	-0.020229	0.069754	0.028460
0.017857	1	-0.008371	-1.0	-0.026227	-0.003906	-0.020229	0.069754
0.026227	1	0.017857	-1.0	-0.008371	-0.026227	-0.003906	-0.020229

NEW ATTRIBUTES:

Day_before(1,2,3,4) = previous day shift(1)

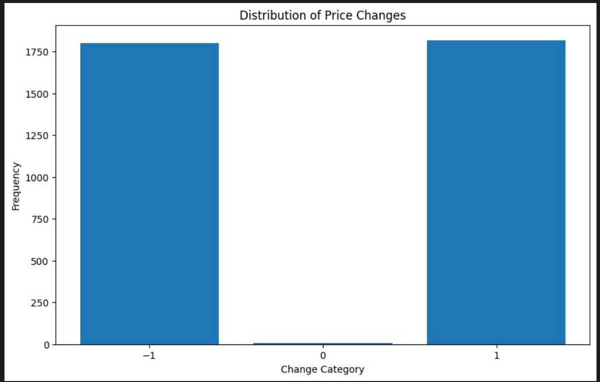
```
for data in stocks:
  data['day_before1']= data['day_before'].shift(1)
  data.drop(index=data.index[0], inplace=True)

for data in stocks:
  data['day_before2']= data['day_before1'].shift(1)
  data.drop(index=data.index[0], inplace=True)

for data in stocks:
  data['day_before3']= data['day_before2'].shift(1)
  data.drop(index=data.index[0], inplace=True)

for data in stocks:
  data['day_before4']= data['day_before3'].shift(1)
  data.drop(index=data.index[0], inplace=True)
```

Data Visualization



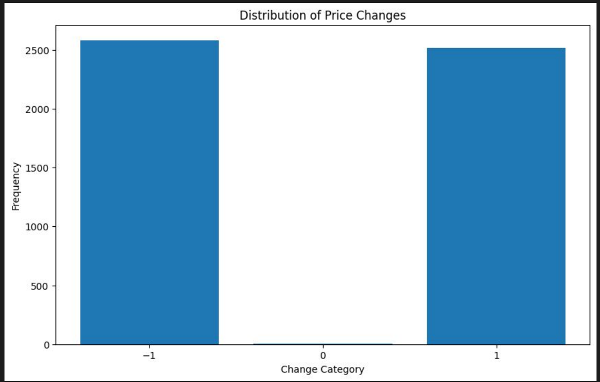
APPLE

Apple had more - days than + but not by much



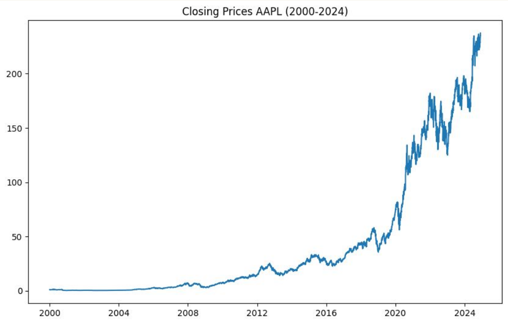
Tesla

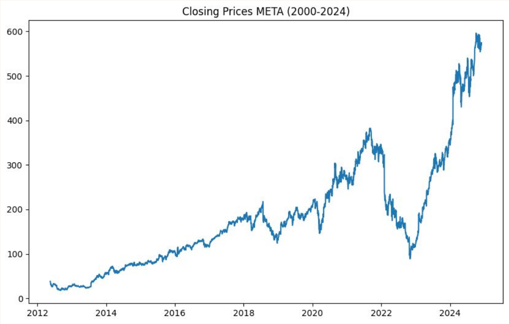
Tesla had more + days than - but also not by much



GOOGLE

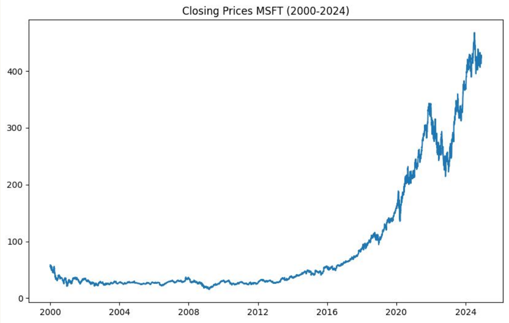
Google and the other 4 stocks all had more - days than + but again not by many

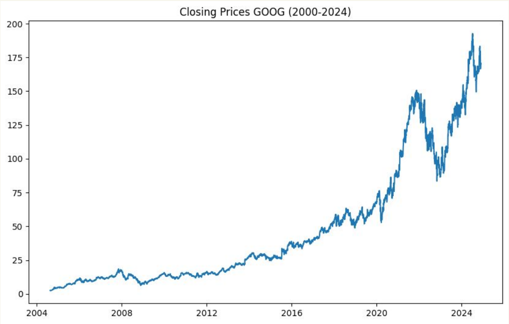

Closing Prices AAPL (2000-2024)
0
50
100
150
200
2000
2004
2008
2012
2016
2020
2024

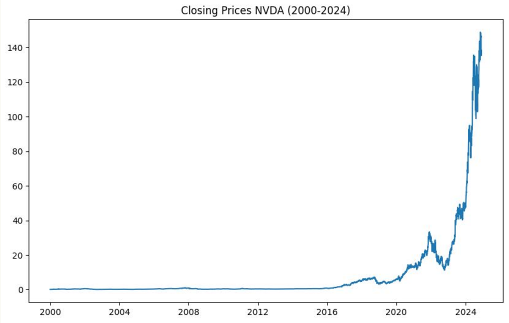

Closing Prices META (2000-2024)
0
100
200
300
400
500
600
2012
2014
2016
2018
2020
2022
2024

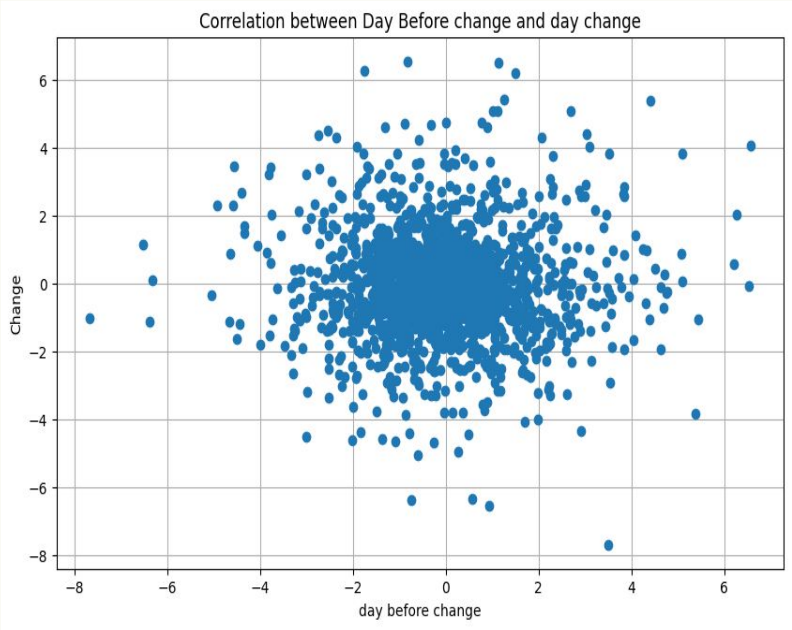

Closing Prices TSLA (2000-2024)
0
50
100
150
200
250
300
350
400
2010
2012
2014
2016
2018
2020
2022
2024

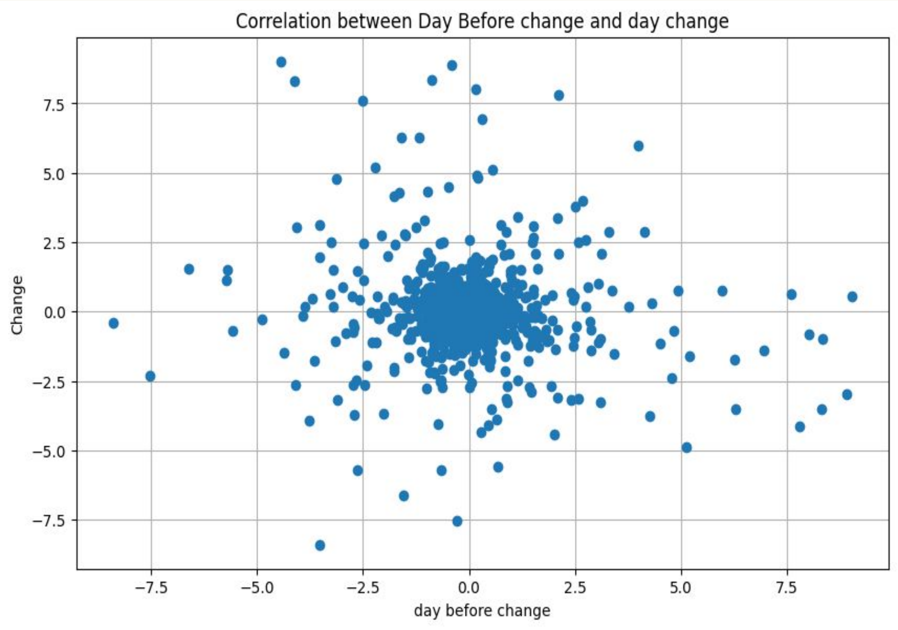

Closing Prices AMZN (2000-2024)
0
50
100
150
200
2000
2004
2008
2012
2016
2020
2024


Closing Prices MSFT (2000-2024)
0
100
200
300
400
2000
2004
2008
2012
2016
2020
2024


Closing Prices GOOG (2000-2024)
0
25
50
75
100
125
150
175
200
2004
2008
2012
2016
2020
2024


Closing Prices NVDA (2000-2024)
0
20
40
60
80
100
120
140
2000
2004
2008
2012
2016
2020
2024





Opposite Correlation?

Methods

Decision Tree Classifier

Using a GINI tree

Evaluation Metric: Accuracy Score

Added a double for loop to check various test sizes and depths to find the highest accuracy scores

```
for data in stocks:
  max = 0
  depth=0
  size = 0

  for i in range(1,20):

    clf = DecisionTreeClassifier(max_depth = i, criterion = 'entropy')

    X = data[['day_before','day_before1','day_before2', 'day_before3', 'day_before4', 'vol_change', 'Volume']]
    y = data['pos_neg']
    for j in range(1,20,1):

      x_train, x_test, y_train, y_test = train_test_split(X, y, test_size= j/20, random_state=42)
      clf.fit(x_train, y_train)
      y_pred = clf.predict(x_test)
      accuracy = accuracy_score(y_test, y_pred)
      if accuracy > max:
        max = accuracy
        depth = i
        size = j

  clf = DecisionTreeClassifier(max_depth = depth, criterion = 'gini')
  X = data[['day_before','day_before1','day_before2', 'day_before3', 'day_before4','vol_change', 'Volume']]
  y = data['pos_neg']
  x_train, x_test, y_train, y_test = train_test_split(X, y, test_size= size/20, random_state=42)
  clf.fit(x_train, y_train)
  print(max)
  plt.figure(figsize=(20,10))
  plot_tree(clf, feature_names=['day_before','day_before1','day_before2', 'day_before3', 'day_before4',
                                'Volume_change', 'volume'], class_names=['-1', '0', '1'], filled=True, rounded=True)
  plt.show()
```

Methods

KNN Model

KNN model using 1000 Neighbors ≅ 20% of data size

Evaluation Metric: Accuracy Score

```
k= 0
for data in stocks:
  X = data[['day_before','day_before1','day_before2', 'day_before3', 'day_before4', 'Volume']]
  y = data['pos_neg']

  x_train, x_test, y_train, y_test = train_test_split(X, y, test_size= 0.20, random_state=42)

  knn_model = KNeighborsClassifier(n_neighbors=1000)


  knn_model.fit(x_train, y_train)
  y_pred = knn_model.predict(x_test)
  accuracy = accuracy_score(y_test, y_pred)
  print(f"Accuracy for KNN model ", Mag7[k], " ", accuracy)
  k=k+1
  y_pred_proba = knn_model.predict_proba(x_test)[:,1]
```

Results

KNN

```
Accuracy for KNN model  AAPL   0.5307262569832403
Accuracy for KNN model  MSFT   0.5362745098039216
Accuracy for KNN model  TSLA   0.5426975259377494
Accuracy for KNN model  GOOG   0.5227586206896552
Accuracy for KNN model  AMZN   0.4932162809257781
Accuracy for KNN model  META   0.4860335195530726
Accuracy for KNN model  NVDA   0.48412698412698413
```

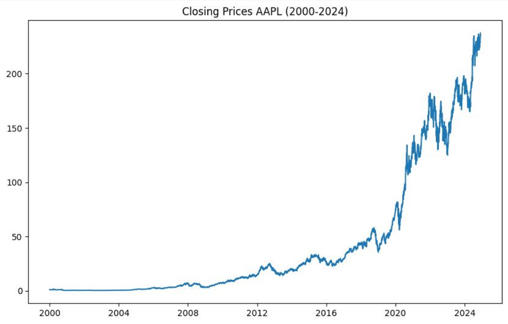
Decision Tree

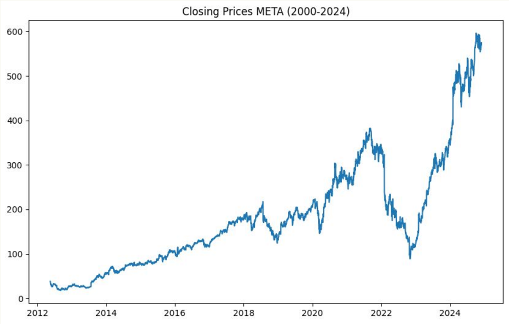
```
0.5521276595744681
0.5568627450980392
0.5458898643256185
0.53125
0.5573248407643312
0.5732484076433121
0.5506329113924051
```

For every stock, the decision tree outperformed the KNN model in accuracy score.

But why are the values so low?

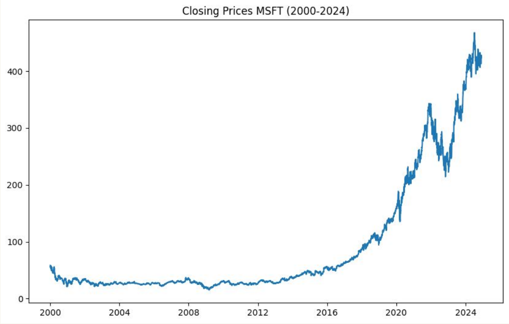
Does this disprove the hypothesis?


Closing Prices AAPL (2000-2024)
0 50 100 150 200
2000 2004 2008 2012 2016 2020 2024

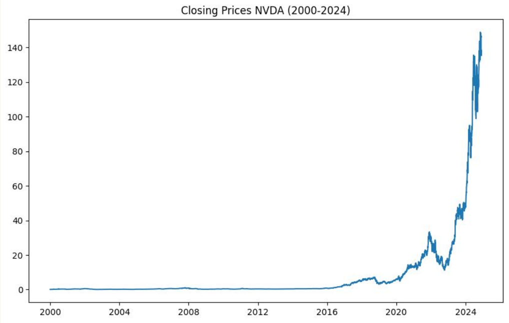

Closing Prices META (2000-2024)
0 100 200 300 400 500 600
2012 2014 2016 2018 2020 2022 2024


Closing Prices TSLA (2000-2024)
0 50 100 150 200 250 300 350 400
2010 2012 2014 2016 2018 2020 2022 2024


Closing Prices AMZN (2000-2024)
0 50 100 150 200
2000 2004 2008 2012 2016 2020 2024


Closing Prices MSFT (2000-2024)
0 100 200 300 400
2000 2004 2008 2012 2016 2020 2024


Closing Prices GOOG (2000-2024)
0 25 50 75 100 125 150 175 200
2004 2008 2012 2016 2020 2024


Closing Prices NVDA (2000-2024)
0 20 40 60 80 100 120 140
2000 2004 2008 2012 2016 2020 2024

Interpretation


```
positive max:  50.033355712890625
negative min:   -33.46000671386719
```

```
negatives median -0.1758403778076172
positives median 0.15999984741210938
```

```
positive mean: 0.7980327411367214
negative mean: -0.8018823389204086
```

Min/Max

The max and min differ more extremely. This would account for the increases in stock price over the years despite equal amounts of + and - days

Median

The Median is relatively the same on both ends meaning it wouldn't make a large difference

Mean

The Means are extremely similar (one + one -) which would make a positive or negative day be the same gain or loss

Risk or Reward?

Based on our decision Tree model, we can predict the outcome for the 7 stocks daily between 53%-57% accuracy.

Break even rate

To break even at a 50% win rate you need to win as much as you are losing

EX) Flip a coin 100 times, win 50/100. As long as you put in x and receive 2 x then you will break even.

½ win rate = 2/1 return rate.

Decision Tree Rate

Worst Case:
53/100 win rate
= 100/53 return rate
= 1.886x return rate

```
negatives median -0.1758403778076172
positives median 0.15999984741210938
```

.16/ .1758 = .9101

Or 1.901X return rate

2.41 %-15.6% advantage
.9101-.886 = .0241
.9101- .754 = .156

Best Case:
57/100 win rate
= 100/57 return rate
= 1.754x return rate

```
positive mean: 0.7980327411367214
negative mean: -0.8018823389204086
```

.798/ .801 = .996

Or 1.996X return rate

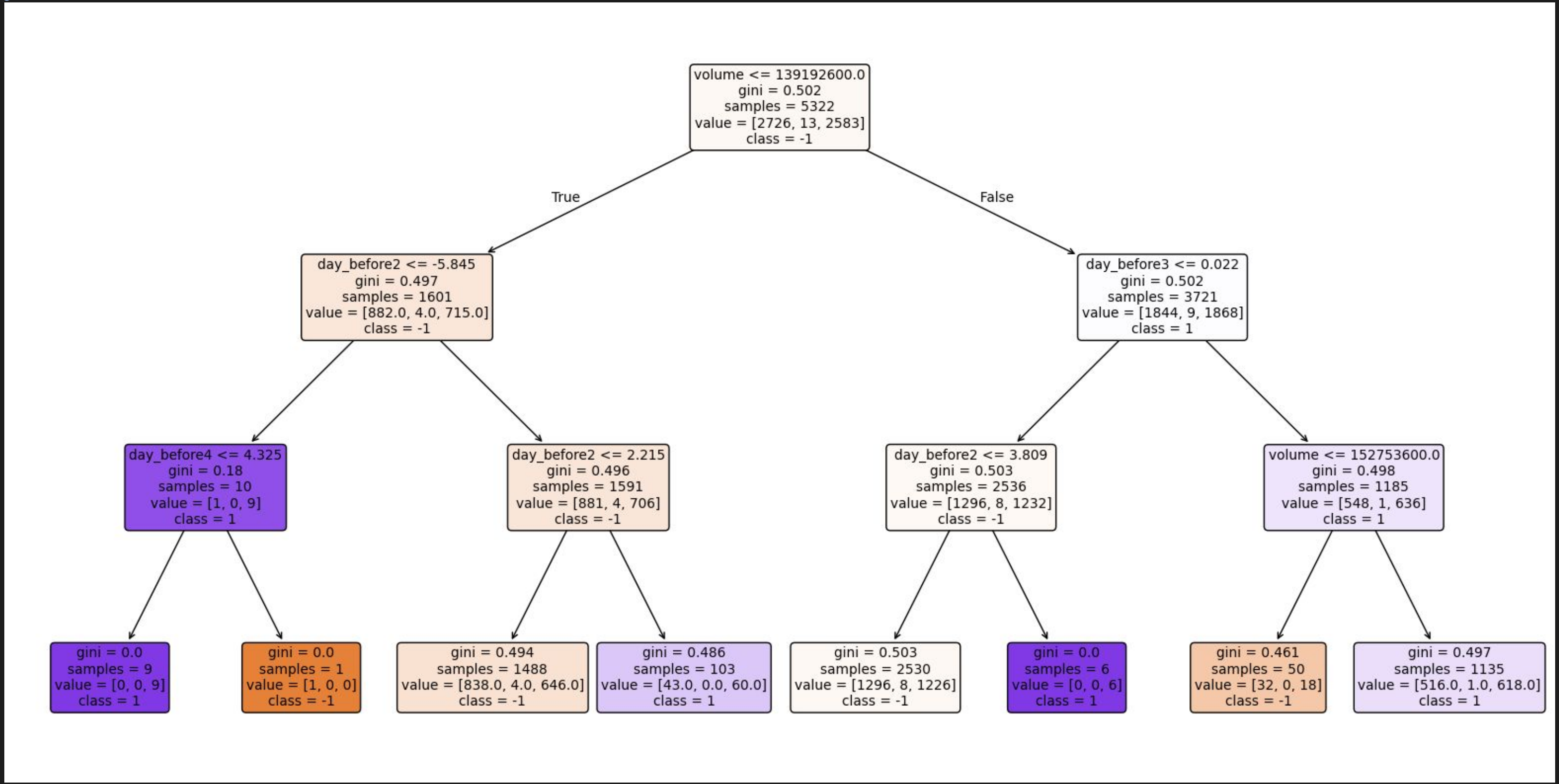
10%- 22.4% advantage
.996- .886= .10 .
996- .754= .224

Outliers

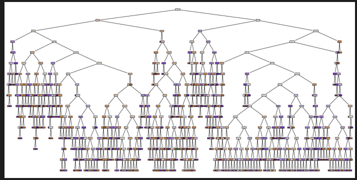
The outliers only work in our favor, proving the validity of the Decision Tree model. On a pure average and median the system is as profitable as it can get, and on the highest moving days in either direction, the positive direction takes the win

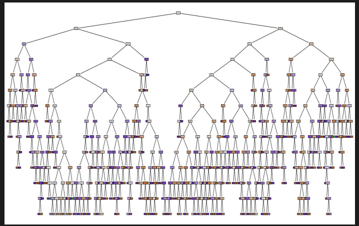
```
Third Quartile of Positives: 16.48087219238285
Third Quartile of Negatives: -14.399478912353516
```

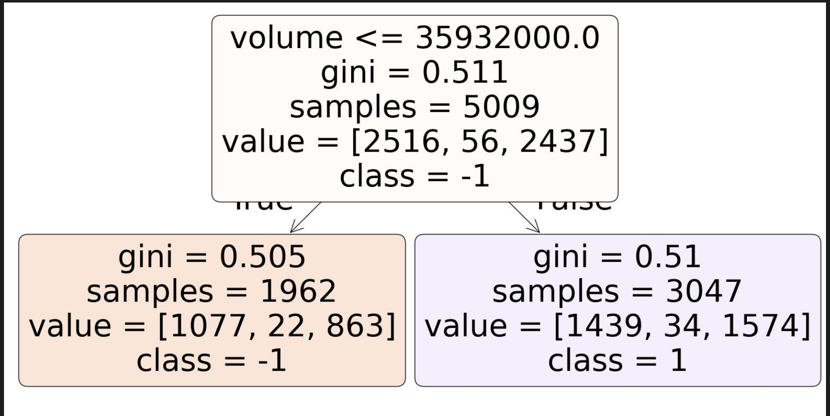
```
positive max: 50.033355712890625
negative min:  -33.46000671386719
```

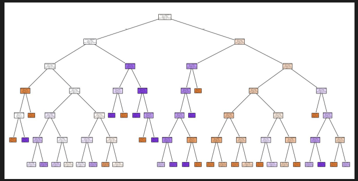


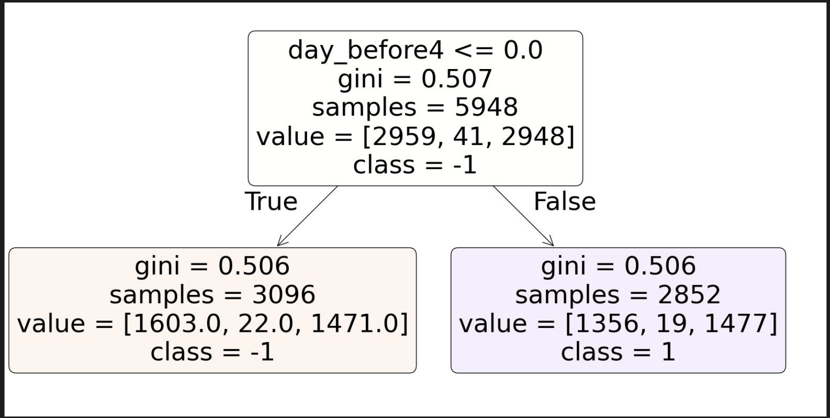

```
0.5521276595744681 APPLE
```

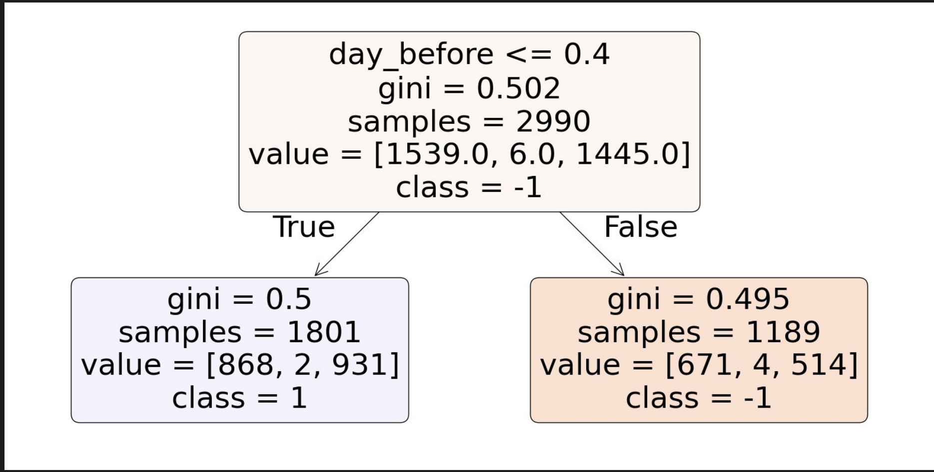





volume <= 35932000.0
gini = 0.511
samples = 5009
value = [2516, 56, 2437]
class = -1
gini = 0.505
samples = 1962
value = [1077, 22, 863]
class = -1
gini = 0.51
samples = 3047
value = [1439, 34, 1574]
class = 1




day_before4 <= 0.0
gini = 0.507
samples = 5948
value = [2959, 41, 2948]
class = -1
True
False
gini = 0.506
samples = 3096
value = [1603.0, 22.0, 1471.0]
class = -1
gini = 0.506
samples = 2852
value = [1356, 19, 1477]
class = 1


day_before <= 0.4
gini = 0.502
samples = 2990
value = [1539.0, 6.0, 1445.0]
class = -1
True
False
gini = 0.5
samples = 1801
value = [868, 2, 931]
class = 1
gini = 0.495
samples = 1189
value = [671, 4, 514]
class = -1

Conclusion

Based on previous days volume and price changes, the Stock market can be predicted with some element of certainty.

Using a decision tree model, the worst possible advantage one may have is a 2.41% and at best 22.4% not including outliers, which have positive effects in all cases

Although 2.4 % may seem small, many banks use these types of models for predictions, and trade millions of $, 2.4% of 1M$ is 20k$! In the scope of statistical validity and profitability we can reliably conclude that the stock market can be predicted with statistical advantage.

No wonder why they only give up to 5% on interest rates!