Project on Decision Trees

The project consists of implementing an algorithm for building and postpruning a decision tree. In particular, you should implement the algorithm whose pseudocode has been provided in the slides our lecture on decision trees. The input (extracted from the Titanic Dataset) is stored in a file (*data.csv*), one line per record with each attribute separated by a comma (Comma Separated Values - CSV file). There are four features: Sex, Pclass, Embarked and Survived. Sex, Pclass and Embarked are categorical attributes with *Sex* taking values in $\{0, 1\}$, *Pclass* indicates the passenger class and takes values in $\{1, 2, 3\}$, *Embarked* indicates the port of embarkation and takes values in $\{0, 1, 2\}$ while *Survived* is the (binary) class taking values in $\{0, 1\}$. The attributes are stored in that order in the input file. You should build a binary decision tree where each node has two branches, except the leaves. The attribute used to split and the values for each split (i.e. the values in the constraints c_1,...,c_k in the slides) are those that maximize the Gini Split. You should also implement the postpruning algorithm where the generalization error is defined as: number of training errors + $\alpha\cdot$ number of leaves.

You should implement the following functions:

1.a a function *BuildDecisionTree()* which receives in input a file with the input data and the paramenter $minNum$ (minimum number of recored per node) and builds and returns a decision tree object (not yet pruned). You can choose the type of decision tree object to create and return (e.g., a Python object, a dictionary, etc.)

1.b a printing function *printDecisionTree()* that takes in input the decision tree previously built and prints it in output a string (see instructions below).

2 a function *generalizationError()* that receives in input the input data, a tree (e.g. the one previously built with the *BuildDecisionTree()* function) and a parameter α. The function has to compute and return the generalization error.

3 a function *pruneTree()* that receives in input a tree, parameters α and $minNum$ and returns a pruned decision tree. Use the printing function previously developed [1.b] to print the pruned tree.

Printing the Tree: We recall that the level of the root is 0, while the level of any node in the tree is equal to the level of its parent +1. You should print the tree starting from the root node in a breadth first search (BFS) fashion. After printing the root node, print all the nodes at level 1 starting from the left child of the root, that is, the node which satisfies the feature constraint of the root node. Then print all nodes at level 2 while printing left children first and so on. For each node you should print the following information (one element per line) in this order:

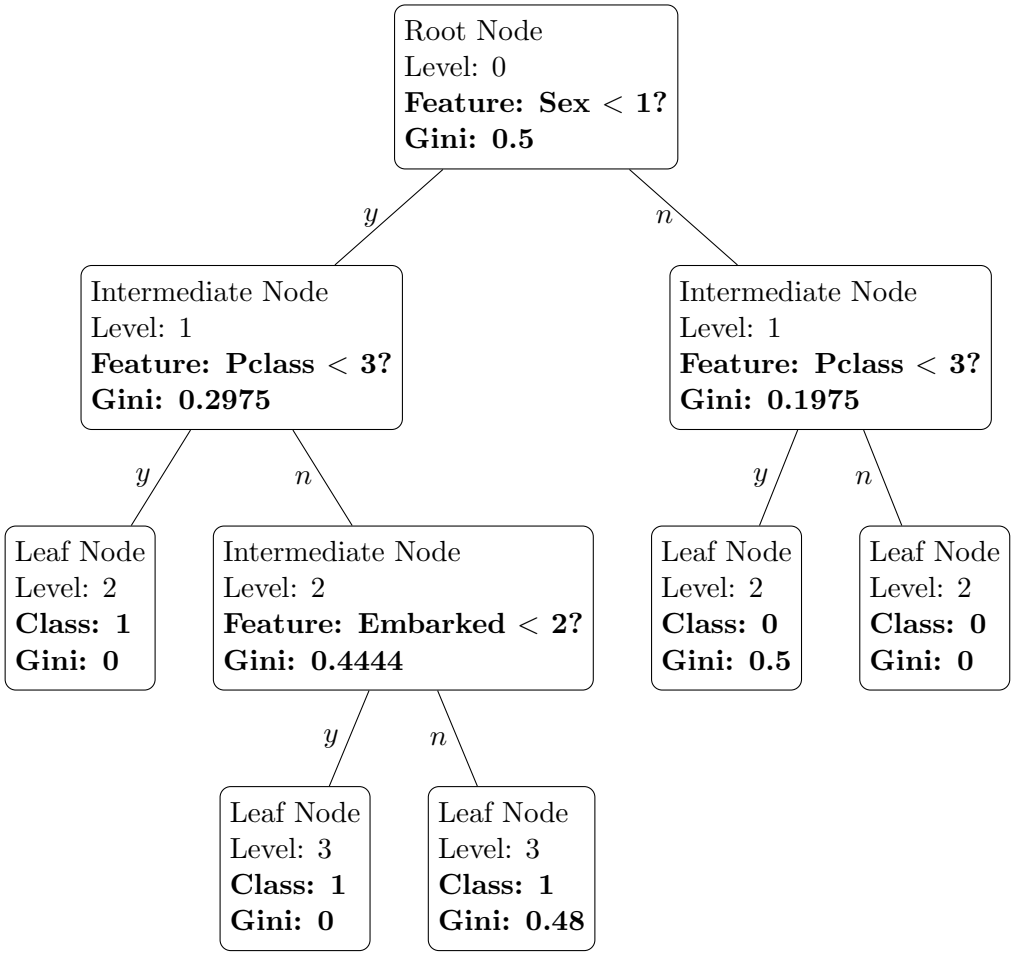
- whether it is the root, in which case you should print "Root", a leaf (print "Leaf"), or an intermediate node (print "Intermediate")
- the string "Level" followed by the level of the node
- the string "Feature" followed by the attribute name, e.g. "Sex" followed by a list of values of the attribute. Each element is separated by the space character. E.g. "Feature Sex 0". This lines specified the constraint on the feature. If the node is a leaf, skip this line.
- the string "Gini" followed by the gini value of the records associated with that node.

Example:

1. Let's consider the first 20 rows of the dataset only.

```
Sex,Pclass,Embarked,Survived
1,3,2,0
0,1,0,1
0,3,2,1
0,1,2,1
1,3,2,0
1,3,1,0
1,1,2,0
1,3,2,0
0,3,2,1
0,2,0,1
0,3,2,1
0,1,2,1
1,3,2,0
1,3,2,0
0,3,2,0
0,2,2,1
1,3,1,0
1,2,2,1
0,3,2,0
0,3,0,1
```

2. The tree you should obtain is the following one:



3. The output of the print function is the following one:

```
Root
Level 0
Feature Sex 0
Gini 0.5

Intermediate
Level 1
Feature Pclass 1 2
Gini 0.2975
*****
Intermediate
Level 1
Feature Pclass 1 2
Gini 0.1975

Leaf
Level 2
Class 1
Gini 0
*****
Intermediate
Level 2
Feature Embarked 0 1
Gini 0.4444
*****
Leaf
Level 2
Class 0
Gini 0.5
*****
Leaf
Level 2
Class 0
Gini 0

Leaf
Level 3
Class 1
Gini 0
*****
Leaf
Level 3
Class 1
Gini 0.48
```

What to send: You should send the python code of all the three different functions - all the functions have to be contained in a python file called **decision_functions.py**. You should send moreover a *main.py* file in which you import the functions and run them in order. The printed tree output obtained in [1.b] should be written on a file named "output_tree.txt", the generalization error obtained in [2] should be written on a file named "generalization_error.txt" and, finally, the postpruned tree obtained in [3] should be printed and written on the "postpruned_tree.txt" file. You should send BOTH the code and the txt files containing the results obtained running your code on the whole dataset.

Self-evaluation:

- You have to test that your functions can be properly imported in a script running the test script *test_import_functions.py*. The script returns success if it is able to properly import the 3 functions. An error is raised otherwise.
- You can test that your code is properly running by training the decision tree on the first 20 lines of the dataset (as in the above example) and checking that the results in the output file coincides (gini, levels, etc.) with the output of the example. You have to use $minNum = 5$ and $alpha = 0.5$ as input parameters.
- **Best**: You can choose a different subset of rows in the dataset (e.g. rows 20 - 40, rows 53-78, etc.), manually produce the corresponding decision tree (compute splits, etc.) and check that your code produces the same tree.