Programming Assignment 1: Decision Tree Report

1. Group member

   a) Hao Yang, USCID: 5284267375. (Contributions: All)

2. Part 1: Implementation

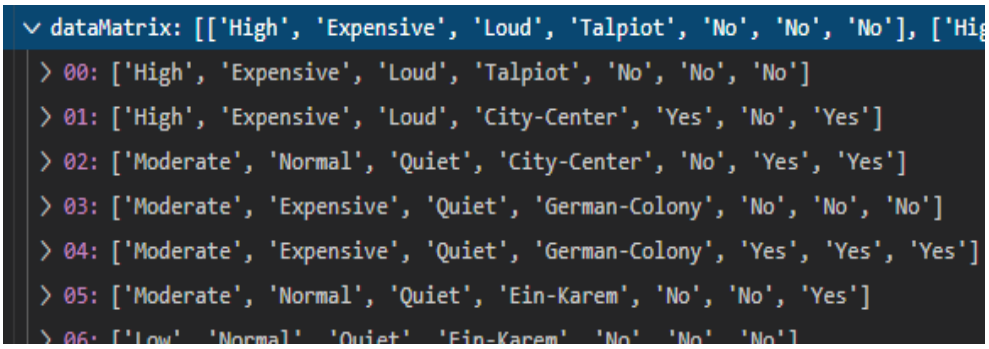
   1. Programming language: Python 3.7

   2. Libraries used: math, copy, random

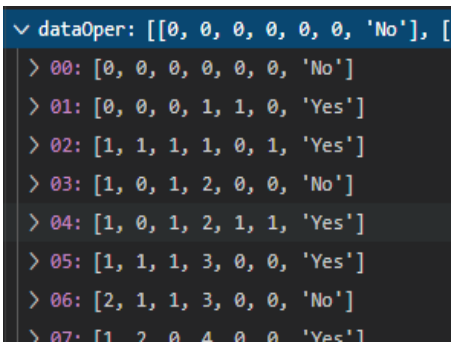
   3. Decision tree algorithm: ID3 principle, based on splitting by information gain.

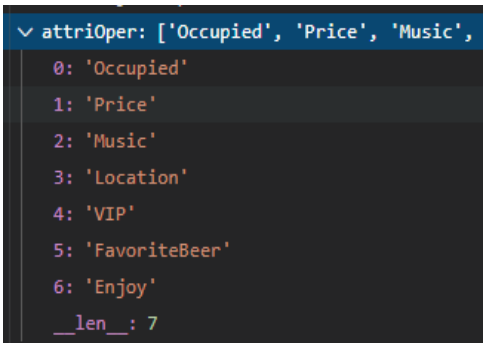
   4. Script description:

      1. Data processing: read txt file, save first line as list of attributes, save each line as dataset splitting by ‘,’.



      2. Transform the value of each attribute into int type (0 ~ k, based on how many different values in this attribute). Then getting the training dataset (dataOper) and attributes list(attriOper).

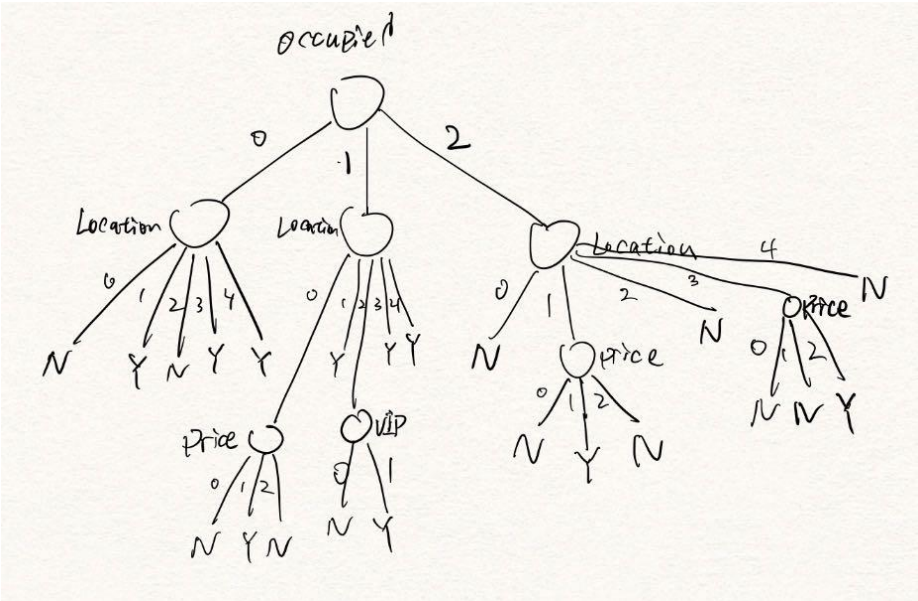




3. Construct decision tree based on information gain of each attribute (input: list of datasets, list of attributes, output: json-like dictionary of tree, for each pair in dictionary, the key means parents-node and the value of key means the children of that node. The value of key is a list of ‘Yes’/’No’ or a dictionary represents leaf and sub tree respectively).

   Decision tree result:


```
result: {'Occupied': [{...}, {...}, {...}]}
 'Occupied': [{'Location': [...]}, {'Location': [...]}, {'Location': [...]}]
  0: {'Location': ['No', 'Yes', 'No', 'Yes', 'Yes']}
   'Location': ['No', 'Yes', 'No', 'Yes', 'Yes']
    __len__: 1
  1: {'Location': [{...}, 'Yes', {...}, 'Yes', 'Yes']}
   'Location': [{'Price': [...]}, 'Yes', {'VIP': [...]}, 'Yes', 'Yes']
    0: {'Price': ['Yes', 'Yes', 'No']}
    1: 'Yes'
    2: {'VIP': ['No', 'Yes']}
    3: 'Yes'
    4: 'Yes'
    __len__: 5
   __len__: 1
  2: {'Location': ['No', {...}, 'No', {...}, 'No']}
   'Location': ['No', {'Price': [...]}, 'No', {'Price': [...]}, 'No']
    0: 'No'
    1: {'Price': ['No', 'Yes', 'No']}
    2: 'No'
    3: {'Price': ['Yes', 'No', 'Yes']}
    4: 'No'
    __len__: 5
```

5. Optimizations

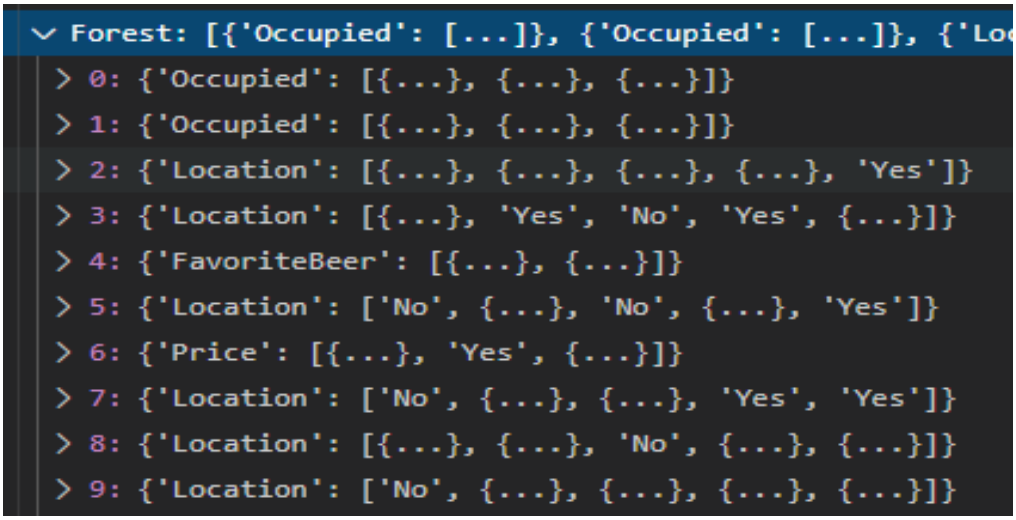
    1. Algorithm optimizations:

        Used random forest optimization.Bootstrapping: randomly resample from original dataset to get a series of smaller dataset and use them to generate a series of decision tree to be used to predict new data.

        Generate forest of 10 trees with size 15:

```
Forest = Forest(dataOper,attriOper,15,10)
```

        Forest, a list of decision tree:



    2. Code-level optimizations: recursively write code to achieve generating decision tree, which can make code more brief and easier to read.

6. Challenges faced

    1. Write recursive code requires tighter logic. It's easier to get bug, but it helps me understand local and global variables better.

    2. I found that if I use A = B directly to copy a list, value of list B will be changed when I change the value of list A because they point to a same

address. But I fixed this problem with using A = copy.deepcopy(B).

7. Result of prediction

   Run code below:

```
test_case = ['Moderate', 'Cheap', 'Loud', 'City-Center', 'No', 'No']

predic = Prediction(test_case, result)

Forest = Forest(dataOper,attriOper,15,10)

predic2 = Prediction_Forest(test_case,Forest)

print(result)
print('Prediction of ID3 decision tree is:', predic)
print('Prediction of ID3 decision tree with random forest is:', predic2)
```

   Result:

```
PS D:\Users\yangh\Desktop\552\HW1> & C:/Users/yangh/AppData/Local/Programs/Python/Python37/python.exe d:/Users/yangh/Deskt
op/552/HW1/DecisionTrees.py
{'Occupied': [{'Location': ['No', 'Yes', 'No', 'Yes', 'Yes']}, {'Location': [{'Price': ['Yes', 'Yes', 'No']}, 'Yes', {'VIP
': ['No', 'Yes']}, 'Yes', 'Yes']}, {'Location': ['No', {'Price': ['No', 'Yes', 'No']}, 'No', {'Price': ['Yes', 'No', 'Yes'
]}, 'No']}]}
Prediction of ID3 decision tree is: Yes
Prediction of ID3 decision tree with random forest is: Yes
PS D:\Users\yangh\Desktop\552\HW1>
```

   (The format description of the decision tree is on the 5th section above).

3. Part 2: Software familiarization

   1. Library function: sklearn.tree

   2. How to use: use clf = tree.DecisionTreeClassifier(criterion=’entropy’) to create tree. Then use clf.fit(x,y) to train and generate decision tree with training data x,y which is list of dataset and label respectively. Then use clf.predict([[test case]]) to predict the class of test case. ([1] means ‘Yes’, which is same as the result of my code).

```
for rows in dataOper:···
xx = np.array(X)
yy = np.array(Y)

clf = tree.DecisionTreeClassifier(criterion='entropy')
print(clf)
clf.fit(xx,yy)
print(clf.predict([[1,2,0,1,0,0]]))
```

Result:

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
[1]
```

3. Comparison: sklearn.tree use C4.5 or CART decision tree principle to generate decision tree, which perform better than ID3 principle. Because ID3 principle is based on the sum of information gain of each value of attribute, it will tend to choose the attributes with more possible values. Therefore, use C4.5 principle, which use information gain rate to generate decision tree will perform better. And CART decision tree use Gini index to divide attributes, which select sample randomly from dataset, can perform better. Moreover, sklearn.tree has some helpful index to achieve pruning of decision tree to avoid overfitting, which is what my implement don't have. And sklearn.tree can treat with continuous value, my implement can only treat with discrete value. However, I implemented random forest by bootstrapping to improve performance of decision tree.

4. Part 3: Applications

The essential thing of decision tree is give an answer of 'which class does it belong to'(Yes or No in many cases) with a series of judgement for it attributes. An application of decision tree for a big case today is it can be used in judging whether a patient has coronavirus(result) with patients' symptom(attributes). Through a series of diagnosis, if having cough, fever, dyspnea etc., decision tree can make a preliminary screening of which patients are more likely to have been infected.