Project 1

SiCheng Yi

The purpose of this project is to implement a decision tree classifier to classify some data (the breast cancer data used in the project).

There are many groups of these data, and each array contains 11 data:

1. Samplecodenumber:idnumber(Note:Thisisnotafeature.)

2. ClumpThickness:1–10

3. UniformityofCellSize:1–10

4. UniformityofCellShape:1–10

5. MarginalAdhesion:1–10

6. SingleEpithelialCellSize:1–10

7. BareNuclei: 1–10

8. BlandChromatin: 1–10

9. NormalNucleoli:1–10

10. Mitoses: 1–10

11. Class(2isBenign, 4isMalignant)

Before starting, we should check all the data, open the file "breast-cancer-wisconsin", we can see that each set of data is composed of 11 numbers, the first number is meaningless, and the last number is 2 or 4. The others are one of 1 to 10. Therefore, we can classify the last data first, similar to 0 and 1, and then classify other data. Other data is any one from 1 to 10, which belongs to continuous data. For simplicity, we need to convert it into discrete data. We can determine which data should be classified according to their information gain and then according to the information gain. By calculating the average of the sum of the respective averages of the two groups of "2" and "4", a value can be obtained, and they can be put into the two categories according to the value greater than or less than this value. We classify the first 90% as test data, and the last 10% as training data.

But when reading the file, I encountered difficulties. I tried the following functions, but an error occurred:

def createDataSet():

# 1. Read in data from file.

print("Reading File...")

with open("breast-cancer-wisconsin.data") as fp:

reader = csv.reader(fp, delimiter=",", quotechar='"')

dataSet = [row for row in reader]

print

labels = ['Samplecodenumber','ClumpThickness','UniformityofCellSize','UniformityofCellShape','MarginalAdhesion','SingleEpithelialCellSize','BareNuclei','BlandChromatin','NormalNucleoli','Mitoses']

return dataSet, labels

In order to ensure the degree of completion, I directly copied part of the series. I tried to delete the first column, using:

dataSet = np.delete(dataSet,0,axis=1), but an array will appear in the deleted sequence.

Information entropy-conditional entropy = information gain, and the maximum amount of information gain is the basis for determining the node currently to be used. The parameters I used are described in the program notes.

prob = float(labelCounts[key]) / numEntries

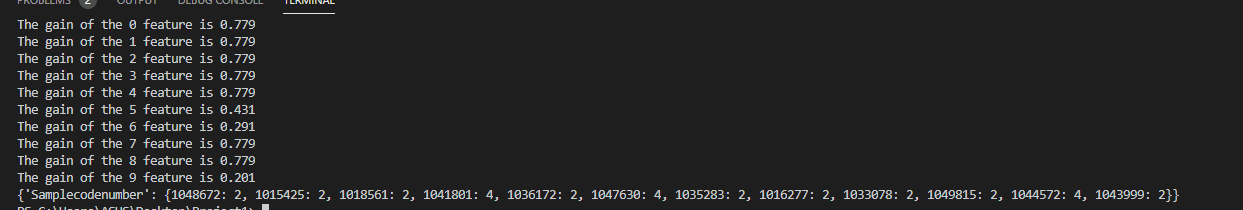
shannonEnt -= prob \* log(prob,2)

Conditional entropy += prob \* calcShannonEnt(subDataSet) #∑pH(Y|X=xi)

infoGain = baseEntropy-newEntropy, information gain = information entropy-conditional entropy

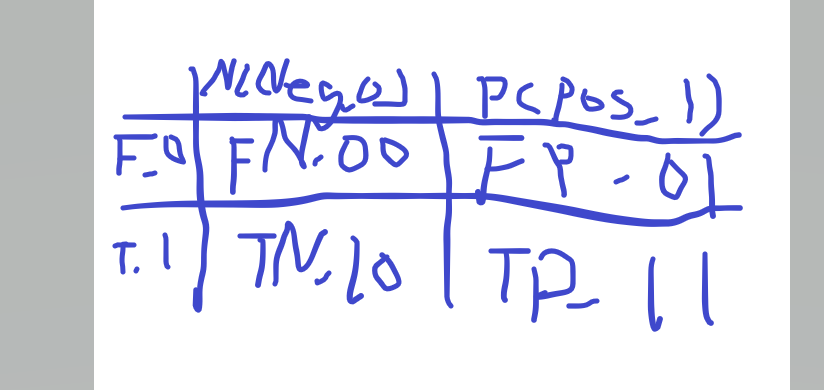
These variables are the frequency of the data and its calculation results.

Result:



Here the tree is represented in the form of parentheses, the data inside, the lower the position of the tree.

The result is in the form of a set. When I calculate the information gain, I calculated all the gains (because the first data cannot be cleared smoothly, the accuracy is problematic), and then take the final two classification as an example. Suppose 2 is a positive example-Positive, and 4 is a negative example-Negative; if the prediction is correct, it is True, otherwise it is False. We can get the following table representing FP, FN, TP, TN:



Then like:

(

y\_true, # array, Gound true (correct) target values

y\_pred, # array, Estimated targets as returned by a classifier

labels=None, # array, List of labels to index the matrix.

sample\_weight=None # array-like of shape = [n\_samples], Optional sample weights

)

