Machine Learning Assignment 1: Decision Tree

Priyambada Jain (priyambj@usc.edu)
Sai Sree Kamineni (skaminen@usc.edu)
Varad Kulkarni (vdkulkar@usc.edu)

Part 1: Implementation Description

Data Structures Used: numpy array, pandas dataframe, dictionary, list.

Implementation Steps:
Step 1: Converted the training data into pandas data frame
Step 2: Separated the target ['Enjoy'] attribute from the train data
Step 3: Construction of Decision tree using **initialEntropy** function (used to calculate the distribution of the classes) and **informationGain** function (used to evaluate the attribute selection method) recursively.
Step 4: Module **predict**, predicts the test case using the decision tree model designed.

Modules Explained:

1) initialEntropy (attribute):

Input: array with the attribute value.
Output: Randomness of the attribute / distribution of the classes of attribute
Process: Calculate the unique classes and frequency of each class, followed by the probability of each class. Now using the formula,

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

2) informationGain (target, attribute):

Input: array of target variable, array of attribute on which we want to split the train data.
Output: Information Gain achieved by splitting the dataset across the specified attribute.
Process: Calculate entropy of the target. The dataset is then split on the attribute passed and calculate the entropy for each branch of the split. Now add it proportionally to get total entropy for the split. The resulted entropy is subtracted from the entropy before the split. The result is IG, Information Gain or decrease in entropy(randomness).

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

3) partition (a):

Input: array of an attribute value
Output: Returns indices of rows for each attribute value.
Process: Splits the entire data into groups based on attribute values.

4) recursive_split (x, y):

Input: array of attribute on which we want to split the train data, array of target variable.
Output: Returns a decision Tree
Process: It's a recursive function. It initially checks if the classified data is a pure class. It is the base condition. Then calculate information gain for each attribute and pick the attribute with highest information gain. If all the IGs are below a base level (0), it returns a tie. If not, it calls partition and splits the data and calls recursive_split on these partitioned data.

5) predict(tree, lis):

Input: Takes the decision tree returned by recursive_split function & the prediction string as a list.
Output: Prints the prediction.
Process: It's a recursive function which continuously checks the keys in the decision tree which is a dictionary and matches with the prediction variables given. The base condition is reached when leaf node is reached.

Command to run the script -

python MLDT.py 'Enjoy' '(occupied = Moderate; price = Cheap; music = Loud; location = City-Center; VIP = No; favorite beer = No)'

python *filename* '*Prediction Attribute*' '(*prediction String)*'
Prediction String format – attribute = value separated by semi colon

Output Format –

The decision tree & Prediction in the next line.

Example –

```
{'Occupied = High': {'Location = City-Center': 'Yes',
                     'Location = German-Colony': 'No',
                     'Location = Mahane-Yehuda': 'Yes',
                     'Location = Talpiot': 'No'},
 'Occupied = Low': {'Price = Cheap': 'No',
                    'Price = Expensive': 'No',
                    'Price = Normal': {'Location = City-Center': 'Tie',
                                       'Location = Ein-Karem': 'No'}},
 'Occupied = Moderate': {'Location = City-Center': 'Yes',
                         'Location = Ein-Karem': 'Yes',
                         'Location = German-Colony': {'VIP = No': 'No',
                                                      'VIP = Yes': 'Yes'},
```

```
'Location = Mahane-Yehuda': 'Yes',
'Location = Talpiot': {'Price = Cheap': 'No',
                       'Price = Normal': 'Yes'}}}
```

Prediction: Yes

Challenges Faced:

1) The first challenge was to build a modular structure for the program. Coming up with the most appropriate and optimal modular structure was a big challenge in itself.
2) Deciding on the data structures to be used in the program was a challenge as time and space complexity were of utmost importance.

Part 2: Software Familiarization:

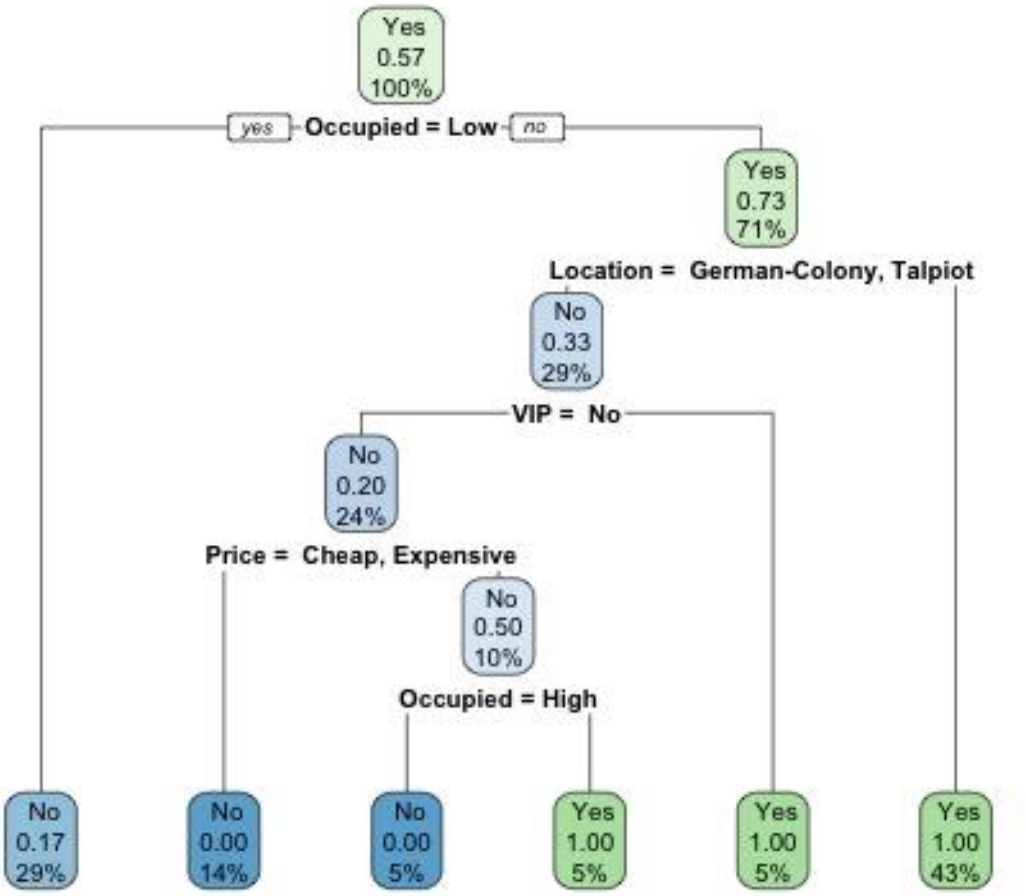
We researched to find out good libraries that were providing the Decision tree implementation in R. Here are the findings:

rpart : The R package **rpart** implements recursive partitioning

Syntax : rpart(formula, data, weights, subset, na.action = na.rpart, method, model = FALSE, x = FALSE, y = TRUE, parms, control, cost, ...)

```
DT1 <-  rpart(formula=formula,data=data,method='class',control=rpart.control
(minsplit=1, minbucket=1, cp=0.001))
rpart.plot(DT1)
print(DT1)
node), split, n, loss, yval, (yprob)
     * denotes terminal node

 1) root 21 9  Yes (0.4285714 0.5714286)
   2) Occupied=Low 6 1  No (0.8333333 0.1666667) *
   3) Occupied=High,Moderate 15 4  Yes (0.2666667 0.7333333)
     6) Location= German-Colony, Talpiot 6 2  No (0.6666667 0.3333333)
      12) VIP= No 5 1  No (0.8000000 0.2000000)
        24) Price= Cheap, Expensive 3 0  No (1.0000000 0.0000000) *
        25) Price= Normal 2 1  No (0.5000000 0.5000000)
          50) Occupied=High 1 0  No (1.0000000 0.0000000) *
          51) Occupied=Moderate 1 0  Yes (0.0000000 1.0000000) *
      13) VIP= Yes 1 0  Yes (0.0000000 1.0000000) *
     7) Location= City-Center, Ein-Karem, Mahane-Yehuda 9 0  Yes (0.0000000 1.0000000)
*
```



---

where formula <- Enjoy ~ Occupied + Price + Music + Location + VIP + Favorite.Beer
data <- CSV having all 21 instances of dt-data.txt

Also tried looking into following decision tree libraries included in R:
randomForest,J48,DecisionStump,train,C5.0 they differ on split criteria and the construction algorithm.
We chose rpart since it followed recursive partitioning which is much similar to the algorithm we have implemented using python.

Part3: Decision Tree Applications:

1) **Business Management**: Decision trees are a possible way to extract useful information from databases and they have already been employed in many applications in the domain of business and management.
2) **Customer Relationship Management: A frequently used approach to manage customers'** relationships is to investigate how individuals access online services. Such an

investigation is mainly performed by collecting and analyzing individuals' usage data and then providing recommendations based on the extracted information.

3) **Energy Consumption**: Energy consumption concerns how much electricity has been used by individuals. Decision trees are preferred due to the fact that a hierarchical structure provided by decision trees is useful to present the deep level of information and insight.
4) **Healthcare Management**: As decision tree modelling can be used for making predictions. The classification rate indicated by the decision tree is highly accurate for predicting the survivability of breast cancer patient.
5) **Educational Purpose:** Application of Decision Tree Approach to Student Selection Model. By applying data mining techniques classification, can be built 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. These criteria are determined by using rules that appear based on the classification of the academic achievement (GPA) of the students in previous years who entered the university. through the same way.