CS271

Dongho Joseph Lee

# Assignment 2 Final Report

## How the code works

### 1. Data Preprocessing

a. Read the data files in the same directory.
b. Change the string data into number for calculation. For example, ‘sunny’ will be 0, ‘overcast’ will be 1, ‘rain’ will be 2.
c. If the range of the value is too big, check the minimum and maximum and divide the max - min by 40 to get interval of the value. Then, modify the value with the interval.
d. Split the data into 2 parts, train set and test set.

### 2. Build Tree

a. Calculate the information gain, entropy or gini index to find the appropriate attribute to make the beginning of the tree.
b. Check the values in the attribute to get proper point to make tree.
c. Call the build tree function recursively.

### 3. Prediction

a. Get the first row of the test set and tree to make a prediction.
b. For every class of the test set, check the class of the tree is leaf or not.
    i. If it is leaf, return the target, yes or no.
    ii. If it is not, it will call predict function again recursively to go to the leaf condition.
c. If the predicted result is same as test, it will be true positive or true negative.
    i. If the result is yes, it is true positive. If the result is no, it is true negative.
d. If the predicted result is not same as test, it will be false positive or false negative.
    i. If the result is yes, it is false positive. If the result is no, it is false negative.

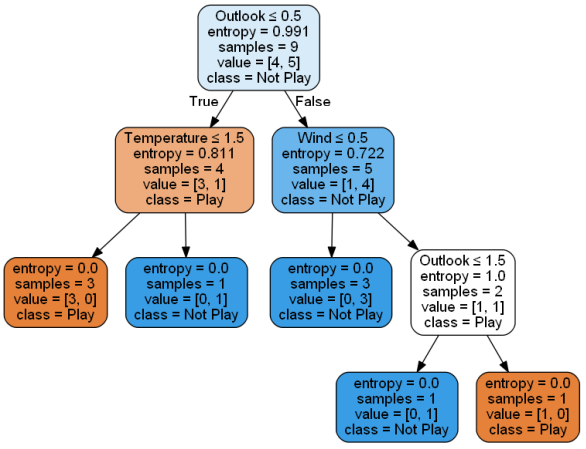
### 4. Analysis and data output

a. Calculate the accuracy, precision and recall and print on the console.
b. Make a figure of confusion matrix and heatmap.
c. Make a graph with train size = 0.5, 0.6, 0.7, 0.8, and 0.9 cases to check the overall accuracies.

# Analysis

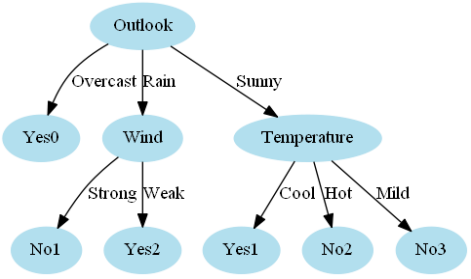
## 1. Sample Dataset

a. Graph of decision tree




i.

ii. It is the library version.

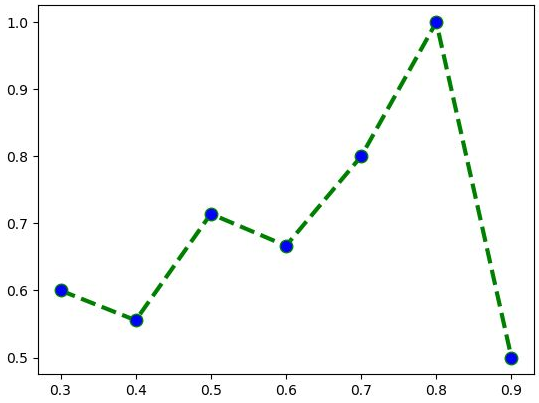



iii.

iv. It is the scratch version.

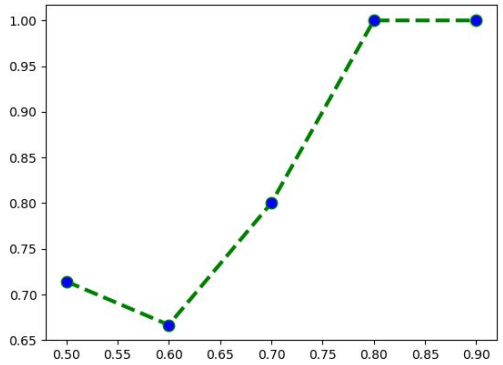
v. There are slight difference between 2 graphs. I assume this difference is coming from the algorithm of scikit-learn library. Scikit-learn only can make the decision tree with number, it required the process to change the status as value. Therefore, I can see the case like outlook <= 0.5. I think it is very weird because there are no middle status in string.

b. Accuracy graph




i.

ii. It is the library version. X axis is train set rate, y axis is accuracy.

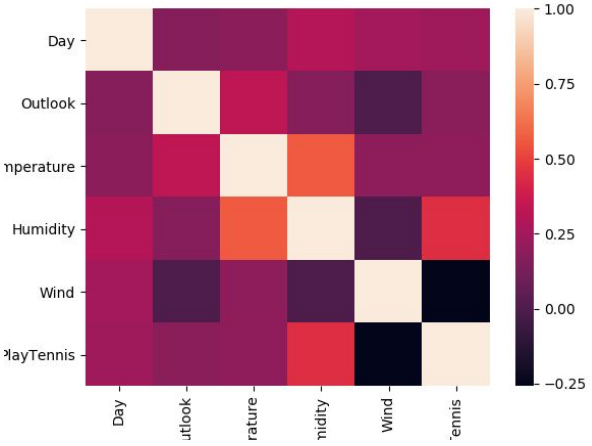



iii.

iv. It is the scratch version. X axis is train set rate, y axis is accuracy.

v. By compared 2 graphs, it is possible to know that the most accurate portion is train set = 0.8, test set = 0.2 in this given sample dataset. However, it is also possible to know that dataset is too small because the accuracy's difference between train set = 0.7 and 0.8 is too huge.

c. Heatmap correlation with features




i.

ii. It uses the whole dataset to calculate the graph. Therefore, there is no difference between scratch version and library version.

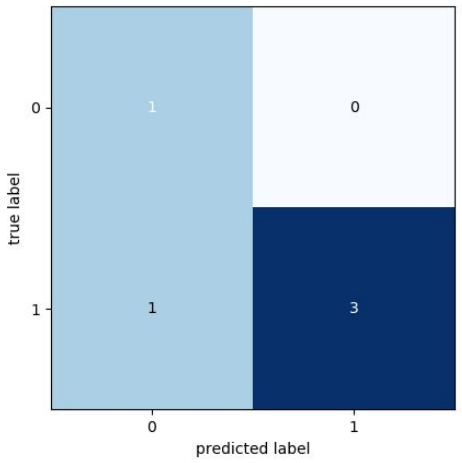
d. Heatmap correlation with features and target


i.

ii. It uses the whole dataset to calculate the graph. Therefore, there is no difference between scratch version and library version.

e. Accuracy / Recall / Precision / Confusion Matrix

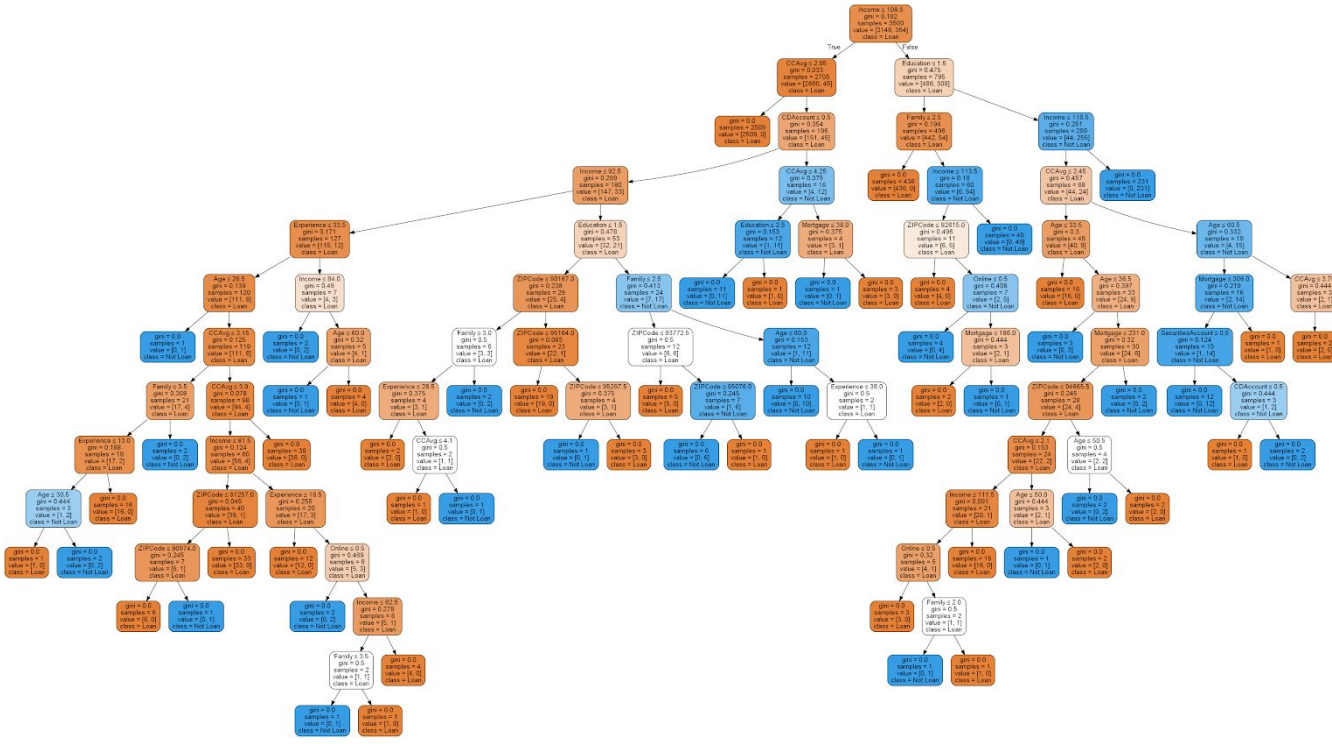
i.



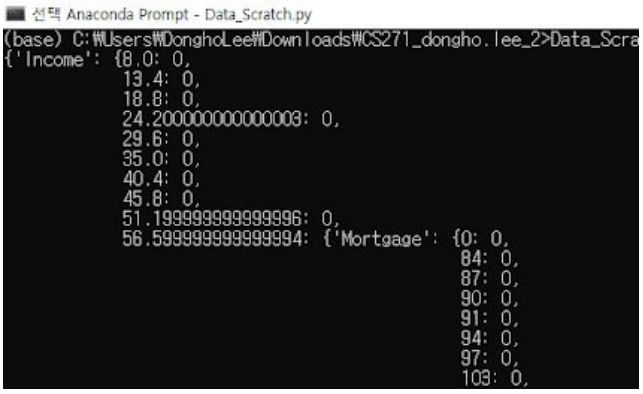

ii. This is the confusion matrix from the both library and scratch version when the rate of train set is 0.7. The array is [[T N, F P], [F N, T P]]. From the matrix, it is possible to calculate the accuracy, precision and recall. Accuracy is 0.8, precision is 0.75 and recall is 1. Because the confusion matrix is perfectly same, I can assume that the overall implementation is correct.

## 2. Given Dataset

a. Graph of decision tree



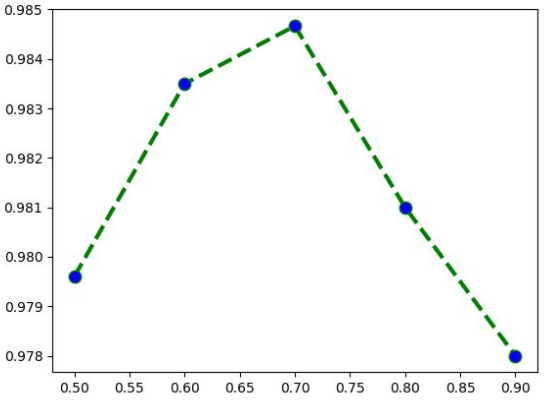
i. It is the library version decision tree.




ii. It is the scratch version decision tree.

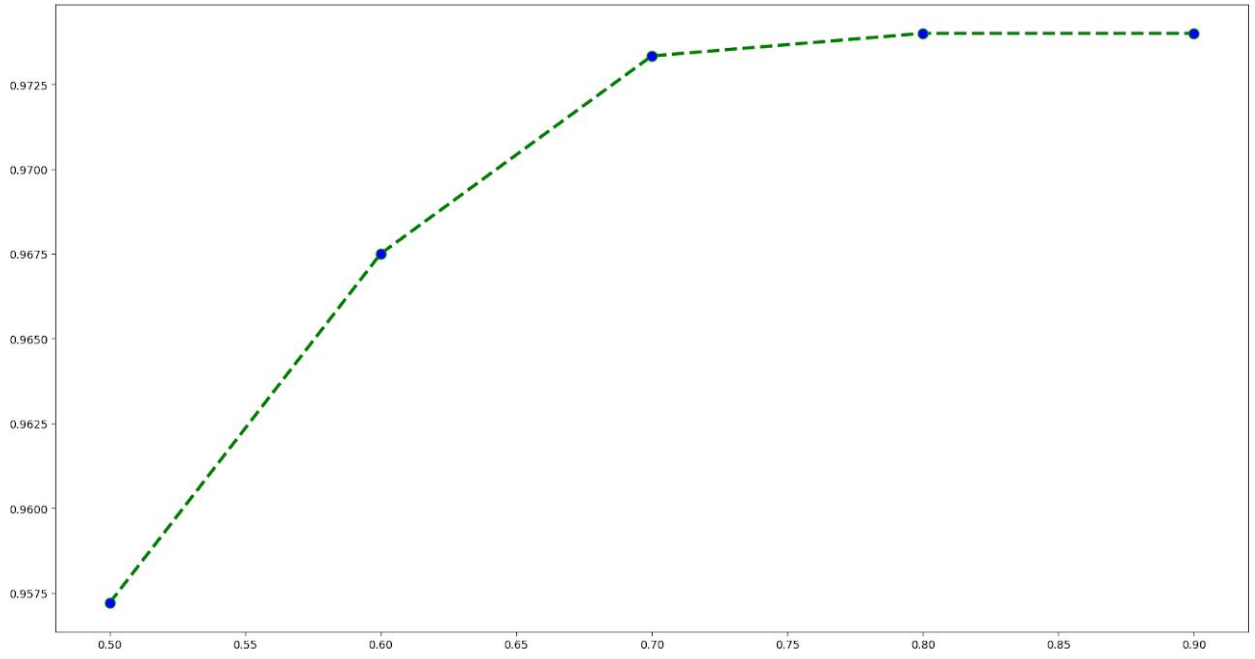
iii. I used pprint to express the decision tree. Because the tree is too wide, it was impossible to draw with my own graph visualizer and also impossible to analysis.

b. Accuracy graph




i.

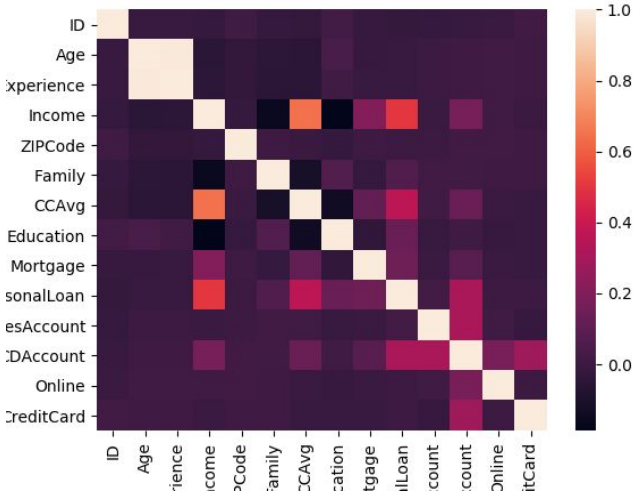
ii. It is the library version. X axis is train set rate, y axis is accuracy.



iii. It is the scratch version. X axis is train set rate, y axis is accuracy.

iv. By compared 2 graphs, it is possible to know that the accuracy is high enough. From this result, I can assume that the overall implementation is correct.

c. Heatmap correlation with features




i.

ii. It uses the whole dataset to calculate the graph. Therefore, there is no difference between scratch version and library version.

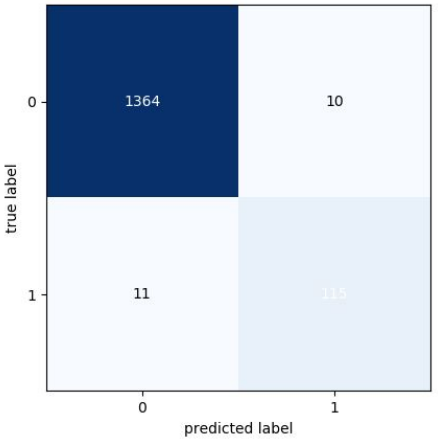
d. Heatmap correlation with features and target


i.

ii. It uses the whole dataset to calculate the graph. Therefore, there is no difference between scratch version and library version.

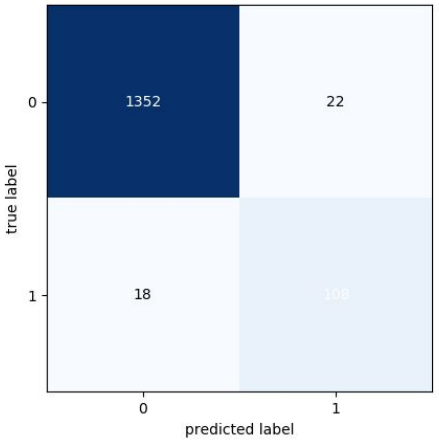
e. Accuracy / Recall / Precision / Confusion Matrix

i.




ii. This is the confusion matrix from the library version when the rate of train set is 0.7.

iii.




iv. This is the confusion matrix from the library version when the rate of train set is 0.7.

v. From 2 matrices, it is possible to calculate the accuracy, precision and recall. True positive, true negative, false positive, and false negative are all almost similar, accuracy, precision and recall are all similar, too. From this result, I can assume that the overall implementation is correct.