Programming Assignment 1: Decision Trees

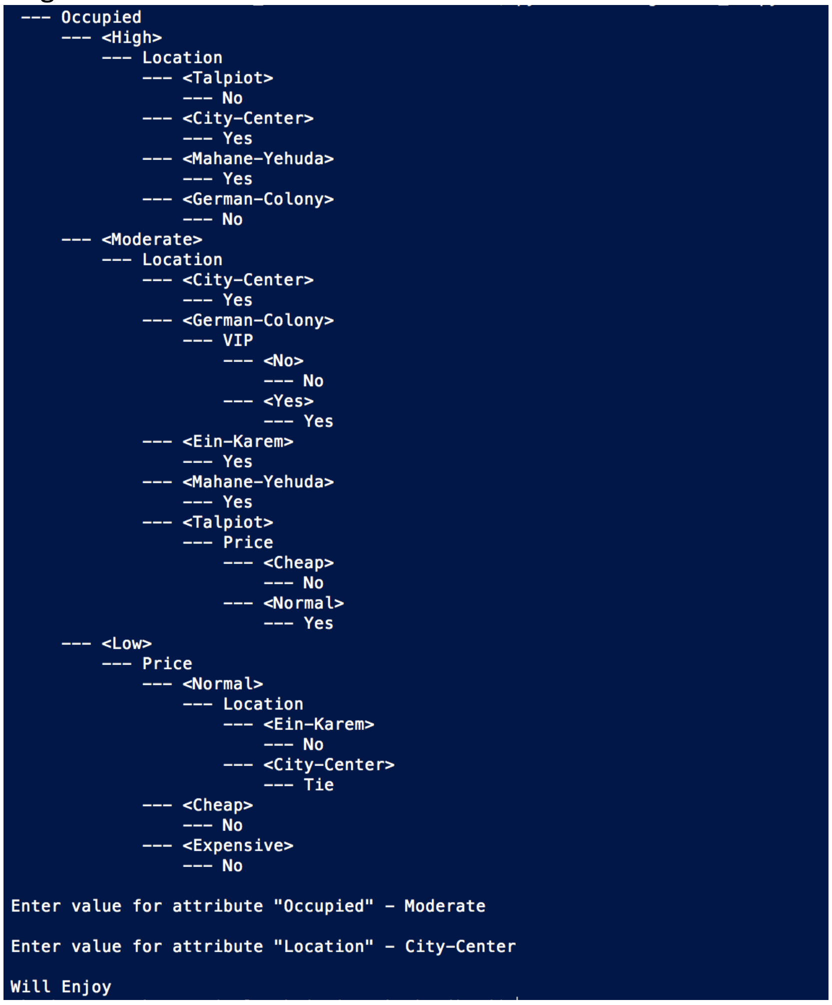
Srihari Akash Chinam
USC ID - 2953497706
Net ID - chinam

Sayan Nanda
USC ID - 2681592859
Net ID - snanda

Part 1: Implementation

- Language used – Python version 2
- Libraries used – Pandas
- Data structure used to save decision tree – Nested List

The structure of the decision tree is shown below in Figure 1, in a format different from the suggested format. Here, the indentation represents the level of the tree. Attributes and leaf node values are written as it is, while attribute values are written in angular brackets. The prediction for the input "occupied = Moderate; price = Cheap; music = Loud; location = City-Center; VIP = No; favorite beer = No" using this tree comes as "Yes" implying that the person will enjoy their night in Jerusalem.



```
 --- 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>
         --- Price
             --- <Normal>
                 --- Location
                     --- <Ein-Karem>
                         --- No
                     --- <City-Center>
                         --- Tie
             --- <Cheap>
                 --- No
             --- <Expensive>
                 --- No

Enter value for attribute "Occupied" - Moderate

Enter value for attribute "Location" - City-Center

Will Enjoy
```

Figure 1 Decision Tree generated from dataset

Individual contributions for the assignment from Srihari Akash Chinam:

- Implementation
    - Functions: Tree print, Predictor, Resolver, Decision tree
- Software Familiarization
    - Implementation of Decision Tree from SciKit-Learn
    - Comparison of outputs from SciKit-Learn and designed code
- Applications
    - Streaming Decision Trees

Individual contributions for the assignment from Sayan Nanda:

- Implementation
    - Preprocessing of Data
    - Functions: Entropy of Set, Information Gain, Decision Tree
- Software Familiarization
    - Extraction of Decision Tree structure built by SciKit-Learn
    - Comparison of outputs from SciKit-Learn and designed code
- Applications
    - Medical and astronomical applications

Part 2: Software Familiarization

The python library scikit learn offers an implementation of the decision tree. For familiarization with this software, we implemented a decision tree based on the same data set.

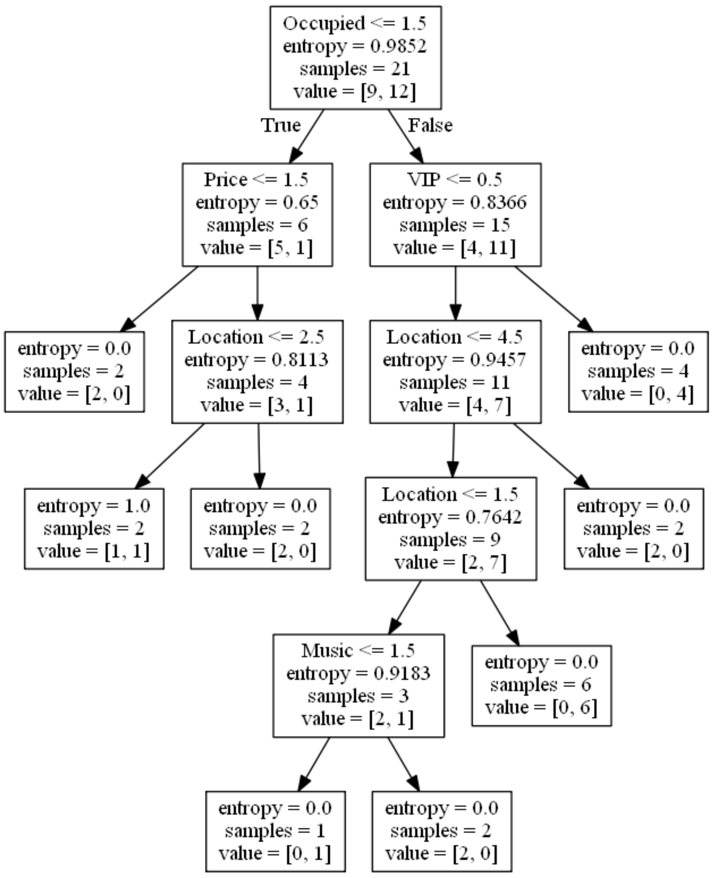


Figure 2 Decision Tree using scikit learn

Scikit implements a decision tree similar to CART (Classification and Regression Trees), which is very similar to C4.5. It supports numerical target attributes and it does not compute rule sets. Scikit implements an optimized version of the same.

The decision tree implementation of scikit learn uses a binary tree. It only accepts numerical input. For implementing the given data set, we had to map the value to integer values. For example, the attribute Occupied had three possible values, high, medium and low which we mapped to 1, 2 and 3.

Scikit learn uses a binary decision tree because for larger data set there are too many possible splits. An attribute that has 1000 distinct values will have 999 possible binary splits. On the other hand, if trinary splits are accepted, there are 999x998 possible splits. Hence the implementation restricts the decision tree to binary splits.

Another reason Scikit learn only uses binary splits is because it only uses numerical features. With numerical features it is hard to decide where to split the feature in order to produce more than two children. Categorical features are not accepted because if there are too many values, the tree becomes difficult to manage.

To improve our algorithm, we can add a tree postprocessing step. For example, a pruning step could help improve the performance by decreasing the complexity of the final classifier. Looking at the Scikit learn implementation, we can build our multiway tree to be a binary tree. Another possible improvement is building our tree such that it is balanced.

Part 3: Applications

Decision trees are used in a multitude of medical applications. Recent uses include diagnosis, cardiology, psychiatry, gastroenterology, diagnosing thyroid disorders. Some other applications of decision trees are detection of physical particles, medical text classification, control of nonlinear dynamical systems and drug analysis. Another interesting application of decision trees is in astronomy. They have been used to filter noise from Hubble Space telescopes, determine galaxy counts, discovering quasars and they have also helped in star-galaxy classification.

Decision trees are the base structures for many complex machine learning algorithms, such as random forests, where a group of decision trees provide a solution to a classification problem, using majority voting. Another such example recently published upon is Streaming Decision trees which are decision trees that adapt to streaming data than a normal decision tree that is built on a snapshot of data.