# Assignment 5

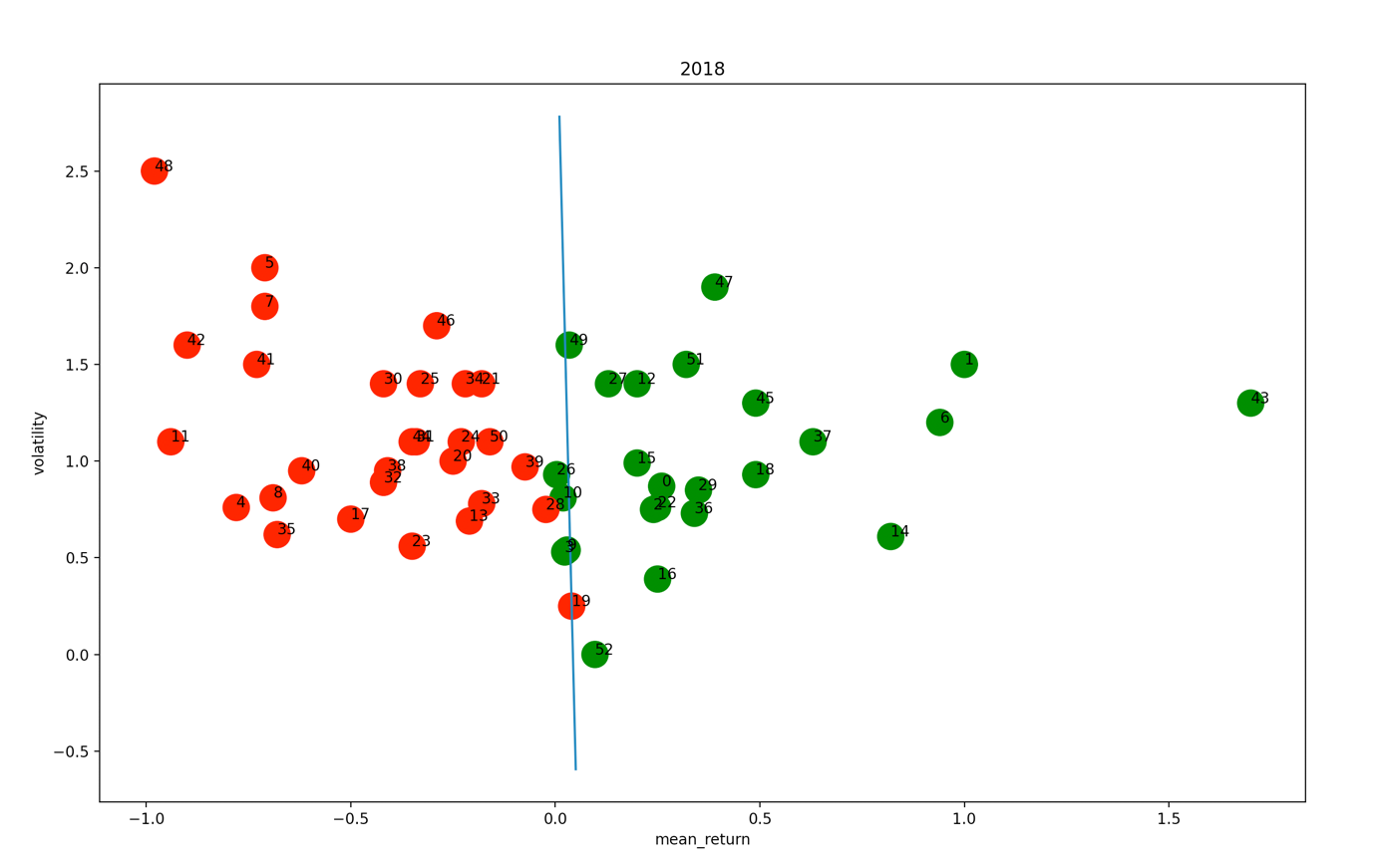
For all code in this assignment, I extracted all the reusable parts and put them in method.py.

## Part 1

For this part I’m not sure the standard scaler gives me the better answer or worse, so I tried both. And for the “implement the strategy”, I asked the professor, and he said it is ok to use the weekly mean\_return to calculate the weekly price, so I just use it. It’s slightly inaccurate, but the chart can show the trend of the year

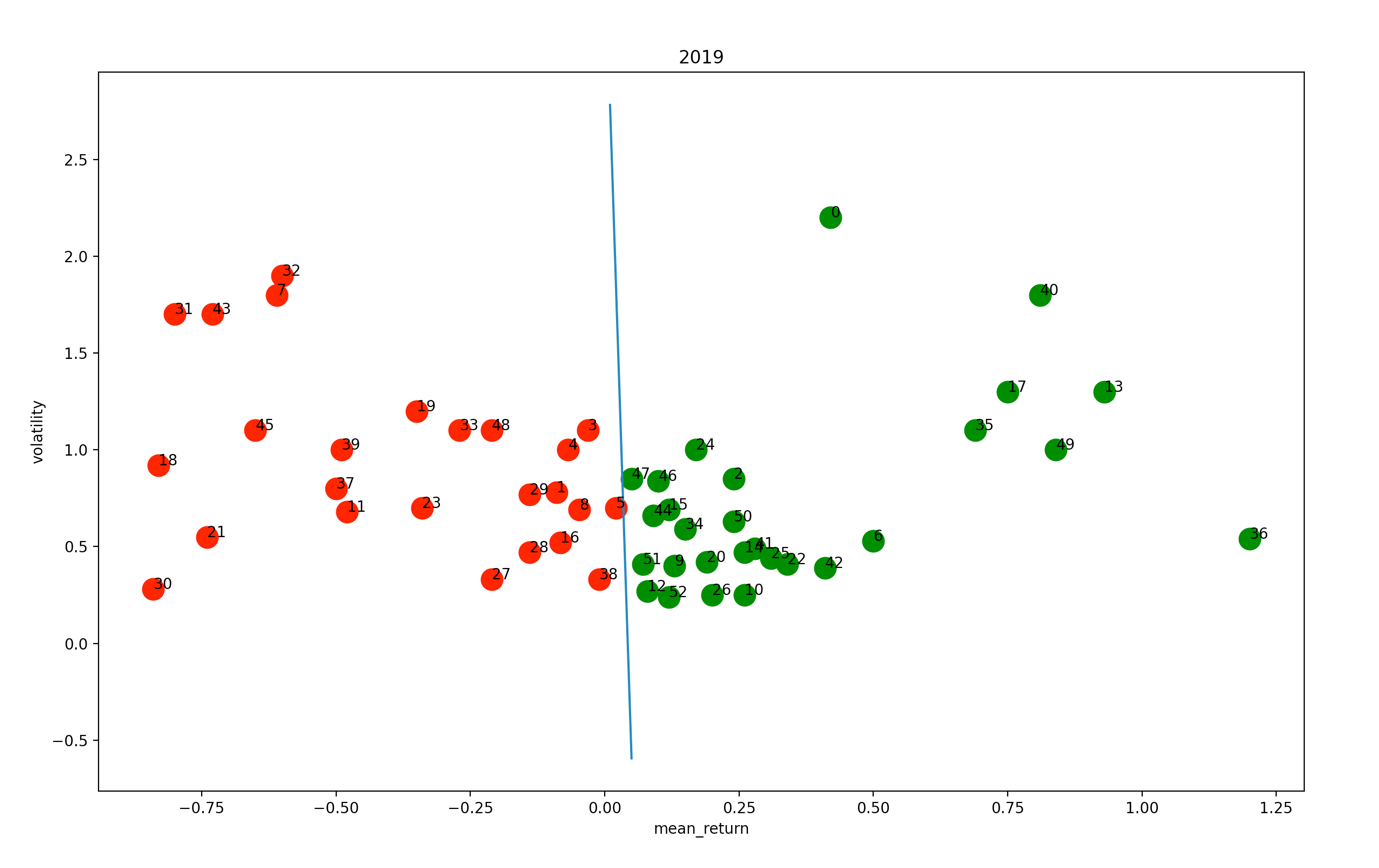
* Linear Separability (part1/linear\_separability.py)

1. By examining the plot of year 2018, I decided to use week 19 and 49 to draw a line

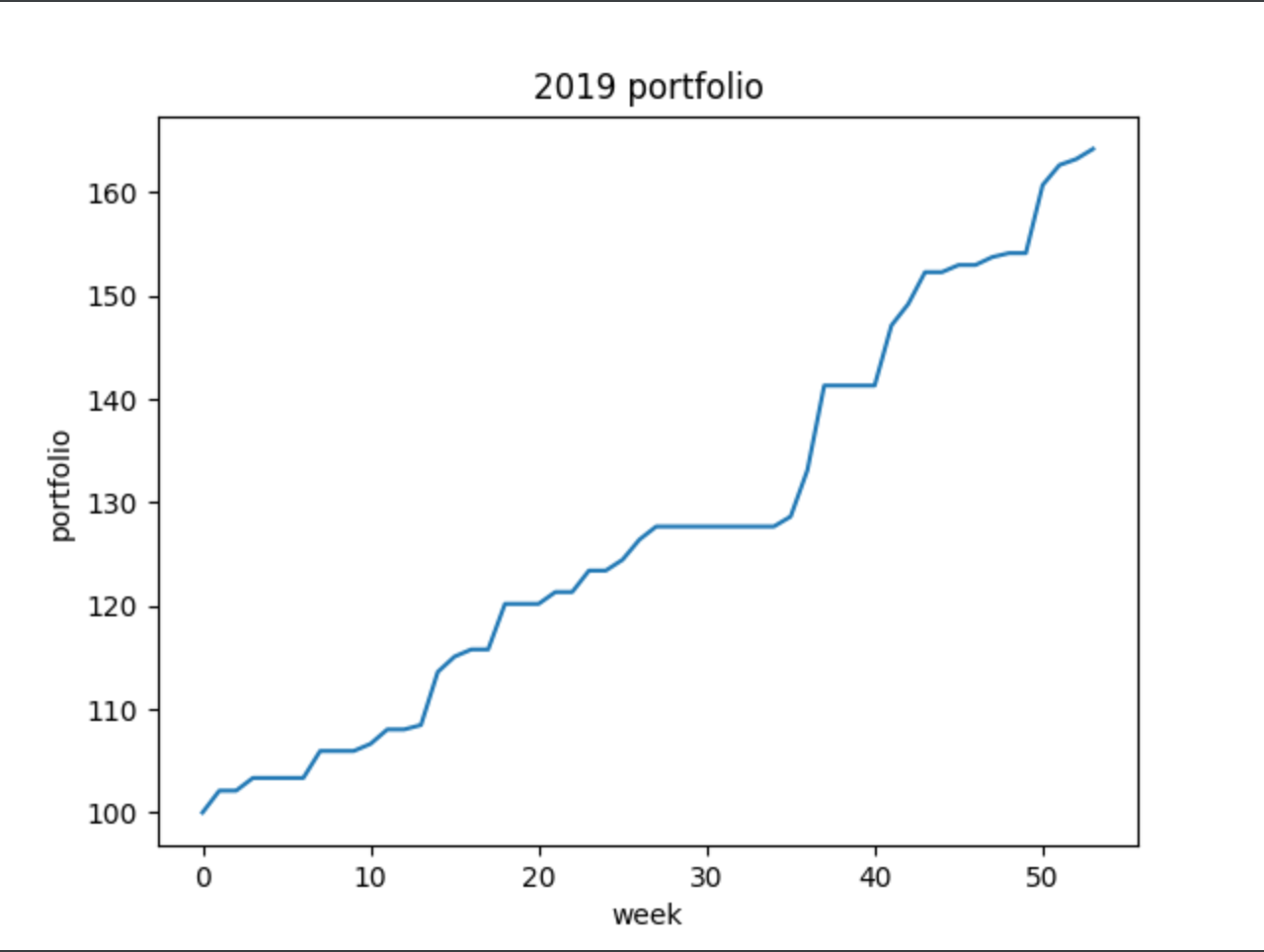


And after draw the line, I remove the week 3, 9, 10, 19, 25, 49.

And the equation is

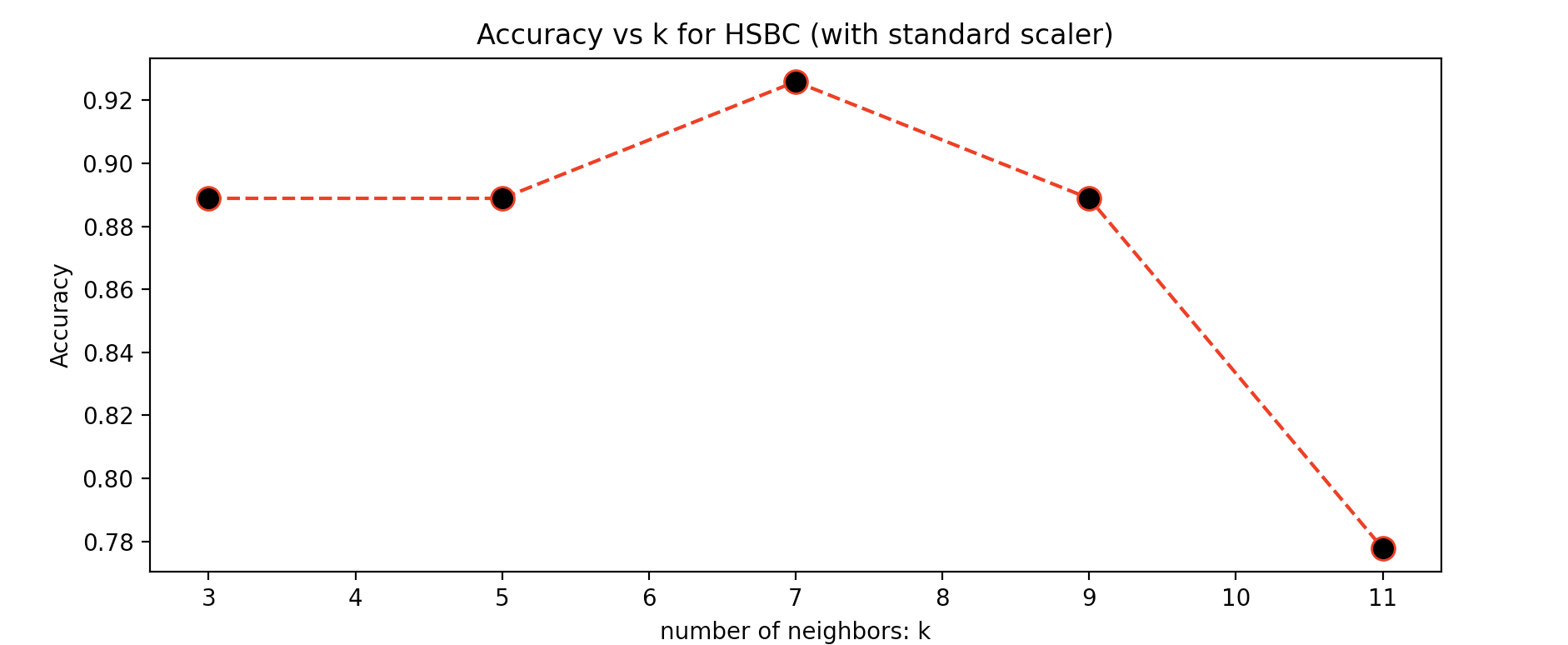


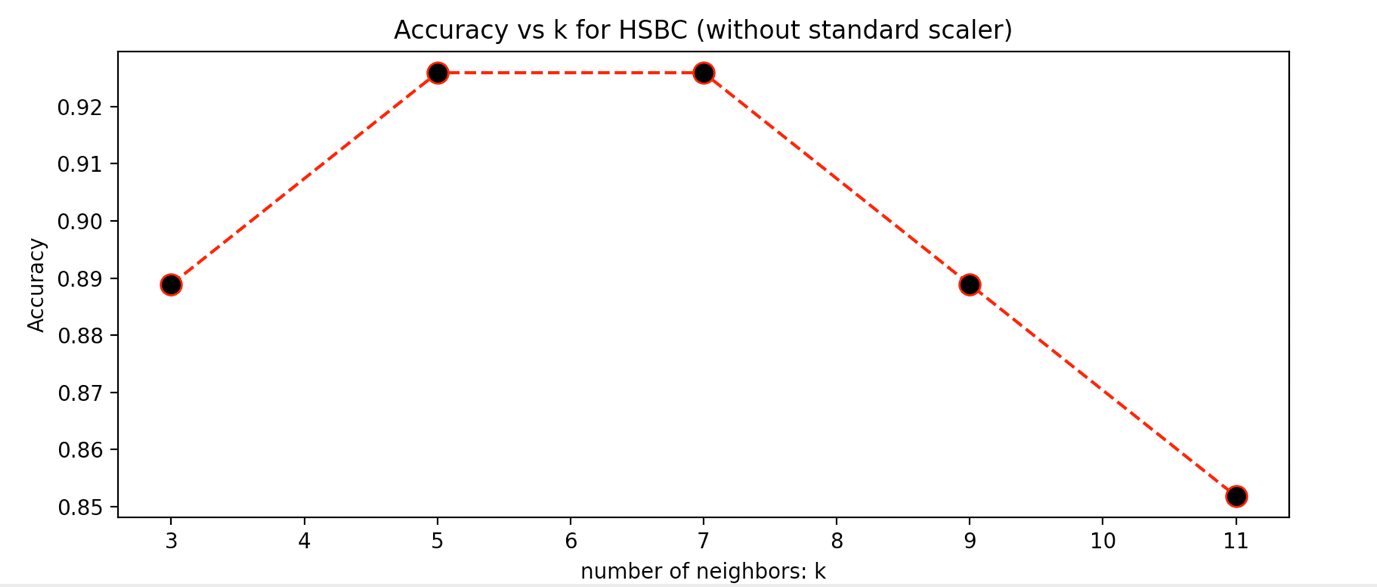
1. Implement trading strategy start with $100



* kNN (part1/kNN.py)

1. plot for k = 3, 5, 7, 9, 11, so the k = 7 is the optimal value. And for the 2 charts, first one I didn’t use the standard scaler and second one I used it. So, compare two ways k = 7 is the optimal value in both charts.





For predict labels in 2019 I used the 2018 data for train and 2019 data for test.

Also, for the accuracy and confusion matrix, I did two edition. One use standard scaler, other one didn’t.

1. accuracy 94.3% (with standard scaler), 96.2% (without)
2. confusion matrix (with standard scaler)

[[26 3]

[0 24]]

confusion matrix (without)

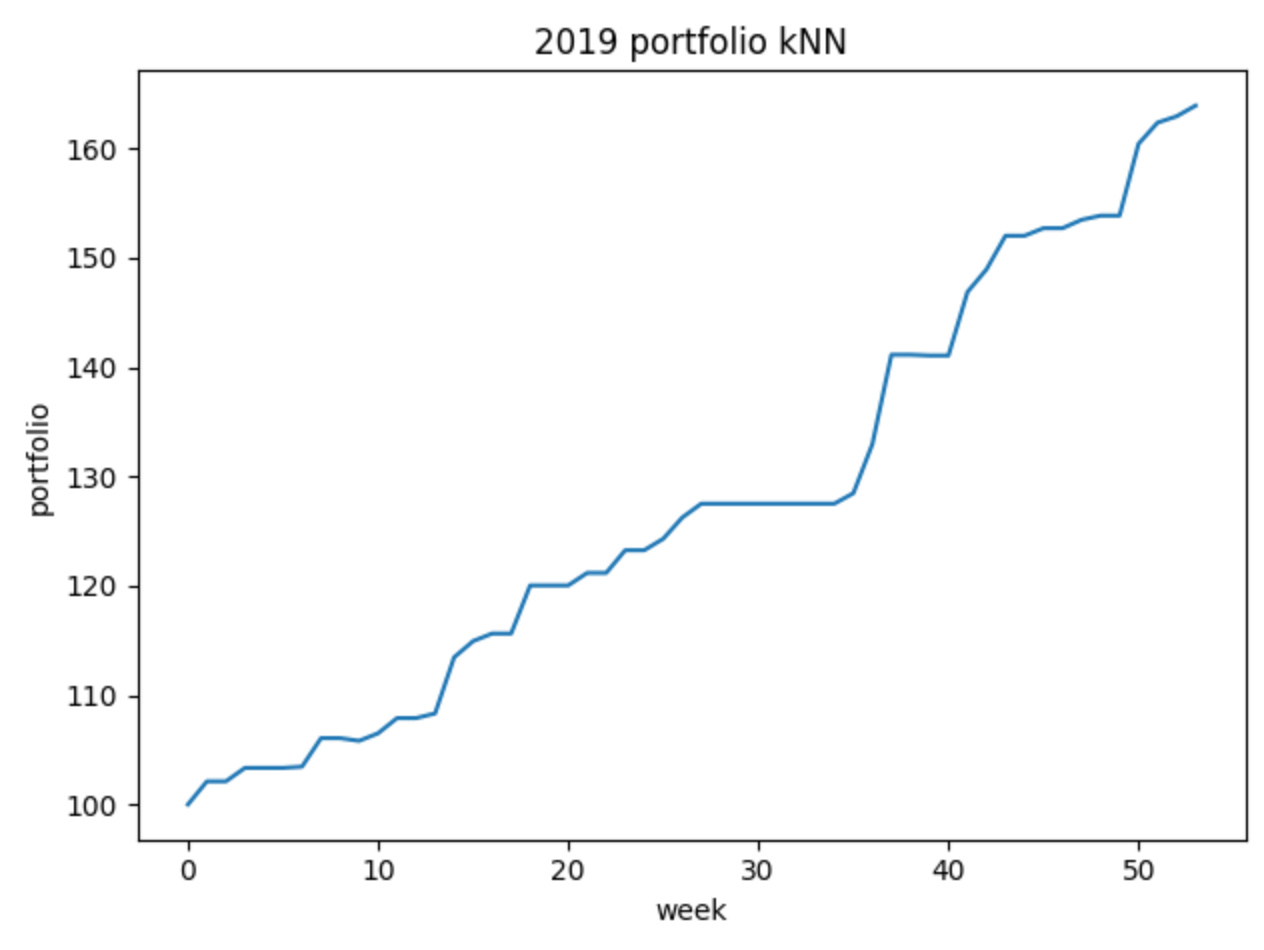
[[29 0]

[2 22]]

1. true positive rate and true negative rate (with standard scaler)

true positive rate and true negative rate (without)

1. Implement trading strategy start with $100, without standard scaler is more accuracy. So, this strategy didn’t use that.



* Logistic regression (part1/logistic\_regression.py)

1. Equation:
2. Accuracy 96.2% (with standard scaler) 98.1% (without)
3. Confusion matrix (with standard scaler)

[[27 2]

[0 24]]

Confusion matrix (without)

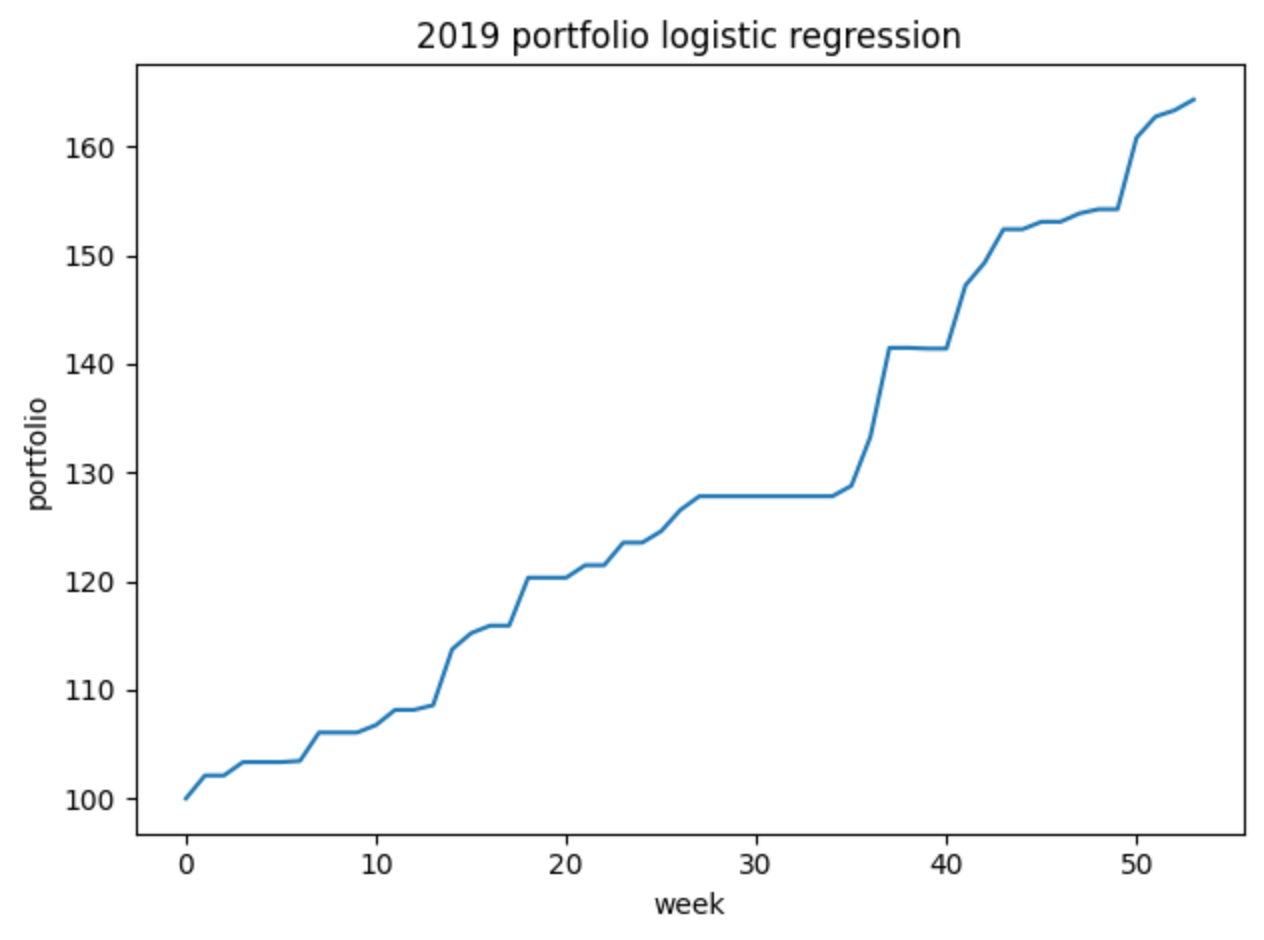
[[29 0]

[1 23]]

1. True positive rate and true negative rate (with standard scaler)

True positive rate and true negative rate (without)

1. Implement trading strategy start with $100, without standard scaler is more accuracy. So, this strategy didn’t use that.



## Part 2

* Question 1

1. Shown by running code part2/basic.py, and data saved in data\_banknote\_color.csv
2. Table

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Class |  |  |  |  |  |  |  |  |
| 0 | 2.28 | 2.02 | 4.26 | 5.14 | 0.80 | 3.24 | -1.15 | 2.13 |
| 1 | -1.87 | 1.88 | -0.99 | 5.40 | 2.15 | 5.26 | -1.25 | 2.07 |
| All | 0.43 | 2.84 | 1.92 | 5.87 | 1.40 | 4.31 | -1.19 | 2.10 |

1. For feature 1, 2 and 4 the mean of class 0 is greater than class 1, and for feature 3 the mean of class 0 is smaller than class 1. And the standard deviation of feature 1, 2 and 4 between class 0 and 1 is nearly, but for feature 3 has a large difference.

* Question 2

1. Shown by running code part2/simple\_classifier.py, and plot saved in /plots folder
2. Three simple comparisons: f\_1>2, f\_2>5, f\_3<8
3. Code part2/question2.py
4. Code part2/question2.py
5. Table

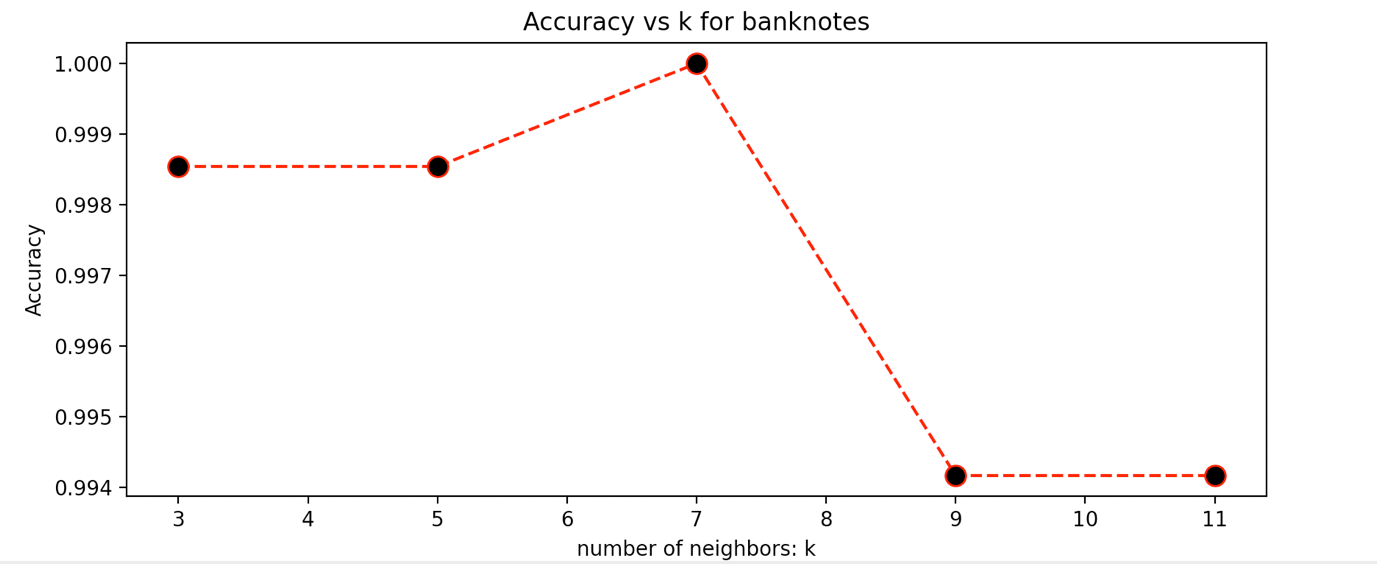
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | accuracy | TPR | TNR |
| 111 | 0 | 301 | 274 | 60.1% | 28.8% | 100% |

1. My simple classifier gave me higher accuracy on identifying “fake” bills, and my accuracy is better than 50%.

* Question 3

In this question, I didn’t use standard scaler, because if I use it the accuracy of all five k will be 1.00.

1. Shown by running code part2/kNN.py
2. Graph



1. Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | accuracy | TPR | TNR |
| 385 | 0 | 301 | 0 | 100% | 100% | 100% |

1. Yes, my k-NN classifier is better than the simple classifier.
2. My BUID is 9113, if use the simple classifier the class label is ‘red’, and if use the k-NN with k = 7 the label is ‘green’.

* Question 4

1. Drop feature
   1. Drop f1 accuracy: 96.2%
   2. Drop f2 accuracy: 97.2%
   3. Drop f3 accuracy: 96.4%
   4. Drop f4 accuracy: 98.4%
2. Of cause not, because the k = 7 gives 100% accuracy when use 4 features
3. Feature 1 contributed the most to loss of accuracy
4. Feature 4 contributed the least to loss of accuracy

* Question 5

1. Shown by running code part2/logistic\_regression.py
2. Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TP | FP | TN | FN | accuracy | TPR | TNR |
| 373 | 1 | 300 | 12 | 98.1% | 96.9% | 99.7% |

1. Yes, my logistic regression is better than the simple classifier.
2. No, because the accuracy of my k-NN is 100%.
3. My BUID is 9113, if I use the logistic regression the label is ‘green’, and it is same as the label predicted by k-NN.

* Question 6
  1. Drop feature
     1. Drop f1 accuracy: 82.1%
     2. Drop f2 accuracy: 89.4%
     3. Drop f3 accuracy: 87.3%
     4. Drop f4 accuracy: 98.1%
  2. Only drop feature 4 will get the same accuracy with all 4 features used, others are lower than that.
  3. Feature 1 contributed the most to loss of accuracy
  4. Feature 4 contributed the least to loss of accuracy
  5. Yes, the contributed most feature is same as k-NN and the contributed least feature is also same as k-NN.