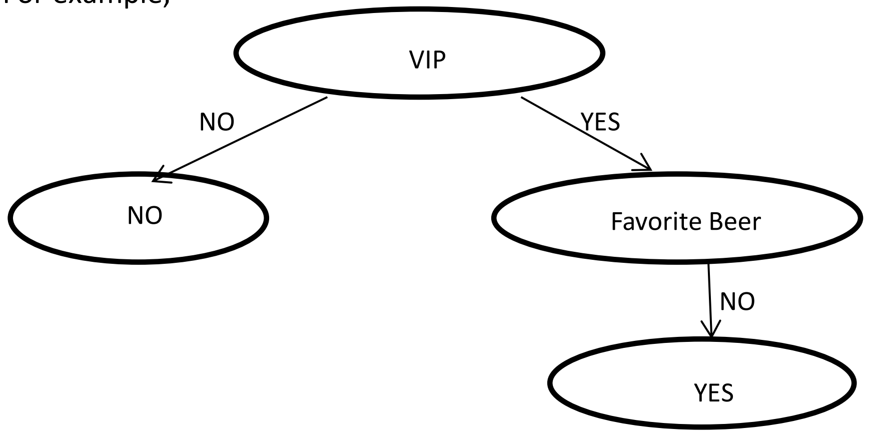
Programming Assignment 1: Decision Trees

Part 1: Implementation

(a) Write a program to construct decision trees based on the idea of splitting by Information Gain.

·Software:
Python3.6 +numpy,math,collection,
For visualization:uuid

·Data Structure:
We use a dictionary to store the branches and layers information. The "key" of the dictionary is an attribute or the value of an attribute and we use the key to split trees and grow branches. The "value" of the dictionary is a pure result of a dictionary which contains a sub-tree.
For example,



The father node is the attribute VIP. Son nodes of attribute VIP are result "NO" and attribute Favorite Beer. Son node of attribute Favorite Beer is result "YES". The dictionary data structure should be {'VIP':{ {'NO':'NO'}, {'YES':{'Favorite Beer':{'NO':'YES'}}}}

Each row of the data set is a list, which contains the values of each attribute.
The total data set is a list contains all the rows of dataset. The length of the data set list is the number of rows of the data. We name it "data_set".
The result of each row is in another list. We name it "classes".
Feature names are stored in a list. We name it "feat_names".

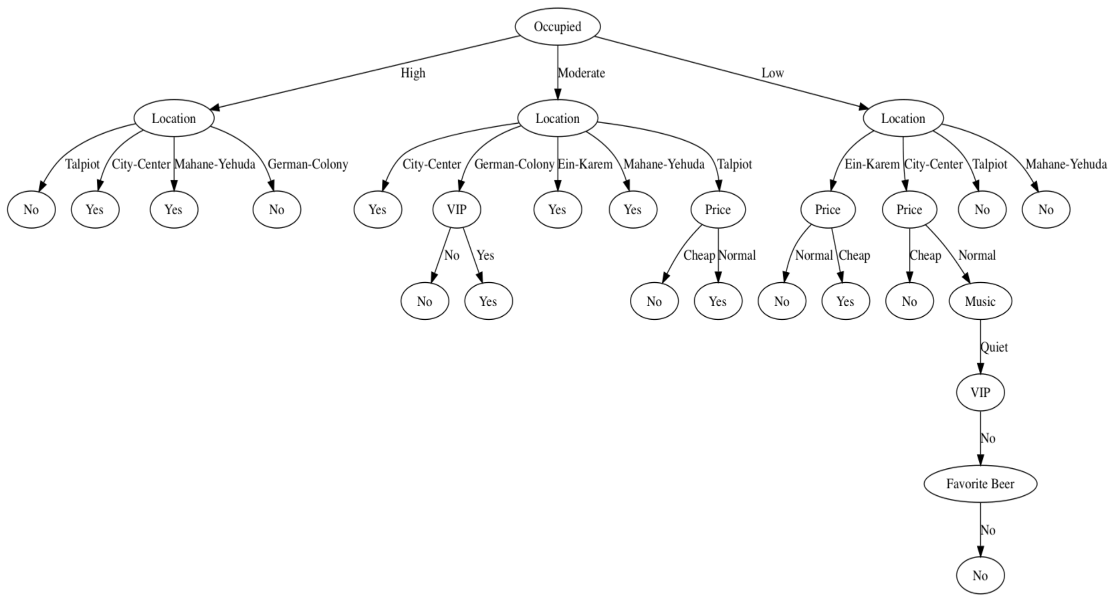
·Challenge:

1.Split nodes
The challenge is to sort the process of growing trees. Every time we choose the best feature, we need to split the data according to the values of the best feature. Therefore we get sub-sets of data and only focus on those data rows.

We also need to pay attention to the priority of the attribute. If there are two attributes which have the same information gain. We need to choose the prior one. This condition is easy to forget when writing the program.

2. A recursive algorithm to print the result decision tree
The biggest challenge, for me, is to develop a recursive algorithm. I spent most of the time thinking about how to let the tree grow.

·Print Format:



```
Occupied
  High
    Location
      Talpiot
        No
      City-Center
        Yes
      Mahane-Yehuda
        Yes
      German-Colony
        No
```

```
Moderate
  Location
    City-Center
      Yes
    German-Colony
      VIP
        No
          No
        Yes
          Yes
    Ein-Karem
      Yes
    Mahane-Yehuda
      Yes
    Talpiot
      Price
        Cheap
          No
        Normal
          Yes
Low
  Location
    Ein-Karem
      Price
        Normal
          No
        Cheap
          Yes
    City-Center
      Price
        Cheap
          No
        Normal
          Music
            Quiet
              VIP
                No
                  Favorite Beer
                    No
                      No
    Talpiot
      No
    Mahane-Yehuda
      No
```

Since the leaves nodes are “yes” or “no”. If we cannot decide the result based on given situation, the leaf node will be omitted. For example, in this case, (occupied=High, Location=Ein-Karem) cannot make a prediction, because the given training data does not include such situation.

·Optimization:

We changed our data structure after first attempt.

1. **From list to dictionaries for tree nodes storage**
   Use python dictionaries to store splitted trees instead of lists. Since each node of the tree is an attribute or the value of an attribute,we can look up and update the keys and values easily with dictionaries.We do not need to search for the value’s index first .In addition,it help us to finish graph visualization with dictionaries.

2. **Build an appropriate class**
   In the beginning, we only write some functions in our scripts. However, sometimes we need to add an new attribute to the tree and need to specify the certain subtree we are working on. In this case, we build a class to implement all these functions and tasks.

(b) Run your program on the data file.

(c) Make a prediction for (occupied = Moderate; price = Cheap; music = Loud; location = City-Center; VIP = No; favorite beer = No).

The prediction is “Yes”. The route of decision tree is :
Occupied=Moderate->Location=City-Center->Yes.

Part 2: Software Familiarization

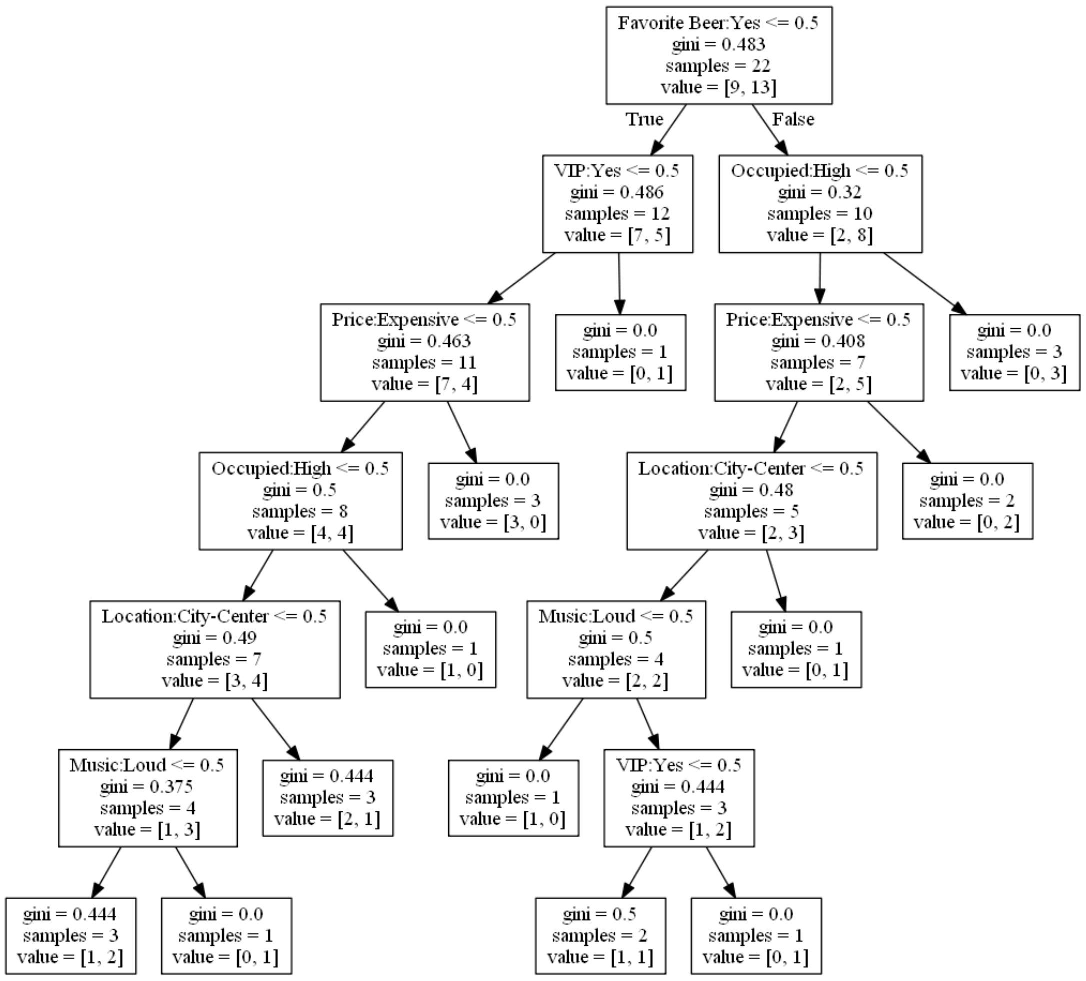
We find a python library sklearn, which uses Scikit-Learn Decision Tree Classifier.
A CART tree is a binary decision tree that is constructed by splitting a node into two child nodes repeatedly, beginning with the root node that contains the whole learning sample.

·Comparison:

	Python Scikit-Learn Decision Tree Classifier	Our Tree
Algorithm	optimized CART tree	ID3
Attribute	Categorical and numerical value	Categorical value

Advantages	The dot file can visualize easily; support continuous attributes	Easy to use and understand
Disadvantages	Need to assign and adjust different parameters	Cannot handle continuous attributes

·Tree of Python Scikit-Learn Decision Tree Classifier:


Favorite Beer:Yes <= 0.5 gini = 0.483 samples = 22 value = [9, 13]
True
False
VIP:Yes <= 0.5 gini = 0.486 samples = 12 value = [7, 5]
Occupied:High <= 0.5 gini = 0.32 samples = 10 value = [2, 8]
Price:Expensive <= 0.5 gini = 0.463 samples = 11 value = [7, 4]
gini = 0.0 samples = 1 value = [0, 1]
Price:Expensive <= 0.5 gini = 0.408 samples = 7 value = [2, 5]
gini = 0.0 samples = 3 value = [0, 3]
Occupied:High <= 0.5 gini = 0.5 samples = 8 value = [4, 4]
gini = 0.0 samples = 3 value = [3, 0]
Location:City-Center <= 0.5 gini = 0.48 samples = 5 value = [2, 3]
gini = 0.0 samples = 2 value = [0, 2]
Location:City-Center <= 0.5 gini = 0.49 samples = 7 value = [3, 4]
gini = 0.0 samples = 1 value = [1, 0]
Music:Loud <= 0.5 gini = 0.5 samples = 4 value = [2, 2]
gini = 0.0 samples = 1 value = [0, 1]
Music:Loud <= 0.5 gini = 0.375 samples = 4 value = [1, 3]
gini = 0.444 samples = 3 value = [2, 1]
gini = 0.0 samples = 1 value = [1, 0]
VIP:Yes <= 0.5 gini = 0.444 samples = 3 value = [1, 2]
gini = 0.444 samples = 3 value = [1, 2]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.5 samples = 2 value = [1, 1]
gini = 0.0 samples = 1 value = [0, 1]

·Execution code: sklearn File (attached in the zip)

·How to improve our code:

At first,we found that there are some tie situations which lead to null predict result.After we did it with sklearn, we knew that our result is correct. Sometimes it is null, we finally just output the result with non-null value as the sklearn do.I think we could manage the continued data or set default value for the missing data based on the major value in this attribute.

We can modify our code to support numerical values, just like the decision tree in the sklearn library. In addition, sklearn library pre-process the data to accelerate the data division. When the size of the dataset is very large, we can also roughly divide the data to help improve the speed. We can also standardize the raw data, to improve the accuracy of decision tree classification.

Part 3: Applications

In the field of Astronomy, decision trees are used for filtering noise from Hubble Space Telescope images, star-galaxy classification , determining galaxy counts and discovering quasars in the Second Palomar Sky Survey.
In the field of Finance, CART decision trees are used for asserting the attractiveness of buy-writes.
In the field of Biomedical Engineering, decision trees are used for identifying features to be used in implantable devices
Decision trees have also been used for building personal learning assistants and for classifying sleep signals.

Part 4: Group Members

Yidan Xue	9760308850 yidanxue@usc.edu	Tree visualization, Algorithm Optimization, Report editing
Wenjie Shi	8999907980 wenjiesh@usc.edu	Create tree algorithm, Software Comparison, Application

Part 5: Reference

[1]ftp://ftp.boulder.ibm.com/software/analytics/spss/support/Stats/Docs/Statistics/Algorithms/14.0/TREE-CART.pdf
[2]http://www.cbcb.umd.edu/~salzberg/docs/murthy_thesis/survey/node32.html