Homework 2

February 4, 2025

1 Pull in Data and Convert to Monthly

[1]:
```
import yfinance as yf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
apple_data = yf.download('AAPL')
apple_data
```

```
[**********************100%%**********************]  1 of 1 completed
```

[2]:

Date	Open	High	Low	Close	Adj Close	Volume
1980-12-12	0.128348	0.128906	0.128348	0.128348	0.098834	469033600
1980-12-15	0.122210	0.122210	0.121652	0.121652	0.093678	175884800
1980-12-16	0.113281	0.113281	0.112723	0.112723	0.086802	105728000
1980-12-17	0.115513	0.116071	0.115513	0.115513	0.088951	86441600
1980-12-18	0.118862	0.119420	0.118862	0.118862	0.091530	73449600
…	…	…	…	…	…	…
2025-01-29	234.119995	239.860001	234.009995	239.360001	239.360001	45486100
2025-01-30	238.669998	240.789993	237.210007	237.589996	237.589996	55658300
2025-01-31	247.190002	247.190002	233.440002	236.000000	236.000000	101075100
2025-02-03	229.990005	231.830002	225.699997	228.009995	228.009995	72896300
2025-02-04	227.199997	232.949997	226.649994	232.500000	232.500000	

```
2025-02-04   30352449

[11127 rows x 6 columns]
```

[3]:
```
df = apple_data.resample("M").last()[["Adj Close"]]
df
```

[3]:
```
             Adj Close
Date
1980-12-31    0.117313
1981-01-31    0.097116
1981-02-28    0.091100
1981-03-31    0.084224
1981-04-30    0.097545
…                …
2024-10-31  225.661728
2024-11-30  237.330002
2024-12-31  250.419998
2025-01-31  236.000000
2025-02-28  232.500000

[531 rows x 1 columns]
```

2 Create columns.

- Current Stock Price, Difference in stock price, Whether it went up or down over the next month, option premium

[9]:
```
# difference in stock price
df['Diff'] = df['Adj Close'].diff().shift(-1)

# target up or down
df['Target'] = np.sign(df['Diff'])

# option premium
df['Premium'] = .08 * df['Adj Close']
```

[10]:
```
0.117887 - 0.020296
```

[10]:
```
0.09759100000000001
```

[11]:
```
df.head()
```

[11]:
```
            Adj Close      Diff  Target   Premium
Date
1980-12-31   0.117313 -0.020197    -1.0  0.009385
1981-01-31   0.097116 -0.006016    -1.0  0.007769
```

```
1981-02-28   0.091100 -0.006876    -1.0  0.007288
1981-03-31   0.084224  0.013321     1.0  0.006738
1981-04-30   0.097545  0.016329     1.0  0.007804
```

Above, the 'Diff' column is the difference between the Adj Close between that month and the next month. I will call Diff D_X and Adj Close A_X, so $D_1 = A_2 - A_1$, and on and on. A negative Diff means a price decrease over the next month, and a positive Diff means a price increase over the next month. If Diff is negative, Target is assigned -1, and if Diff is positive, Target is assigned +1. The option premium is calculated as the 8% of the adjusted close, which is purely theoretical.

3 Pull in X data, which is already normalized, and build a logistic regression on column 2

[50]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

[51]:
```
Xdata = pd.read_csv("Xdata.csv", index_col = "Date", parse_dates = ["Date"])
Xdata
```

[51]:
```
                VAR1
Date
1980-12-31  0.163261
1981-01-31  0.437449
1981-02-28 -0.334994
1981-03-31  2.550820
1981-04-30  3.170655
…                …
2023-05-31  2.330573
2023-06-30  3.033257
2023-07-31  1.007072
2023-08-31  0.504651
2023-09-30  0.669328

[514 rows x 1 columns]
```

Here, VAR1 is treated as a useful predictor of Apple's stock price.

[52]:
```
# extracting target from df
y = df.loc[:"2023-09-30","Target"].copy()

# full dataset up to 2023-09-30
df = df.loc[:"2023-09-30",:].copy()
```

Logistic Regression Being Run:

$$P(Target = 1) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 VAR_1)}}$$

which calculates the probability of an Apple monthly price increase with the predictor VAR1.

[53]:
```
logreg = LogisticRegression()

logreg.fit(Xdata, y)

y_pred = logreg.predict(Xdata)
```

4 Add columns, prediction and profits.

[54]:
```
df['Predictions'] = y_pred
df
```

[54]:

Date	Adj Close	Diff	Target	Premium	Predictions	Profits
1980-12-31	0.117313	-0.020197	-1.0	0.009385	-1.0	0.0
1981-01-31	0.097116	-0.006016	-1.0	0.007769	-1.0	0.0
1981-02-28	0.091100	-0.006876	-1.0	0.007288	-1.0	0.0
1981-03-31	0.084224	0.013321	1.0	0.006738	1.0	0.0
1981-04-30	0.097545	0.016329	1.0	0.007804	1.0	0.0
…	…	…	…	…	…	…
2023-05-31	175.916260	16.594193	1.0	14.073301	1.0	0.0
2023-06-30	192.510452	2.461319	1.0	15.400836	1.0	0.0
2023-07-31	194.971771	-8.263657	-1.0	15.597742	-1.0	0.0
2023-08-31	186.708115	-16.556961	-1.0	14.936649	-1.0	0.0
2023-09-30	170.151154	-0.437271	-1.0	13.612092	-1.0	0.0

[514 rows x 6 columns]

Options strategy: Likely selling puts, but real mechanics are ignored.

[55]:
```
# initializing profits column
df['Profits'] = 0.

# true positives
df.loc[(df['Predictions'] == 1) & (df['Target'] == 1), 'Profits'] =␣
 ↪df['Premium']

# false positives
df.loc[(df['Predictions'] == 1) & (df['Target'] == -1), 'Profits'] =␣
 ↪100*df['Diff'] + df['Premium']
df
```

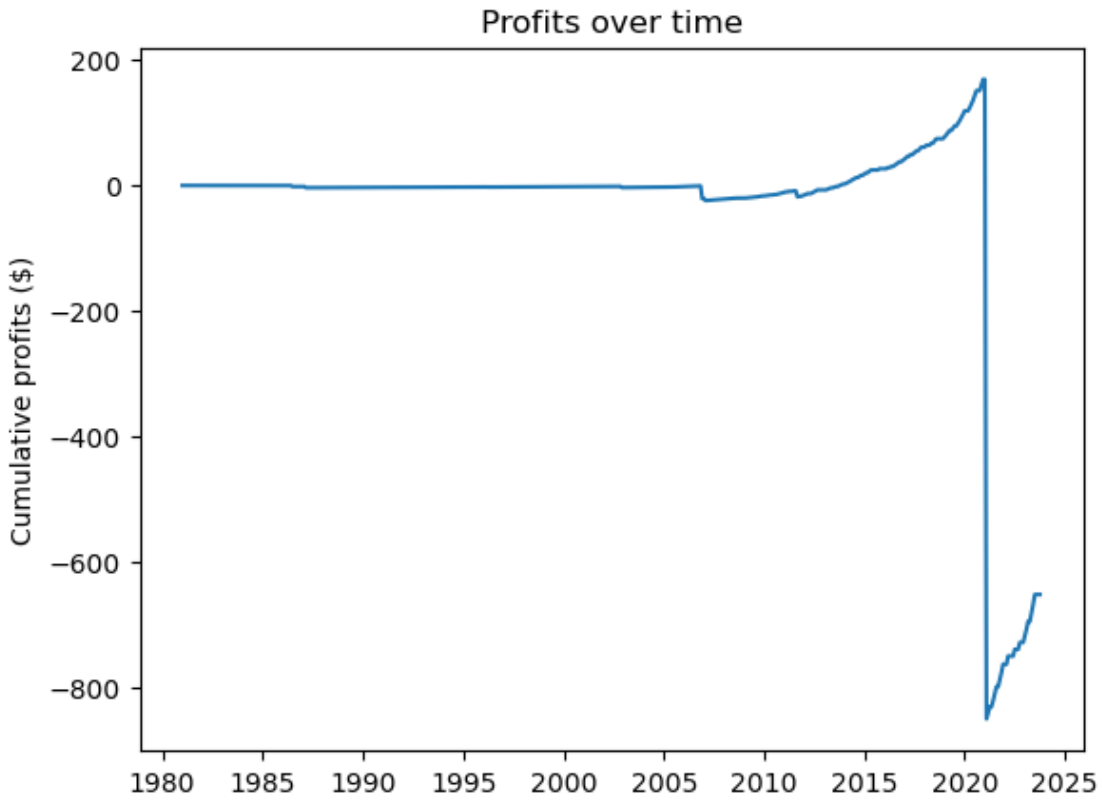
[55]:

Date	Adj Close	Diff	Target	Premium	Predictions	Profits
1980-12-31	0.117313	-0.020197	-1.0	0.009385	-1.0	0.000000
1981-01-31	0.097116	-0.006016	-1.0	0.007769	-1.0	0.000000
1981-02-28	0.091100	-0.006876	-1.0	0.007288	-1.0	0.000000
1981-03-31	0.084224	0.013321	1.0	0.006738	1.0	0.006738
1981-04-30	0.097545	0.016329	1.0	0.007804	1.0	0.007804
…	…	…	…	…	…	…
2023-05-31	175.916260	16.594193	1.0	14.073301	1.0	14.073301
2023-06-30	192.510452	2.461319	1.0	15.400836	1.0	15.400836
2023-07-31	194.971771	-8.263657	-1.0	15.597742	-1.0	0.000000
2023-08-31	186.708115	-16.556961	-1.0	14.936649	-1.0	0.000000
2023-09-30	170.151154	-0.437271	-1.0	13.612092	-1.0	0.000000

[514 rows x 6 columns]

5 Plot profits over time

[56]:
```
plt.plot(np.cumsum(df['Profits']))
plt.ylabel('Cumulative profits ($)')
plt.title('Profits over time')
plt.show()
```



6 Create a loop that re-runs the logistic regression with different thresholds for classifying the strategy predictions and stores the total profits over time.

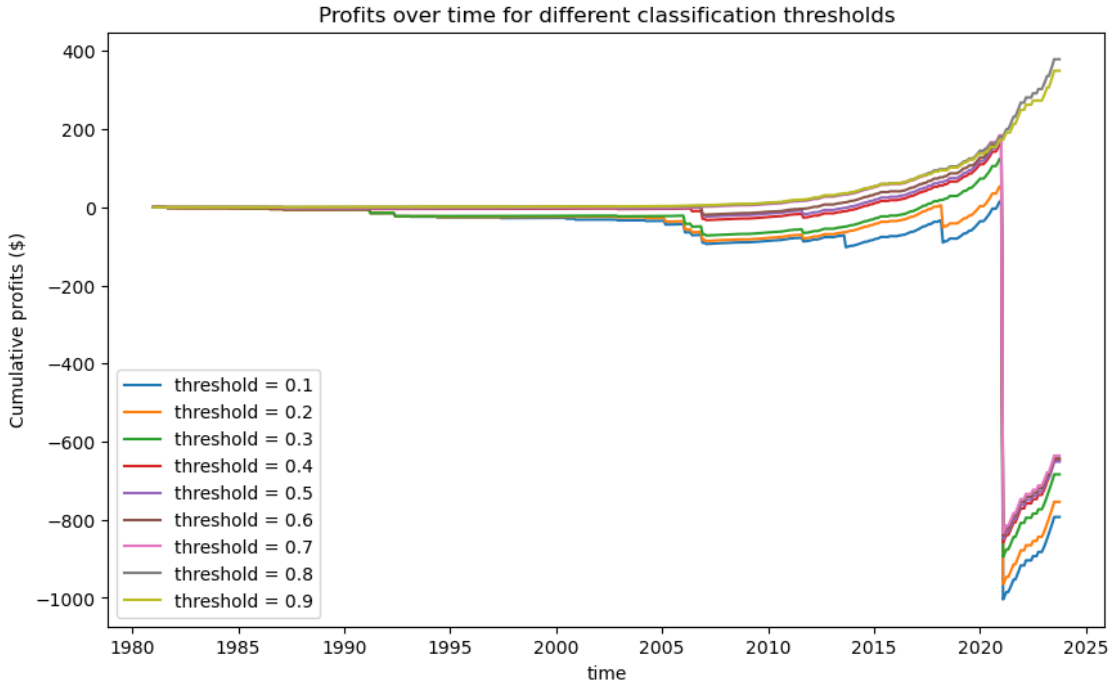
[57]:
```
thresholds = np.arange(0.1, 1.0, 0.1)
profits_over_thresholds = {}

# find index for positive class
pos_class_index = list(logreg.classes_).index(1)
# get probability predictions for positive class
probs = logreg.predict_proba(Xdata)[:, pos_class_index]

for threshold in thresholds:
    predictions = np.where(probs >= threshold, 1, -1)
    temp_df = df.copy()
    temp_df['Predictions'] = predictions
    temp_df['Profits'] = 0.
    # true positives: prediction = 1 and target = 1
    temp_df.loc[(temp_df['Predictions'] == 1) & (temp_df['Target'] == 1),␣
 ↪'Profits'] = temp_df['Premium']
    # false positives: prediction = 1 and target = -1
    temp_df.loc[(temp_df['Predictions'] == 1) & (temp_df['Target'] == -1),␣
 ↪'Profits'] = 100 * temp_df['Diff'] + temp_df['Premium']
    profits_over_thresholds[threshold] = np.cumsum(temp_df['Profits'])
```

7 Plot the different threshold scenarios' profits over time on the same graph so that the trader can pick which strategy to implement.

[58]:
```
plt.figure(figsize = (10, 6))
for threshold, cum_profit in profits_over_thresholds.items():
    plt.plot(cum_profit, label = f"threshold = {threshold:.1f}")
plt.title('Profits over time for different classification thresholds')
plt.xlabel('time')
plt.ylabel('Cumulative profits ($)')
plt.legend()
plt.show()
```



In summary, the strategy is backtested with different classification thresholds (0.1 through 0.9, with increments of 0.1) for the logistic regression, and the profits of each strategy are plotted in the figure above. Notably, a classification threshold of 0.8 and 0.9 did not result in the steep losses during the pandemic when Apple's stock price fell significantly. This shows that the higher the classification threshold, the better. But it is also important to remember that another, possibly more extreme downside event is possible, and that the lower the classification threshold, the fewer number of options will be sold, which will diminish inflation-adjusted profits.