CS 165B – Machine Learning, Spring 2023

Machine Problem #2
Due Thursday, May 16 by 11:00pm

TA: Danqing Wang
Email: danqingwang@ucsb.edu

Note: This assignment should be completed individually. You are not supposed to copy code directly from anywhere else, and we will review the similarity of your submissions. You should write the code from scratch, and you **are not allowed** to directly use any third- party tools such as sklearn. You can use any default Python libraries and the imported packages at the beginning of ***mp2.py***

In this project, we have provided **a basic framework** for you to start with. You can follow the framework and finish the ***TODO*** parts. You can also **start from scratch**, but remember to keep the function name and argument of ***run_train_test(training_input, testing_input)*** unchanged.

For convenience, we recommend using **Pandas** library to deal with the data. If you are not familiar with this package, please spend 10 minutes reading the tutorial **here.** You only need to know the basic operation.

Problem Overview:

Change the **Python 3** program ***mp2.py*** that creates a decision tree classifier for the Covid classification problem. The training and testing data sets are available in the starter package. The file **data.md** has the information about the dataset, label, attribute.

- *data/**: include ***train.csv*** for training and ***dev.csv*** for development. We will have an extra private test set for the final grading.
- *data.md*: provide detailed data information.
- *mp2.py*: the python file you need to work on and submit.
- *MP2.pdf*: this instruction file

Code Instructions

The detailed instructions are provided as comment in the ***mp2.py***. General speaking, you need to design a binary decision tree based on several features to determine whether this patient will die from COVID.

We provide several classes and functions:

- **Node**: The data structure for decision tree node. In this project, we only consider the binary tree, so one node will have at most two children: left and right. For each non-leaf node, it will have a feature to be split and a value performing as the threshold to split data. For leaf nodes, it will have a prediction value indicates which class it belongs to. *(Nothing to do here)*
- **Decision Tree**: The DecisionTree object. There are three hyperparamters to initialize*: max_depth, min_samples_split, min_information_gain*. We provide some default values in the comment, and you can also try various values to improve the performance. This class has the following functions:
    - *fit(self, training_data, training_label)*: the entry to build the decision tree *(Nothing to do here)*
    - *GrowTree(self, data, label, counter)*: the main function to build the decision tree based on the impurity ***(several TODOs here)***
    - *BestSplit(self, data, label)*: the function to find the best split. You can try different impurity functions to split the data. (***TODO***)
    - *predict(self, data)*: predict test data based on the decision tree. You will need to recursively check tree nodes. (***TODO***)
    - print_tree(self) & print_tree_rec(self, node): visualize the tree to help debug *(Nothing to do here)*
- **run_train_test**: the outer function to trigger the whole training and test process. It will be called for grading. Make sure the argument format is not changed. You can refer to main function in ***mp2.py*** to see how it will be called. (***TODO***)
- **cal_accuracy():** used for calculating the accuracy. *(Nothing to do here)*

Evaluation Instructions

You can test your program with the **train.csv** and **dev.csv** provided in this starter package. You can directly run ***mp2.py*** for checking:

$ **python mp2.py**

This will output the accuracy you get on the dev dataset. The final score will be graded on another private test set (use the same training set).

Submission Instructions

You should upload ***mp2.py*** to Gradescope for grading.

Grading Instruction:

The score on Gradescope are computed based on the accuracy using the following rule:
• If your accuracy is above 50%: $Score = min(100, 70 + 100 * (Accuracy - 50\%))$
• If your accuracy is below 50%: $Score = max(0, 70 - 0.2 * 100 * (50\% - Accuracy))$

If your code can run successfully, you will get a basic 70 points. If not, your score will be evaluated based on the degree of completion (no more than 60 points).

You can also check the leaderboard about the performance of others. **Top-3 performers will get extra credit 10%.**

Late submission are accepted with 20% reduced credit each day. For example, you will get 80% of full credit if one day late, 60% if two days late and so on so forth.