Programming assignment 1: Decision Tree
Group Report

Chenqi Liu 2082-6026-02
Che-Pai Kung 5999-9612-95
Mengyu Zhang 3364-2309-80

Part I: Implementation

The decision tree we built is univariate and binary and is split by the idea of Information gain (Entropy).

At first, we tried to build a decision tree that was n-ary based on how many unique values the attributes have. For example, since attribute “location” has five values, the node split by “location” would have five children nodes. However, it is complex to build a tree of nodes with many branches, leading to difficulties in getting the results after inputting test data. Besides, it is also against the rule of making simple decisions, therefore we decided to construct a binary decision tree instead.

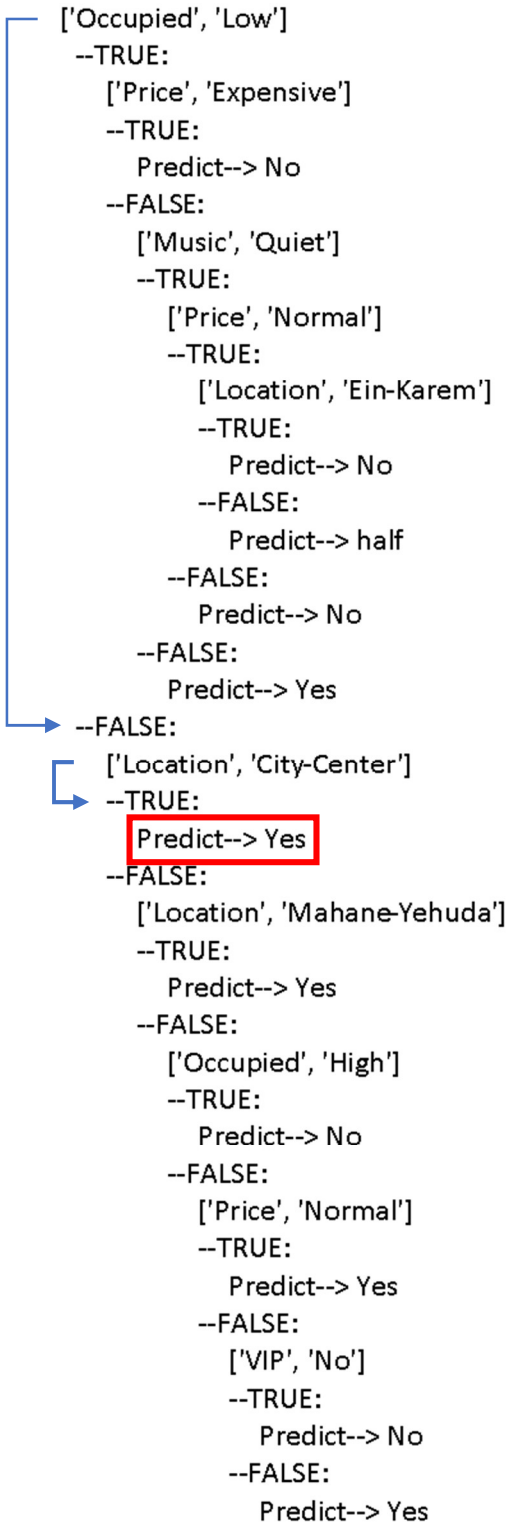
The data structure we use is the tree containing node value, pointer to ‘true’ child and pointer to ‘false’ child. Node value is a list including attribute and value used to filter. For instance, “['Occupied', 'Low']” means question “Is attribute ‘Occupied’ low?”. If yes, then ‘true’ child is returned, otherwise ‘false’ child is returned.

The tree is split based on the node attribute. At each node, we choose the attributes that can result in minimum entropy (maximum information gain) by looping through all the attributes. Since it is the binary tree, further looping through the unique values of each attribute(column) is needed. To make code more organized, the function “column_least_entropy” was written to calculate the lowest entropy of each column. Afterward, the train set is split into two subsets and the above process is repeated on each of them until the subset has the same value or no further split can be made.

Some optimizations were made to eliminate unnecessary calculations/loops. For example, once we get the zero entropy based on a certain split, then it should be returned as the best split of that node. Additionally, to increase the readability of the program, the structure was reorganized, some of the variables/functions were renamed and we also added some comments.

The biggest challenge was due to neglecting the stop point of the loop. In the beginning, the program worked well, but then the error occurred at the specific stage. After careful inspection, we finally found that it was caused by the different results but with the same attributes set (index 17,20). To deal with this issue, we decided to make the predictions based on the majority of the label if such a case happens. Nonetheless, if each label has equal counts, then a random choice of ‘Yes’ and ‘No’ will be made when predicting.

Finally, the decision tree is printed as follows:



The prediction for test data (occupied = Moderate; price = Cheap; music = Loud; location = City-Center; VIP = No; favorite beer = No) is “Yes” based on the above tree.

Part II: Software Familiarization

We choose scikit-learning (sklearn), a well-designed software machine learning library for the Python, as a comparison to the program we designed. In order to maintain the maximum similarity in the algorithm between the program, 'entropy' was chosen here as the criteria while building up the decision tree without pruning. We also assigned 20% of the given data as test data to check for accuracy when using the sklearn library.

Since it is not required to assign a certain percentage of the given data as test data, but we did do our own test for accuracy with 20% of the given data as the test data without randomness. As a result, we get around 60% accuracy from the decision we created and around 67% accuracy from the decision created by sklearn. Here are some possibilities we think might affect the difference between accuracy yield from different programs.

After looking into the decision resulted from the sklearn library, we found that the library used a different method when splitting the data into two subsets with attributes with more than two values. Take the attribute 'Occupied' as an example, it has three values 'Low', 'Moderate' and 'High'. Sklearn library labeled the values of the attributes with numbers, like 'Low'=0, 'Moderate'=1, and 'High'=2, while our program treated the distinct values uniquely. Thus, the minimized entropy calculated from the sklearn will result in the threshold value, like 1.5, to split the data sets into two subsets. Meanwhile, as we explained in Part I, we will ask "Is attribute 'Occupied' low?" instead. Therefore, when dealing with the attributes with more values, like "Location" with 5 values, the sklearn library will make a quicker prediction with higher accuracy because the entropy they calculated could give more information gain than the method we used.

As we were getting familiar with the sklearn library, we noticed that there are many criteria and options, like pruning, to create a decision tree. Since sklearn is a mature library for machine learning applications, their technique might be something we can borrow to apply for the future. Pruning and a combination of different criteria or algorithms will be a good improvement for the program. The criteria we are using for our program is just entropy. Entropy is a good criteria but not sufficient enough. If we can take consideration not only from entropy but also from other criteria like the Gini Index, attributes choice for the node may be improved. Also, pruning on decision trees could save time and calculations on unnecessary branches if the data have more attributes than we have right now.

Below is the output of our program:

Occupied_value	Occupied_index
High	0
Low	1
Moderate	2

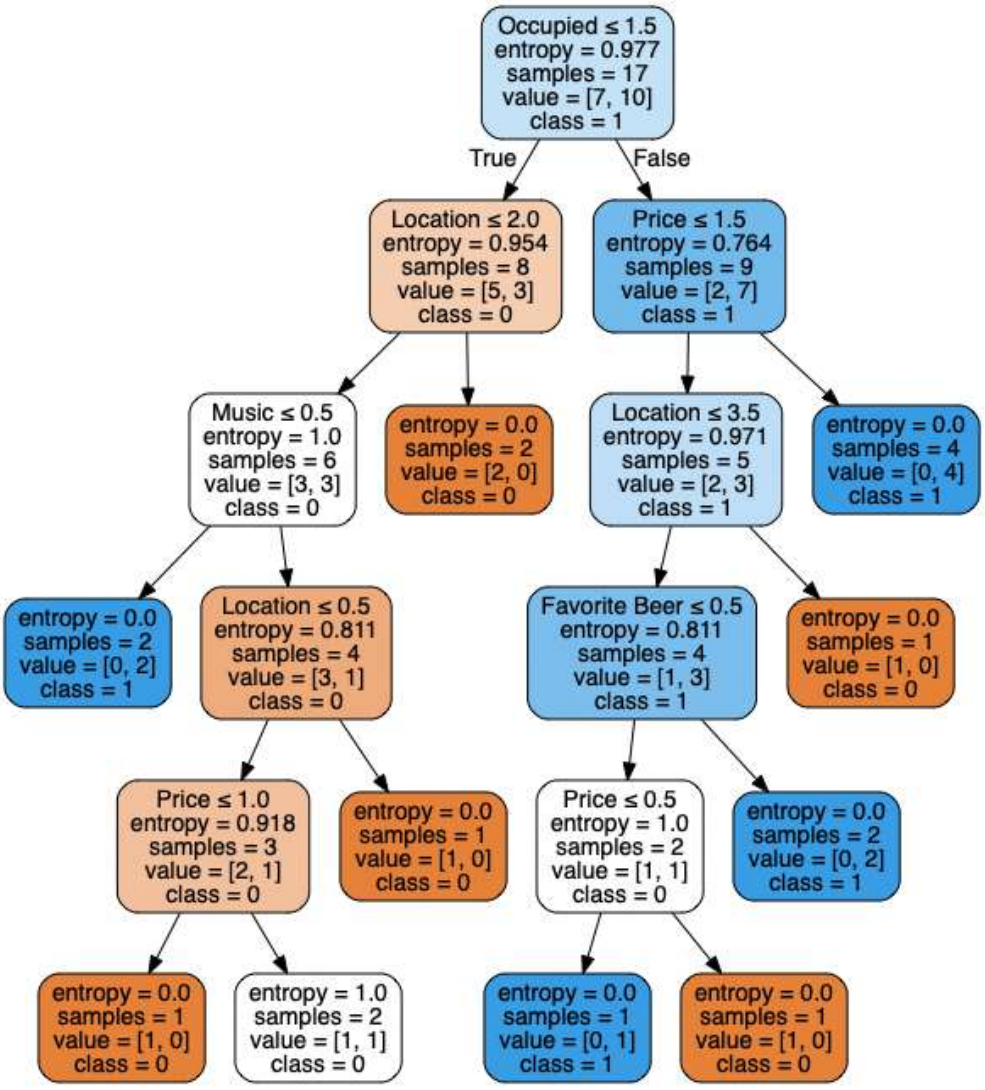
Price_value	Price_index
Cheap	0
Expensive	1
Normal	2

Music_value	Music_index
Loud	0
Quiet	1

Location_value	Location_index
City-Center	0
Ein-Karem	1
German-Colony	2
Mahane-Yehuda	3
Talpiot	4

VIP_value	VIP_index
No	0
Yes	1

Beer_value	Beer_index
No	0
Yes	1



Categories are separated by index. The value and index mapping are shown above as reference.

Part III: Applications

According to our research, decision trees are widely used in the industry, especially business management, engineering, and health management. Take business management as an example. Decision trees are a possible way to extract useful information from databases and they have already been employed in many applications in customer relationship management and fraud detection.

Investigating how individuals access online services is frequently used to manage customers' relationships. Such an investigation is mainly performed by collecting and analyzing individuals' usage data and then providing recommendations based on the extracted information. Man Wai Lee and his team applied decision trees in investigating the relationships between the customers' needs and preferences and the success of online shopping. In their study, the frequency of using online shopping is used as a label to classify users into two categories: (a) users who rarely used online shopping and (b) users who frequently used online shopping. The created decision trees suggest that the success of online shopping highly depends on the frequency of customers' purchases and the price of the products. Findings discovered by decision trees are useful for understanding their customers' needs and preferences.

The detection of Fraudulent Financial Statements (FFS) is a widely used business application. Such an application is particularly important because the existence of FFS may result in reducing the government's tax income. Creating a decision tree is a possible way to address this issue has been proved by previous research as it can consider all variables during the model development process. Kirkos have created a decision tree model to identify and detect FFS. In their study, 76 Greek manufacturing firms have been selected and their published financial statements, including balance sheets and income statements, have been collected for modeling purposes. The created tree model shows that all non-fraud cases and 92% of the fraud cases have been correctly classified. Such a finding indicates that decision trees can make a significant contribution to the detection of FFS due to a highly accurate rate.

Decision trees are useful tools in the real world industry and have been employed in various areas by people for decades. Studying it and understanding its principle is very helpful for students stepping into the industry in the future.

Part IV: Individual Contributions

- Model discussion: Chenqi Liu, Che-Pai Kung, Mengyu Zhang
- Model implementation: Che-Pai Kung
- Model optimization: Chenqi Liu, Mengyu Zhang
- Software Familiarization: Chenqi Liu, Mengyu Zhang
- Applications: Chenqi Liu, Mengyu Zhang