ECE 5464 SP24 – Prof. Jones – HW 2

Due Tuesday, February 27, 2024 – 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
1	0	1	F
0	1	1	F
0	1	1	T
1	0	0	T
0	0	1	T
0	0	0	F
1	1	0	T
1	1	0	T
0	1	1	T
1	1	0	T
1	0	0	T
1	0	0	T
1	1	1	F
1	1	0	T
0	0	0	F
0	1	0	T
0	1	0	T
0	1	0	T
0	1	0	T
0	0	0	T
0	1	0	T
0	0	1	T
1	1	1	F

You should use the criterion of maximum information gain to select the attributes to use in each of the top two levels of the tree, starting from the top. You should have a total of three decisions selected. 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!
To be completely clear, you are not to write any code or anything for part 1. You <u>can</u> use Excel or other environment to calculate the numeric results.

Part 2 (10 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. Use the classifier score function on the training data to measure the tree performance, and write this out to the Python console using print().

Part 3 (15 points):

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

1. Load the dataset in the usual manner.
2. Go through *most* of the usual steps of model development as discussed in class. Here is a bit more detail:
    a. 1 - Check for data errors
    b. 2 – Detect and correct missing values
    c. 3 - Add computed fields – *not this time*
    d. 4 – Normalize – *not this time*
    e. 5 - Sample and partition into training-test-validation
    f. 6 - Feature selection – *not this time*
    g. 7 - Model training
    h. 8 - Measure model performance
3. The raw target value in this file is the variable "readmitted". It has three values: "NO", "<30" and ">30" (indicating whether a patient was readmitted in less than 30 days, greater than 30 days or not at all). We want to develop two models: one to predict whether a patient was readmitted at all,, and the other is the multivariate target 'readmitted'.
4. For each type of target (binary and multiclass), develop a decision tree using the entropy criterion, print out the accuracy results and write the resulting decision tree to a png file. 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. Use the classifier score function on both the training and test sets to measure the tree performance, and write this out to the Python console using print().

Hints:

- Some of the predictors are categorical; to use the decision tree capabilities of scikit-learn, you will need to apply one-hot encoding to the variable. The pandas function `pd.get_dummies()` may be helpful.
- Some of the predictors are ordinal; you will need to replace the values with numeric values. The pandas function `df[featurename].factorize()` may be helpful.

    - Note that you may have to ensure that the values are encoded in the proper order; I had to replace the original values of the feature with something that sorted properly in alphabetical order.

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 (no screenshots);
- The console output of your program that shows the performance on training and test sets for parts 2 and 3;
- 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 four (or more) files via Canvas: your Word or pdf file as described above, your .py file(s) (if you use Jupyter, please export and attach a .py file - don't submit your .ipnyb notebook file), and the two .png images of your decision trees.

```
import pydotplus



# for a two-class tree, call this function like this:
# writegraphtofile(clf, ('F', 'T'), dirname+graphfilename)

# for a multi-class tree, call this function like this:
@ writegraphtofile(clf, featurenames, dirname+graphfilename)

def writegraphtofile(clf, featurelabels, filename):
    dot_data = tree.export_graphviz(clf, feature_names=featurelabels, out_file=None)
    graph=pydotplus.graph_from_dot_data(dot_data)
    graph.write_png(filename)
```