Programming Assignment 1: Decision Trees

Project Report by Arpit Parwal & Yeon-soo Park

1. Implementation of ID3 algorithm

a. Data structures

i. **Tree**: Make a class called Decision Node that has a question and a list of children (True and False Branch). Each child is an instance of the same class.
* *Class Decision_Node: { self.decision, self.true_branch, self.false_branch}*

ii. **List**: Get the attributes and contents from the file and save it as a separate structure. The Contents variable contains all the training sets stored as a nested array.
* *attributes = List(), contents = List()*

iii. **Dictionary**: create a dictionary with labels (yes, or no) and counting them when we recursively count the number of each type of example.
** counts = { key=label, value = count }*

b. Explanation of Modules

1) LoadData: Getting data from the file

- **Step 1)** `dataset = open("dt_data.txt")`

  We declared the variable dataset to open a file named dt_data.txt. Open takes 1 argument, the file that we want to open

- **Step 2)** `line = dataset.readline()`

  Read a file line by line

2) Calculation of Entropy and Information Gain

- **Step 1)** Measure disorder, Mathematical formula for entropy:

  The mathematical formula for entropy = $E(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$

  Pi is the frequentist probability of an element/class 'i' in our data'.

  The value of Entropy is a measure of disorder

- **Step 2)** Measure the reduction of disorder given our additional info.

  Information gain from x to y = IG(Y, X) = E(Y) - E(Y|X).

  The greater the reduction means we can gain more information about Y from X

3) Storing the result and building the tree recursively

- **Step 1)** Find the best split

  Find the best split by iterating over every features/value and calculate information gain

- **Step 2)** Build the tree

  From the base case with no information gain, we store the best features based on information gain.

- **Step 3)** Terminate the recursive call

  If we can't get any further information, terminate the tree process.

4) Printing the tree & classification

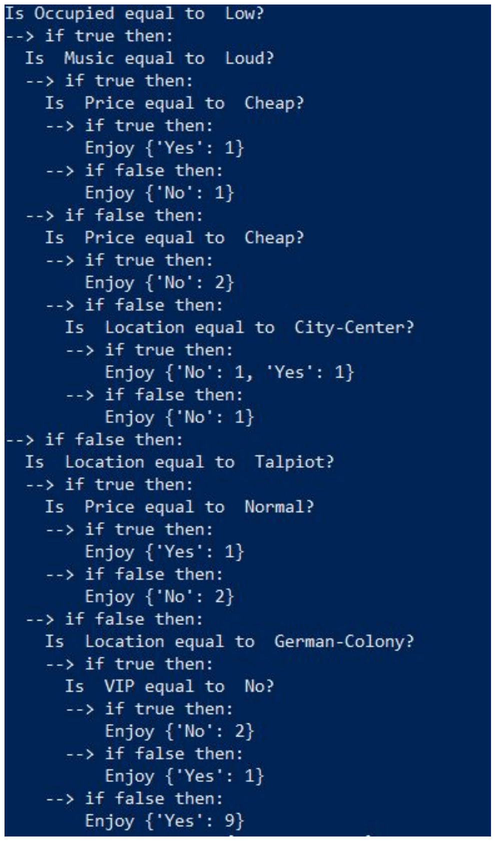
- **Step 1)** Print the tree

  Print all of the nodes using recursion.

- **Step 2)** Classification

  Evaluation of the performance of the decision tree model based on the test dataset

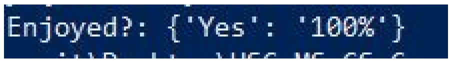
c. Output & Predictions



```
Is Occupied equal to  Low?
--> if true then:
  Is  Music equal to  Loud?
  --> if true then:
    Is  Price equal to  Cheap?
    --> if true then:
        Enjoy {'Yes': 1}
    --> if false then:
        Enjoy {'No': 1}
  --> if false then:
    Is  Price equal to  Cheap?
    --> if true then:
        Enjoy {'No': 2}
    --> if false then:
      Is  Location equal to  City-Center?
      --> if true then:
          Enjoy {'No': 1, 'Yes': 1}
      --> if false then:
          Enjoy {'No': 1}
--> if false then:
  Is  Location equal to  Talpiot?
  --> if true then:
    Is  Price equal to  Normal?
    --> if true then:
        Enjoy {'Yes': 1}
    --> if false then:
        Enjoy {'No': 2}
  --> if false then:
    Is  Location equal to  German-Colony?
    --> if true then:
      Is  VIP equal to  No?
      --> if true then:
          Enjoy {'No': 2}
      --> if false then:
          Enjoy {'Yes': 1}
    --> if false then:
        Enjoy {'Yes': 9}
```

Figure: Output Decision Tree

Description: In the figure above each question represents a node. Each node can have two branches either 'true' or 'false'. Once a leaf node is reached it outputs NO or YES with the label tag. It also has the number of cases or training set for which the output is reached. This helps us with the probability of the said outcome. *For example, there is one such leaf where the program predicts the output with a 50% probability. This is majorly due to insufficient data so as to predict any answer with 100 percent surety.*

For the given data, our program predicts 

```
Enjoyed?: {'Yes': '100%'}
```

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

d. Challenges

One of the major challenges was to decide a data structure for storing our tree. Storing it using class objects was very helpful and intuitive as it helped to reduce the complexity of managing another data structure. It even made the code very efficient as it includes a lot of recursions.

Another minor challenge was to create an efficient way to print the tree. After doing some research we were able to come up with a very clean and user-friendly tree

2. Software Familiarization

Existing Libraries:

Scikit-Learn (for Python) sklearn.tree.DecisionTreeClassifier

It's a standard and famous library which includes various machine learning algorithms. We are using its train_test_split, DecisionTreeClassifier, accuracy_score algorithms. This Library uses the
C4.5 algorithm to Implement Decision tree in PythonDecisionTreeClassifier is a class capable of performing multi-class classification on a dataset. As with other classifiers, DecisionTreeClassifier takes as input two arrays: an array X, sparse or dense, of size [n_samples, n_features] holding the training samples, and an array Y of integer values, size [n_samples], holding the class labels for the training samples

Data. tree (for R)

Data.tree is the most used library to implement the ID3 Algorithm in R. This package provides methods for printing and plotting trees. It supports conversion from and to data.frames, lists, and other tree structures from the ape package, igraph, and other packages. These structures are bi-directional and ordered. It helps to navigate from parents to children and vice versa. Ordered means that the sort order of the children of a parent node is well-defined

Code-level optimizations

In order to optimize our code, we can do some optimization which can help us to generate faster results using list comprehensions over explicit for loops in certain places, making code more compact and execution faster. We also modularized parts of code into methods, making code more readable.

Furthermore, we used a set data structure instead of a list to maintain remaining attributes, Set in python is inherently faster than list operations.

We used the class objects in order to efficiently use the inheritance property of object-oriented programming. This helped us to efficiently manage large trees and instances at a time. However, a better implementation can be done by using python libraries such as NumPy which provides an easy way to do efficient computation on array and matrices.
Another such library will be Pandas which provides data frame Object for data manipulation. Furthermore, these data frames can hold different types of data of the multidimensional array

3. Applications

a. Travel and Airline booking

Suppose the Decision tree has a huge application for any flight booking and travel portal. We check first if the flight is available on that day or not. All the user filters and preferences can be filtered using the decision tree. We can also understand by using historic data what worked for most of the users and what did not.
For example, the attributes can be duration, direct flights, price, etc. and label can be if the user booked the flight or not.

b. University admission decision

Student Admission is usually done by comparing the candidate application file, so the subjectivity of assessment is most likely to happen because of the lack of standard criteria that can differentiate the quality of students from one another. By applying the decision tree technique for classification, we can build a model selection for new students which includes criteria to certain standards such as the area of origin, the status of the school, the average value and so on.
The results show that students are given priority for admission is that meet the following criteria: come from a specific geographic area, public school, majoring in science, an average value above 75, and have at least one achievement during their study in high school

c. Healthcare and sensitivity analysis

Decision tree analysis in healthcare can be applied when choices or outcomes of treatment are uncertain, and when such choices and outcomes are significant (wellness, sickness, or death). The idea of assigning values to states of health may range from a score of 1 for perfect health, 0 for death, and somewhere in between for sickness to a number. This approach allows physicians to better identify the most favorable option for patients. With the range of modeling techniques available, it can also yield valuable information and risk to patients might be reduced.

d. Online or app-based “chatbots”

Decision trees are how Chatbots help customers find exactly what they're looking for. They map out a step-by-step process to discover the precise answer to the patients’ questions in a conversational format. We can evaluate symptoms 24/7 and make healthcare more accessible and effective, which means that help patients to deal with emergency situations where people don’t understand the significance of some symptoms.

e. User behavior prediction in the online consumer market

We can collect user behavior data from weblogs and predict the purchasing behavior of customers. There are few attributes to build a decision tree: descMatch, titleMatch, popularityWeek, day since release and so on. The Model can predict shopping intention and classify respondents into one of the two categories, depending on whether they intend to shop or not.

4. Individual Contribution

Student Name	**Programming Part**	**Documentation**
Arpit Parwal **aparwal@usc.edu**	- Evaluate the decision tree - Calculate the entropy and Information Gain - Print the tree	- Application of Decision Tree - Software Familiarization Application
Yeon-soo Park **yeonsoop@usc.edu**	- Load data from the file - Store the result and implement the structure for all the nodes	- Data Structures - Explanation of Modules