INF552 Machine Learning for Data Informatics HW1

1. Contribution:

a. Shih-Yu Lai (50%):

Email: shihyula@usc.edu

USC-ID: 7183984563

Predict and draw the tree, debug the code and write the ReadMe.

b. Shiuan-Chin Huang (50%):

Email: shiuanch@usc.edu

USC-ID: 9815633745

Calculate Entropy, build a decision tree and write the ReadMe.

2. Data Structure

We create a class called **TreeNode.** And in **TreeNode**, we have *attribute*, *attrCounter*, *children*, *label*, and *childrenTreeNodes*.

attribute: e.g. Occupied

attrCounter: e.g. High : {Yes : 3, No : 2}

children: has children nodes or not e.g. High --> 3 Yes,

self.children[High] = True

childrenTreeNodes: e.g. occupied -> High : VIP

3. Our Prediction

We use dt_data.txt as our train data.

The test case we should predict is:

Occupied = Moderate; Price = Cheap; Music = Loud; Location = City-Center; VIP = No; Favorite beer = No

And the predict answer of this test case's Enjoy is:

Yes

4. Challenge

At first, we found that if we use all the data in dt_data.txt, then our code will be stuck in the infinite loop. After we debug, then we found that is because the 18th and 21st data in dt_data.txt are inconsistent, they are all same but have different Enjoy outcomes (18th is No while 21st is Yes).

Finally, we decide to delete one of them and our code can run smoothly.

5. Draw Tree

We will print our path through the tree.

For example, if we want to predict this test case :

Occupied = Moderate; Price = Cheap; Music = Loud; Location = City-Center; VIP = No; Favorite beer = No

Then, we will print

Occupied (Moderate) ->

Location (City-Center) ->

Enjoy : Yes

Another example :

Occupied = Low; Price = Cheap; Music = Loud; Location = Ein-Karem; VIP = Yes; Favorite beer = Yes

Then, we will print

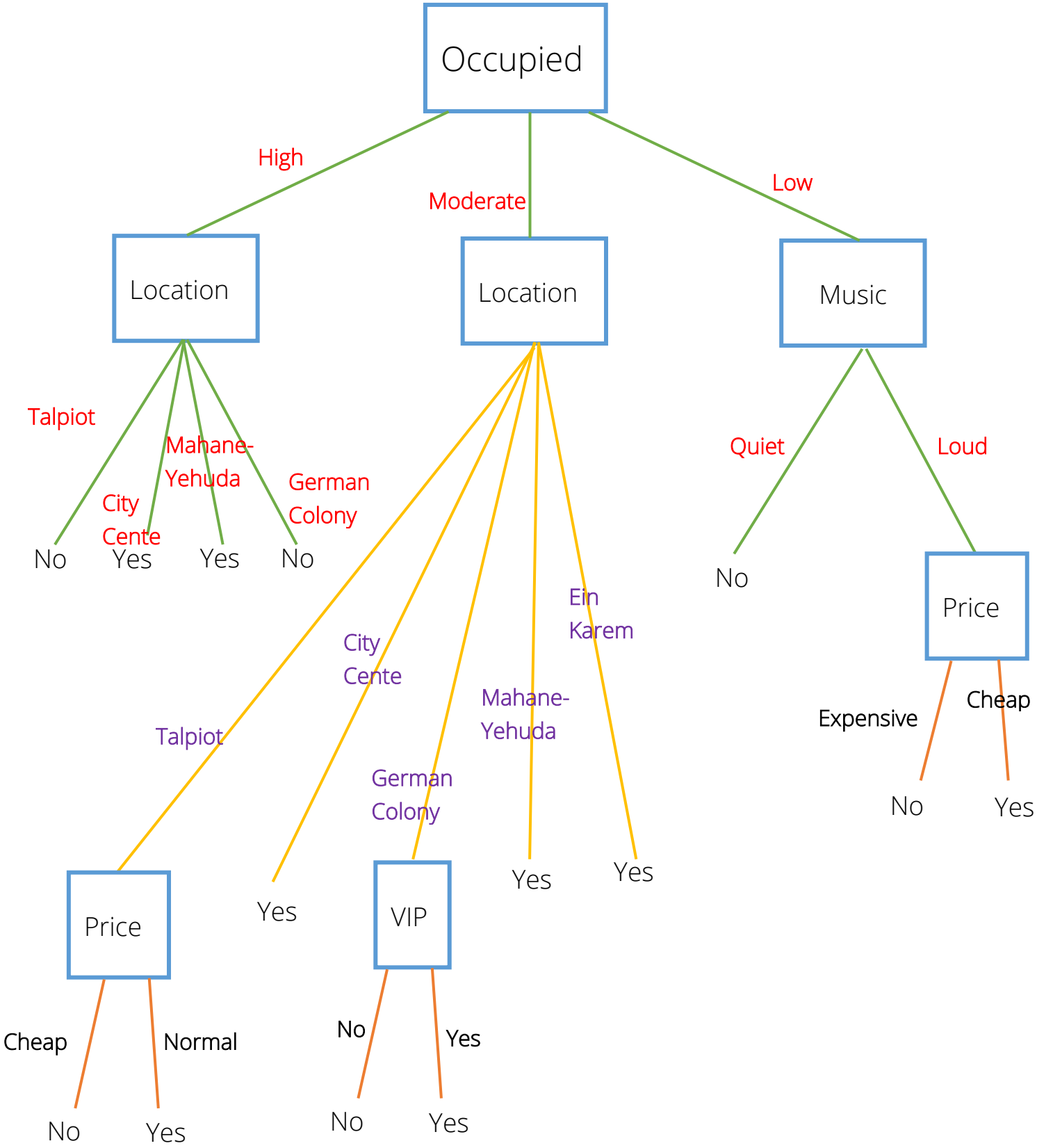
Occupied (Low) ->

Music (Loud) ->

Price (Cheap) ->

Enjoy : Yes

6. Decision Tree



7. Software Familiarization

Scikit-learn DecisionTreeClassifier:

There are some parameters in this classifier we can adjust conveniently when the data set is too large or there are too many dimensions of features. For instance,

max_depth: The maximum depth of the tree. We can adjust this parameter to prevent the tree from expanding too deep.

min_samples_split: Minimum number of samples required for a tree node.

min_samples_leaf: Minimum number of samples required to become a leaf node. If the samples are not enough, this node will be pruned.

The performance of our own implementation turns out to be great, and the execution time is 0:00:00.001004. In addition, our result is the same as the prediction of Scikit-learn DecisionTreeClassifier.

However, that's because the dataset now is small, so in the future, if we want to improve our code, we will try to combine Scikit-learn library's advantages.

8. Applications

Automated Insurance Plan Recommendations:

Based on the customers' personal information, the automated insurance plan recommendation system provides customers with the insurance plans which best meet their needs.

The system was trained by random forests. With 398818 training data and 170922 testing data from existing customer profiles, the system constructed a multitude of decision trees by using 85 dimensions of selected features including age, gender, occupation, annual income, location.....etc. Finally, the system has achieved 78% of correctness.

Landing page bounce prediction:

We can use decision tree to predict users' landing page bounce in order to increase or decrease the bounce rate.