Data Minig Project: Random Forest

Due date: 11:59pm December 10

Introduction to Random Forest

Random Forest is a branch of Ensemble Learning. The basic idea of Ensemble Learning is to generate multiple classifiers which learn and make predictions independently, and then to combine the predictions of these classifiers into a single prediction.

Random Forest will create a number of random decision trees automatically. When making a decision, the new record to be predicted will be run down all of the trees and assigned a predicted label. The result will be the voting majority.

A typical Random Forest algorithm is as follows:

1. Draw n_{tree} bootstrap samples from the original data.
2. For each of the bootstrap samples, grow an unpruned Decision Tree, with the following modification: at each node, rather than choosing the best split among all attributes, randomly sample m_{try} of the attributes and choose the best split from among those variables.
3. Predict new data by aggregating the predictions of the n_{tree} trees.

To simplify the task, we are making a change to the algorithm: Instead of randomly choosing a subset of all attributes at each node,before building each decision tree, we randomly choose a subset of attributes and use it for (all nodes) of that decision tree. In other words, your program just needs to randomly choose a subset of all attributes and then create a normal Decision Tree for each bootstrap sample.

Project Description

The goal of this project is to implement a Decision Tree classifier and a Random Forest classifier from scratch. You are given a folder which contains unfinished python code and data set for training and testing. The structure of the folder is as follows:

```
project
├── src
│   ├── DecisionTree.py
│   ├── RandomForest.py
│   └── main.py
├── data
│   ├── mushrooms_training.data
│   ├── mushrooms_testing.data
│   └── description.txt
└── readme.txt
```

1 The *src* folder contains the code of the models. The basic skeletons of Decision Tree and Random Forest have been defined in *DecisionTree.py* and *RandomForest.py* separately. The *main.py* file is the entry of the program, you will use it for running your models. There are some functions that have been commented, those are recommended but not required to implement.

2 The *data* folder contains the data set for training and testing your model.

3 The *readme.txt* file is empty. You need to write down your write-up here after finishing the code.

INTRODUCTION TO PYTHON

You will write Python code in this project. Python is a popular high-level interpreted programming language with good code readability. For those who are not familiar with Python, this *Beginner's Guide* is a good start. You can also find a good interactive Python tutorial *here*. You can check out *PyCharm* if you need an IDE, they provide free student licenses.

DATA

The data set we use is from *UCI Machine Learning Repository*. It includes description of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. We have divided it into two parts: training set and testing set. You can find the detailed description and attribute information from both the website and the description file in the data folder.

REQUIREMENTS

1 You need to implement Decision Tree classifier in *DecisionTree.py*. The *__init__*, *train*, *predict* functions have been defined but not implemented, you need to implement these methods without modifying the definitions.

2 After finishing your Decision Tree classifier, you can run the following command to test your classifier using the given data set:

```
python src/main.py –m 0 –t /path/to/training/file –e /path/to/testing/file
```

3 Then you need to implement Random Forest classifier in *RandomForest.py*. The *__init__*, *train*, *predict* functions have been defined but not implemented, you need to implement these methods without modifying the definitions.

4 After finishing your Random Forest classifier, you can run the following command to test your models using the given data set:

```
python src/main.py –m 1 –t /path/to/training/file –e /path/to/testing/file –n 20
```

There are many machine learning libraries that provide Decision Tree and Random Forest functions (e.g. *scikit-learn*). You **MUST NOT** use any libraries that provide such functions. You need to implement the classifiers from scratch.

GRADING

- Correctness of models (60%): You need to make sure the logic of your models are correct. Your models will be tested using the functions in the main.py file on the CS department server, so you also need to make sure your code can be run on CS Linux servers. Read *New CS Student Quick Reference* for how to access the server.
- Quality of codes: (30%): You will lose points if your code lacks readability. Comments would be helpful for improving your code quality.
- Write-up(i.e. readme.txt) (10%): Your write-up should include your name, descriptions of your implementation, anything the TA needs to know to run your program, acknowledgment of any part that does not yet work, and names of the students with whom you have discussed the project. NOTE: This is a single-person project. Optionally, you may discuss the project with other student(s), but only at a high level. You need to write all the code yourself. Copying code will be easily detected and subject to academic integrity severe punishments. If you do discuss with other students, you need to write down their names here.

SUBMISSION

You need to submit your project from you CS department server, the submit command is:

```
submit mjia dm_project <your file/folder to submit>
```