

DATAMINING FINAL PROJECT REPORT

Name: Yufeng Yuan

UML ID: 01506240

E-mail address: Yufeng_Yuan@student.uml.edu

1. 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
```

- a) 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.
- b) The data folder contains the data set for training and testing your model.
- c) The readme.txt file is empty. You need to write down your write-up here after finishing the code.

2. 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.

3. DECISION TREE

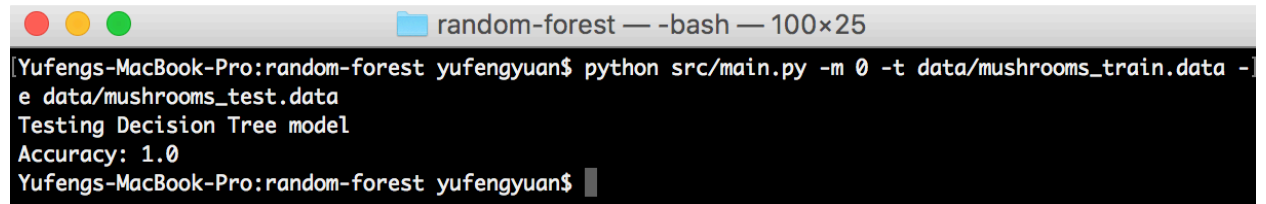
For decision tree, I used the information-gain for finding the best split point. Before split the data, I'll calculate all possible attribute value's information gain and find the maximum one, and select its attribute and its value as the best split point.

For calculate the information gain, I created a new function called "entropy", giving a set of records, this function will return the entropy of the set.

DATAMINING FINAL PROJECT REPORT

After running the main function with the following commend

```
python src/main.py -m 0 -t data/mushrooms_train.data -e data/mushrooms_test.data
```

A terminal window titled 'random-forest — -bash — 100x25' on a Mac. The prompt is 'Yufengs-MacBook-Pro:random-forest yufengyuan\$'. The user enters the command 'python src/main.py -m 0 -t data/mushrooms_train.data -e data/mushrooms_test.data'. The output shows 'Testing Decision Tree model' and 'Accuracy: 1.0'.

```
Yufengs-MacBook-Pro:random-forest yufengyuan$ python src/main.py -m 0 -t data/mushrooms_train.data -e data/mushrooms_test.data
Testing Decision Tree model
Accuracy: 1.0
Yufengs-MacBook-Pro:random-forest yufengyuan$
```

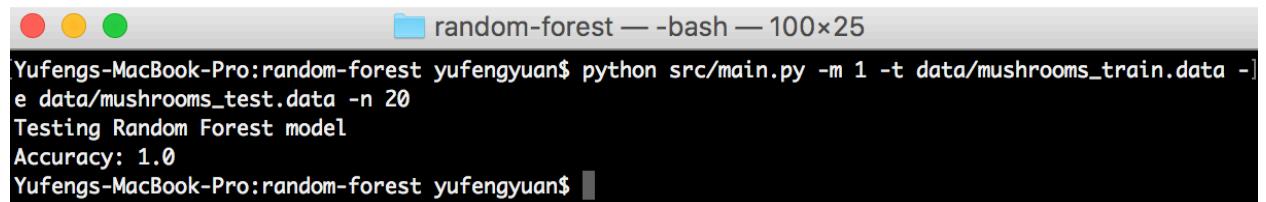
4. RANDOM FOREST

For random forest, I did the following step:

1. Draw n bootstrap sample for training.
2. For each bootstrap sample, randomly selected m attributes (I choose $m = 10$)
3. Training each sample and its attributes by decision tree.
4. Predict the test set in each tree, and select the majority result.

After running the main function with the following commend:

```
python src/main.py -m 1 -t data/mushrooms_train.data -e data/mushrooms_test.data -n 20
```

A terminal window titled 'random-forest — -bash — 100x25' on a Mac. The prompt is 'Yufengs-MacBook-Pro:random-forest yufengyuan\$'. The user enters the command 'python src/main.py -m 1 -t data/mushrooms_train.data -e data/mushrooms_test.data -n 20'. The output shows 'Testing Random Forest model' and 'Accuracy: 1.0'.

```
Yufengs-MacBook-Pro:random-forest yufengyuan$ python src/main.py -m 1 -t data/mushrooms_train.data -e data/mushrooms_test.data -n 20
Testing Random Forest model
Accuracy: 1.0
Yufengs-MacBook-Pro:random-forest yufengyuan$
```