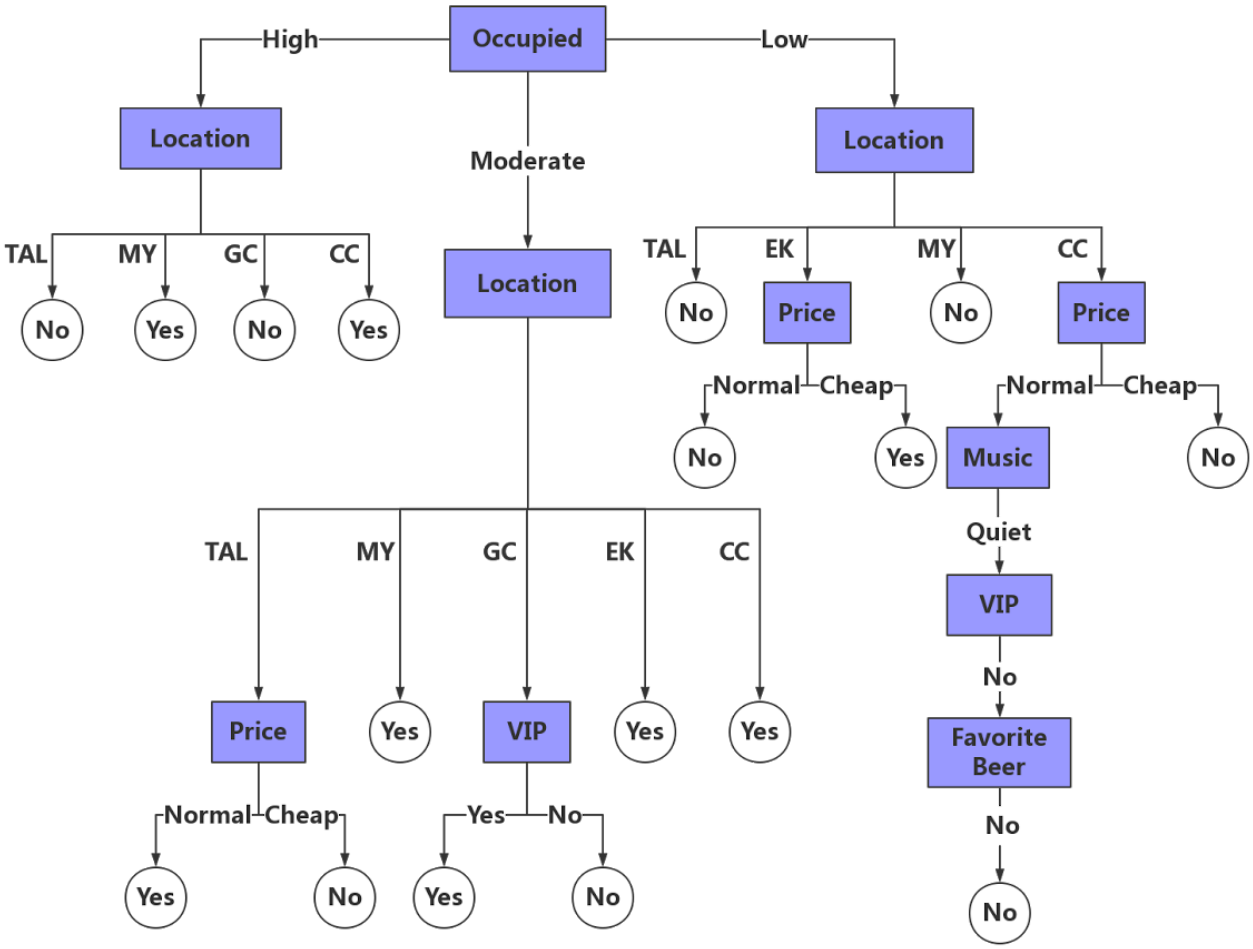
# INF552 Homework 1

Group members and contribution :
Krishna Akhi Maddalil (Code development)
Yash Shahapurkar (Code development and report)
Myungjin Lee (Analysis and research)

## Part 1: Implementation

### The Constructed Decision Tree



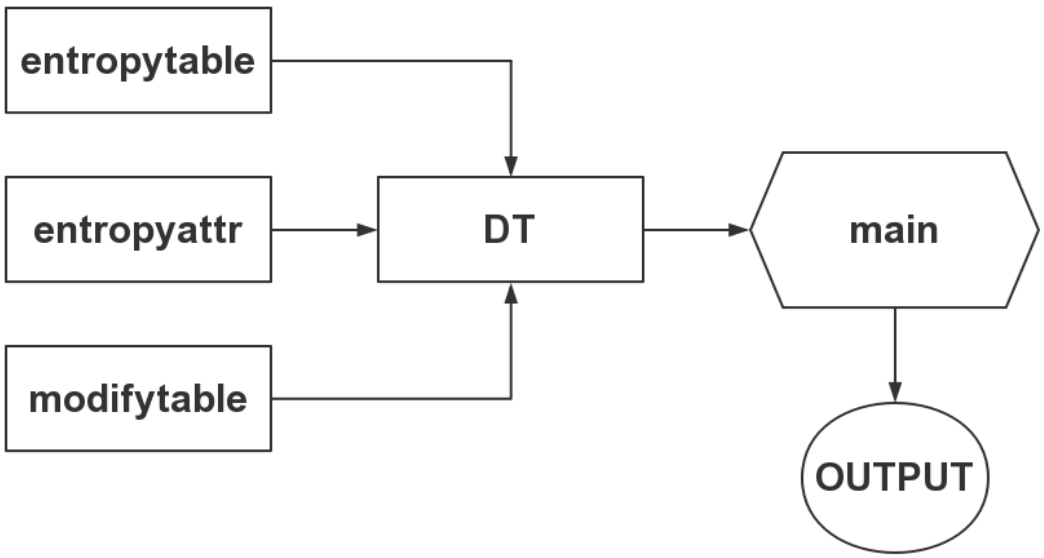
- Output after running the program

This code is designed to print the output from the root node to the leaf node. The indentation give us information for each level of the node. All nodes enclosed by square brackets(‘[ ]’) are classifications. After each node say Occupied the value of that attribute follows next(i.e. Edge values are followed after the nodes).The curly brackets (‘{ }’) are used to segregate nodes into different levels in the tree.
General Format
{Attribute A on first level:{Value 1 of Attribute A:Attribute B1/Classification on level 2,Value 2 of Attribute A:Attribute B2/Classification on level 2}} and so on.
If Attribute B1 exists(node is not a classification node) it is followed by the same structure as above until classification is reached.

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

```
                    'Normal':
                        {'Music':
                            {'Quiet':
                                {'VIP':
                                    {'No':
                                        {'Favorite Beer':
                                            {'No':
                                            ['No']}}}}}}},
        'Ein-Karem':
            {'Price':
                {'Cheap': ['Yes'],
                'Normal': ['No']}},
    'Mahane-Yehuda': ['No']}}}}
```

- Data structure




- DT: Class Table for storing each instance of a table
- Defined constructor
- Used dictionaries to convert strings into variable names
- Used lists for each attribute to keep track of all possible values for that attribute
- Represented table as a list of list containing strings

## Code-level optimization

- Used recursion for printing output
- Use of dictionaries for lookup strings into variable names – provides secure way of conversion over commands such as exec
- Reduced indexing in loop statements

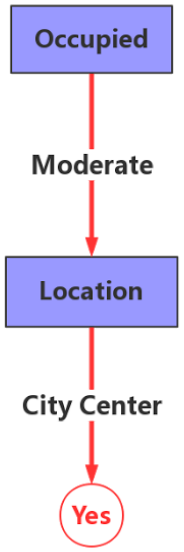
## Challenges

- Determination where to terminate node splitting.
- Math errors arising due to cases where count either Yes or No or both was zero
- Tracking of each branch
- Case of contradiction between two instances – False positives are not allowed so such cases are always classified as No.

## Prediction

Prediction for (occupied=Moderate; price=Cheap; music=Loud; location=City-Center; VIP=No; favorite beer=No)
**enjoy="yes"**

After traversing through the decision tree, you will have a good night-out in Jerusalem for the coming New Year's Eve.



# Part 2: Software Familiarization

In our implementation, the ID3 algorithm has been used for constructing decision tree among several ways such as ID3, C4.5, C5.0, CART, CHAID based on the splitting measure. Here, we have compared ID3 algorithm with CART to comprehend the difference between these two as well as how to improve our algorithm better. The CART algorithm from Python scikit library is chosen for comparing the difference with ours.

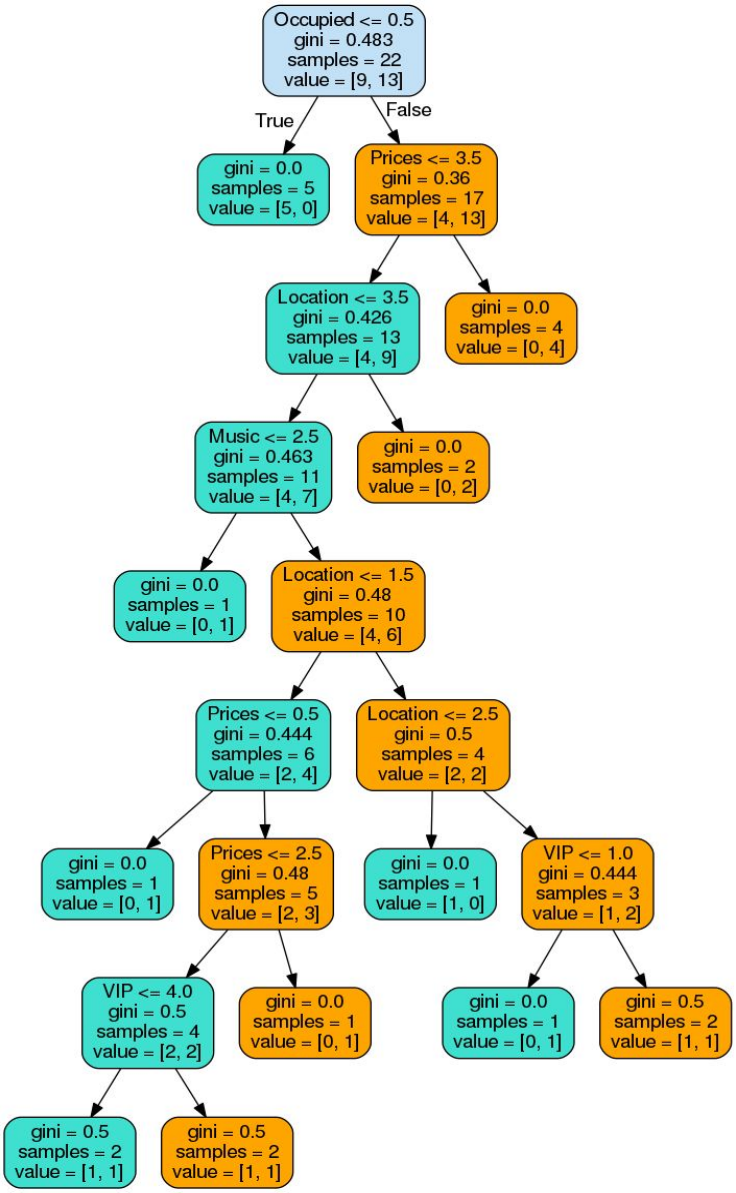
| | **ID3** (Iterative Dichotomiser 3) (Quinlan, 1983) | **CART** (Classification and Regression Trees) (Breiman et al., 1984) |
|---|---|---|
| Features | ● Builds the fastest/short trees.<br>● Very efficient<br>● Possibility for data to be over-fitted/over-classified if a test set is not enough<br>● Does not handle numeric attributes.<br>● Only one attribute at a time can be handled for making decision<br>● Does not handle missing values | ● Constructs binary trees.<br>● Generates regression trees. (it can handle real number leaves and not a class)<br>● Handles missing values<br>● Cost-Complexity pruning is used |
| Splitting criteria | Information gain | Binary tree : twoing criteria<br>Regression tree : minimize the precision squared error |
| Library | Our own code | Python Scikit (sklearn) |

- ## Output

1. ID3 (our code)
   The output of our code is shown in part 1.

2. CART (Python sklearn) - Decision tree by CART

Prediction: “Yes”



- How to improve our code

The chosen ID3 algorithm in this implementation determines the classification of the attributes by calculating entropy (Entropy (t) = -∑ p (i/t) log 2 p (i/t)) difference between the node before splitting and the after splitting. At each node, only one attribute at a time is tested and the classification of the attributes leans to maximizing information gain.

However, even if ID3 algorithm is one of the efficient way to construct the decision tree, there is a chance to over-fit the data if only a small sample is tested. The pruning which CART algorithm takes is a way to reduce overfitting, where branches that make use of features of low importance are removed. If we are getting very less information as we go down the tree, we remove all that part of the tree. Then, replace the node with the most popular class in that region. If we take pruning, the developed code here might reduce the possibility of overfitting.

In addition, since ID3 does not handle the missing values, it could mark for the missing values but of course, the missing values are not going to be used as part of calculation for entropy gain.

Also, numerical values which are continuous can not be handled by ID3 algorithm. In order to deal with the continuous attributes, this algorithm might need to create a threshold and then splits the list into those whose attribute value is above the threshold and those that are less than or equal to it.

# Part 3: Applications

The advantages of decision trees include simplicity, interpretability, and fast implementation. Hence, decision trees have applications in many diverse fields like business and e-commerce, health and wellness, energy systems etc.

We highlight some of the applications and discuss its implementation and usage.

1 - Business and e-Commerce

Online marketing is on a high in the last decade. People tend to buy more products online due to convenience, low price and abundance. Moreover, online agencies offer deals for products, thereby increasing marketing and business. Recommendation systems have been largely adapted by agencies, which allow customers to get exactly what they look for. To build a recommendation system, we assume we have a record of the past buying habits of a customer. Attributes which could be taken into account are: Purchase amount, category of product, preference, season, offers on the products etc. Based on the record, we should predict a product that he is more likely to buy and display the same.

2 - Health care

The application of decision tree on health care is also interesting. The principle of decision tree is to start with a node with several possible choices. Each choice leads to several numbers of outcomes, and each of these outcomes has a value as well as a probability of occurring. In decision tree analysis in healthcare, the value is often represented as a expected ‘life spans’ or ‘quality-adjusted life years’ for the patient. The same can also be used to identify a particular kind of disease based on symptoms observed. For a given choice, the outcomes are mutually exclusive and exhaustive.

3 - Energy systems

There is massive increase in power usage because of increased industrialization. It required for power systems to be utilized effectively. Based on certain attributes in a given area. We can predict how much power is required. Accordingly, power stations can be set up. The attributes could be : population of area, residential or commercial

zone, number of heavy industries, geographical terrain of area etc. Based on these, we could either predict a certain amount ( numerical ) of power required, or predict if a power station is required or not ( categorical )

## CONCLUSION

Having covered all the above mentioned topics, we conclude that decision trees are an easy and interpretable way to perform classification or regression problems. Given the disadvantage of overfitting, due to less data, we suggested some improvement to the performance. We implemented our own version of decision tree and a library supported version. Finally, we provided detailed applications of decision trees for real-world cases.