ECE 5984 SP22 – Prof. Jones – HW 3

Due Thursday, March 1, 2022 – 11:59 PM via Canvas

Part 1 (10 points):

We have landed on an alien planet, and it's covered in mushrooms. We have made some observations of whether certain mushrooms are poisonous (don't ask how) and the results are in the following table.

Using the Information Gain-based method described in class, determine the best decision tree to embody the following information. White, Tall and Frilly are the attributes and Edible is the result or output that your decision tree must generate:

White	Tall	Frilly	Edible
0	0	0	F
0	0	0	F
0	0	0	F
0	0	0	T
0	0	1	T
0	0	1	T
0	1	0	T
0	1	0	T
0	1	0	T
0	1	0	T
0	1	0	T
0	1	1	F
0	1	1	F
1	0	0	T
1	0	0	T
1	0	0	T
1	0	1	F
1	1	0	T
1	1	0	T
1	1	0	T
1	1	0	T
1	1	1	F
1	1	1	F
0	1	1	T

You should use the criterion of maximum information gain to select the attributes to use in each level of the tree, starting from the top. You must show all of your work. Draw the resulting tree.

You should solve this part 1 on paper, then please scan your work into the single pdf or Word file for your submission. You must be complete in your work and clear in your writing!

Part 2 (7 points):

In the Datasets folder in the Files area of our Canvas site, you will find an Excel spreadsheet called "AlienMushrooms.xlsx". This contains the dataset represented above. You are to write a simple Python program to read this spreadsheet and to generate a scikit-learn Decision Tree classifier for the data. Use the "entropy" criterion for the DT generation. Use the "writegraphtofile" function I have provided below to display the resulting tree, and compare it to your manual results above.

Part 3 (8 points):

In the Datasets folder in the Files area of our Canvas site, you will find an Excel spreadsheet called "FlareData.xlsx". This contains the dataset for this homework assignment. You are to do the following:

1. The predictors are all categorical; to use the decision tree capabilities of scikit-learn, you will need to replace the letter or text labels with numeric values. You may do this manually in the spreadsheet OR in your Python code – either approach is just fine.
    a. (note: in general, we need to be VERY careful in doing this, and we will discuss this again. Since we are using a decision tree model, this approach is less risky).
2. In this dataset there are three possible target values that we might want to predict: C class, M class and X class. For this assignment let's concentrate on the C class as our target variable. (Note: M and X cannot be used as predictors).
3. Use python to develop code to read the data tab from the spreadsheet, create two decision tree classifiers and write the resulting trees to a png file. One tree should use the entropy criterion, while the other should use the gini criterion. Limit the depth of the trees to four. Below you will see a simple Python function to write a decision tree classifier to a png image file. Compare the two trees: which one is better? Why?

Assemble a single Word or pdf file containing:

- Images of your answer for part 1;
- Your code for parts 2 and 3 (all of your code) – please paste as plain text with no dark mode;
- Your resulting decision trees;
- Your discussion on any differences between your parts 1 and 2.
- Your discussion on the differences between the two trees generated in part 3.

Please submit your Word or pdf file via Canvas. Also attach your .py file(s); if you use Jupyter, please export and attach a .py file (don't submit your .ipnyb notebook file).

```python
import pydotplus
import collections

# for a two-class tree, call this function like this:
# writegraphtofile(clf, ('F', 'T'), dirname+graphfilename)
def writegraphtofile(classifier, classnames, pathname):
    dot_data = tree.export_graphviz(clf, out_file=None,     # merely to write the tree out
                         feature_names=featurelabels.tolist(),
                         class_names=classnames,
                         filled=True, rounded=True,
                         special_characters=True)
    graph = pydotplus.graph_from_dot_data(dot_data)
    colors = ('lightblue', 'lightgreen')
    edges = collections.defaultdict(list)
    for edge in graph.get_edge_list():
        edges[edge.get_source()].append(int(edge.get_destination()))
    for edge in edges:
        edges[edge].sort()
        for i in range(2):
            dest = graph.get_node(str(edges[edge][i]))[0]
            dest.set_fillcolor(colors[i])
    graph.write_png(pathname)
```