Programming Assignment 1: Decision Trees

Patrick Humphries (pvhumphr@usc.edu)

[1] University of Southern California
[2] Viterbi School of Engineering
[3] INF552: Machine Learning for Data Science (32458)
[4] Spring 2020
[5] This paper was formatted with the Overleaf application using the "Springer Lecture Notes in Computer Science" template.

Abstract. The purpose of this assignment is to inculcate the concept of decision trees of machine learning. This is accomplished by three tasks:

- Part 1: Implementation
- Part 2: Software Familiarization
- Part 3: Applications

The implementation was challenging. Transforming the concepts of entropy and information gain into a Python program was the most time consuming. The familiarization was done with the popular pandas, matplotlib, and sklean libraries.

Keywords: Decision Tree · INF552 · Machine Learning

1 Part 1: Implementation

Python program "inf552_assignment_1_humphries.py" was developed using Jupyter Notebook of the same name. The decision tree was created by evaluating information gain for each attribute and adding the attribute to a branch. When the branch is completed, it is added to the tree.

Since I do not have a programming background in Python, the development of the program was difficult. First, several YouTube lectures were view just to get a definite understanding of entropy and and information gain. There were three iterations of the program code. Each had a different scheme for accumulators and decision tree.

There is no optimization of code. This pedantic code uses strings for labels and values. This results in extensive overhead. The program works fine with the limited number of features and binary labels. However, if the data were millions of rows, performance would be abysmal. Taking a clue from sklearn, all feature

and label values would be encoded as short integer values. The accumulators and decision trees would be multi-dimensional arrays (lists) that would grow dynamically. Maybe next time.

1.1 Algorithm

The root of the decision tree is the first attribute with the most information gain.

Subsequent attributes are added by a recursive function.

The recursive function accumulates the "Yes" and "No" values of the Enjoy label in an accumulator for the attribute and for each of the values. The sum of the weighted entropy values for each of the attribute values is subtracted from the entropy value of the attribute, resulting in information gain.

When information gain is zero does not mean entropy for the attribute is certain. It will be certain if the Enjoy values are either all "Yes" values or all "No" values. There is one case in the data provided that yielded both "Yes" and "No" values. For simplicity, the value for Enjoy returned to the user was "Uncertain". In a more sophisticated algorithm, the majority value would be returned or the weighted values of each Enjoy value.

When there is information gain greater than zero, the attribute is appended to the current branch and the function is called again for each of the attribute values. This is the recursive part.

When information gain is zero, then this attribute is the end of the current branch. The attribute is added to the branch, and the branch is added to the decision tree.

1.2 Accumulators

```
# Build an accumulator for counting values for the Enjoy label.
if Attribute == 'Occupied':
    accumulator = {'High':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'Moderate':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'Low':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0}}

elif Attribute == 'Price':
    accumulator = {'Expensive':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'Normal':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'Cheap':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0}}

elif Attribute == 'Music':
    accumulator = {'Loud':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'Quiet':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0}}

elif Attribute == 'Location':
    accumulator = {'Talpiot':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'City-Center':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'German-Colony':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'Ein-Karem':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0},
                   'Mahane-Yehuda':{'Yes':0,'No':0,'Entropy':0.0,'Weight':0.0}}
```

Calculating the information gain for an attribute requires many numbers. These numbers are stored in an accumulator. Counts are determined by conducting a filtered read of the training data. Entropy is calculated by a function that uses counts for the two Enjoy values. Values for weights of the attribute values are calculated.

1.3 Decision Tree

```
Tree
len(Tree): 17
[{'Occupied': 'High'}, {'Location': 'Talpiot'}, {'Enjoy': 'No'}]
[{'Occupied': 'High'}, {'Location': 'City-Center'}, {'Enjoy': 'Yes'}]
[{'Occupied': 'High'}, {'Location': 'German-Colony'}, {'Enjoy': 'No'}]
[{'Occupied': 'High'}, {'Location': 'Mahane-Yehuda'}, {'Enjoy': 'Yes'}]
[{'Occupied': 'Moderate'}, {'Location': 'Talpiot'}, {'Price': 'Normal'}, {'Enjoy': 'Yes'}]
[{'Occupied': 'Moderate'}, {'Location': 'Talpiot'}, {'Price': 'Cheap'}, {'Enjoy': 'No'}]
[{'Occupied': 'Moderate'}, {'Location': 'City-Center'}, {'Enjoy': 'Yes'}]
[{'Occupied': 'Moderate'}, {'Location': 'German-Colony'}, {'VIP': 'Yes'}, {'Enjoy': 'Yes'}]
[{'Occupied': 'Moderate'}, {'Location': 'German-Colony'}, {'VIP': 'No'}, {'Enjoy': 'No'}]
[{'Occupied': 'Moderate'}, {'Location': 'Ein-Karem'}, {'Enjoy': 'Yes'}]
[{'Occupied': 'Moderate'}, {'Location': 'Mahane-Yehuda'}, {'Enjoy': 'Yes'}]
[{'Occupied': 'Low'}, {'Location': 'Talpiot'}, {'Enjoy': 'No'}]
[{'Occupied': 'Low'}, {'Location': 'City-Center'}, {'Price': 'Normal'}, {'Enjoy': 'Uncertain'}]
[{'Occupied': 'Low'}, {'Location': 'City-Center'}, {'Price': 'Cheap'}, {'Enjoy': 'No'}]
[{'Occupied': 'Low'}, {'Location': 'Ein-Karem'}, {'Price': 'Normal'}, {'Enjoy': 'No'}]
[{'Occupied': 'Low'}, {'Location': 'Ein-Karem'}, {'Price': 'Cheap'}, {'Enjoy': 'Yes'}]
[{'Occupied': 'Low'}, {'Location': 'Mahane-Yehuda'}, {'Enjoy': 'No'}]
```

The decision tree is a list of lists. The outer list is the decision tree. Each subordinate list is a branch of the decision tree. Each branch is a list of dictionaries. Each dictionary contains attribute name and value. The first (far left) dictionary is the root of the decision tree. The last (for right) dictionary is the label Enjoy.

1.4 Decision Tree Processing

After the user has entered attribute values via the dialog, the program searches the branches in the decision tree. When a match is found, the value for Enjoy is returned. There are cases where a branch is not found. The training data did not provide enough data for all possible combinations. In this case, the value "Unknown" is returned. Two rows in the training data had identical feature values but different label values. In this case, "Uncertain" is returned.

Regarding the prediction for Part 1c, the program selected the branch that had "Moderate" for Occupied, "City-Center" for Location. The Enjoy value is 'Yes'. All other feature values did not provide any further information gain.

1.5 User Interface

```
Summary
This program does the following:
1.  Loads a csv file into a Decision Tree.
2.  Give the user the option to display the Decision Tree.
3.  Give the user the option to test the Decision Tree by entering attribute values.

Welcome

Enter one of the following options:
0 Exit
1 Display Decision Tree
2 Check Enjoy by entering attribute values.
==>
```

The user interface has a menu consisting of three items.

- 0 Exit: The user can exit the program at any time by entering zero at the prompt.
- 1 Display Decision Tree: User can view a tabular version of the decision tree.
- 2 Check Enjoy by entering attribute values: The user is presented with a series of prompts for attribute values. After the last prompt, the program will display the Enjoy value as determined by the decision tree. The default values presented answer the requirement for Part 1c with the answer "Yes".

2 Part 2: Software Familiarization

2.1 sklearn

The popular sklearn library was implemented for this task. Program scikit_learn.py implements this library along with pandas and matplotlib. This program not only develops a decision tree, it also displays its score, number of nodes, and visualization. The program was developed with Jupyter Notebook of the same name.

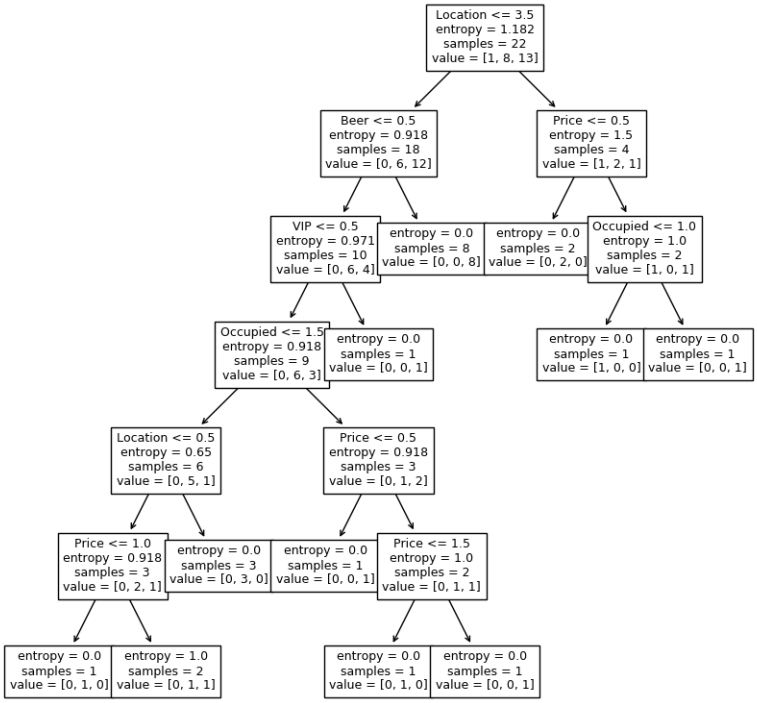
2.2 Challenges

The grader may encounter library issues when executing the Python program. These issues were resolved by updating the libraries. Program update_libraries.py is provided for the grader's convenience.

2.3 Output

Printed output.

```
First five rows of the csv file:

   Key  Occupied      Price  Music       Location  VIP Favorite Beer Enjoy
0    1      High  Expensive   Loud        Talpiot   No            No    No
1    2      High  Expensive   Loud    City-Center  Yes            No   Yes
2    3  Moderate     Normal  Quiet    City-Center   No           Yes   Yes
3    4  Moderate  Expensive  Quiet  German-Colony   No            No    No
4    5  Moderate  Expensive  Quiet  German-Colony  Yes           Yes   Yes

First five rows of inputs_n dataframe after encoding:

   occupied_n  price_n  music_n  location_n  vip_n  beer_n
0           0        1        0           4      0       0
1           0        1        0           0      1       0
2           2        2        1           0      0       1
3           2        1        1           2      0       0
4           2        1        1           2      1       1

Decision Tree Node Count: 21

Decision Tree Score: 0.95

Decision Tree Plot
```



Plotted output.

3 Part 3: Applications

3.1 Using Decision Tree to Predict Armed Conflicts in Sudan

I suspect the intelligence community uses a variety of analytical tools, and decision trees are included. This title piqued my interest because I was in the intelligence community while serving in the USAF. A paper found in the "International Journal of Computer" describes such an application. There were six features used (land, weapon, season, time, force, tension). The package used for analysis is SPSS. What I found interesting is the poor quality of this analysis. Deferring to a software package package for a volatile topic as predicting con-

flict, an alternate method or two should be applied to verify results. Also, the visualization was poor.

3.2 Decision Trees

This not an actual application but a description of how a decision trees are used in strategy. It uses Moses leaving Egypt as an example. The article is a plethora of history, terms, and methods. Above all, it was written by the Central Intelligence Agency.

References

1. Alpaydin, Ethem. Introduction to Machine Learning, Third Edition. MIT Press, 2014.
2. "Decision Trees", "Central Intelligence Agency". www.cia.gov/library/centerforthe-studyofintelligence/kentcsi/vol18no4/html/v18i4a03p_0001.htm
3. Using Decision Tree to Predict Armed Conflicts in Sudan International Journal of Computer, 2015, Volume 16, No. 1, pp 9-17. www.researchgate.net/publication/295907180_Using_Decision_Tree_to_Predict_Armed_Conflicts_in_Sudan