Data Mining

Project 2 - Classification

- 沈育同
- P76061386

Environment

- Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-34-generic x86_64)

Prerequisite

- Python 3.6.4
- g++ 5.5.0

Install Dependency

```
$ pip install -r requirements.txt
```

Makefile

- Compile program

  ```
  $ make
  ```

- Install package

  ```
  $ make package
  ```

- Compile and execute program

  ```
  $ make run
  ```

Usage

```
$ python main.py [-h] [-max_lvl LEVEL] [-g GEN {true|false}]
                 [-d DATA_PATH] [-train TRAIN_SIZE]
                 [-test TEST_SIZE]
```

optional Options	Description
-h, --help	show this help message and exit
-max_lvl LEVEL	The maximum tree level. (default:5)
-g GEN	Generate new data. {True|False}(default:False)
-d DATA_PATH	Training and testing data path. (default:./data/)
-train TRAIN_SIZE	Number of Training data. (default:5000)
-test TEST_SIZE	Number of Testing data. (default:100)

Files Structure

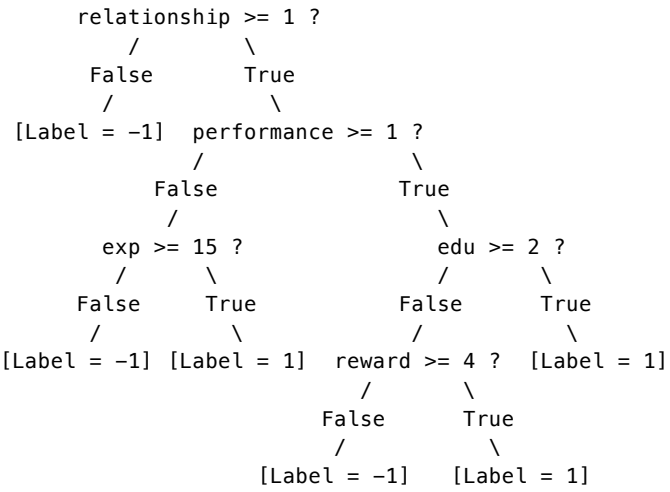
```
.
+-- include
|   +-- DecisionTree.hpp
+-- lib
|   +-- libCWrapper.so
|   +-- libDecisionTree.so
+-- cwrapper
|   +-- CWrapper.cpp
+-- dtree
|   +-- DecisionTree.cpp
+-- data
|   +-- train_x.npy
|   +-- train_t.npy
|   +-- test_x.npy
|   +-- test_t.npy
+-- Makefile
+-- requirements.txt
+-- main.py
```

- `include/DecisionTree.hpp`：DecisionTree物件prototype宣告
- `lib/`：編譯完之.so檔(library)，當python程式運行時將會引入這些library
- `cwrapper/CWrapper.cpp`：實作python與C++溝通介面
- `dtree/DecisionTree.cpp`：DecisionTree物件實作，包含constructor、fit、predict、print等功能
- `data/train_x.npy`：訓練資料之輸入。
- `data/train_t.npy`：訓練資料之標籤。
- `data/test_x.npy`：測試資料之輸入。
- `data/test_t.npy`：測試資料之標籤。
- `Makefile`：自動編譯C++ source code產生library並放置在，lib/目錄底下
- `requirements.txt`：python套件需求
- `main.py`：主程式

Data

- 本次實驗的資料是以公司內部升遷人選的情境作分類，我們定義age(年紀)、exp(工作經驗)、edu(教育程度)、performance(做事效率)、reward(獲得獎項或記功)、relationship(人際關係)等6種屬性，各種屬性分佈狀況如下：
  - age: 22 ~ 65
  - exp: 0 ~ 25
  - edu: 0 ~ 2 (大學、碩士、博士)
  - performance: 0 ~ 2 (差、普通、好)
  - reward: 0 ~ 10
  - relationship: 0 ~ 2 (差、普通、好)

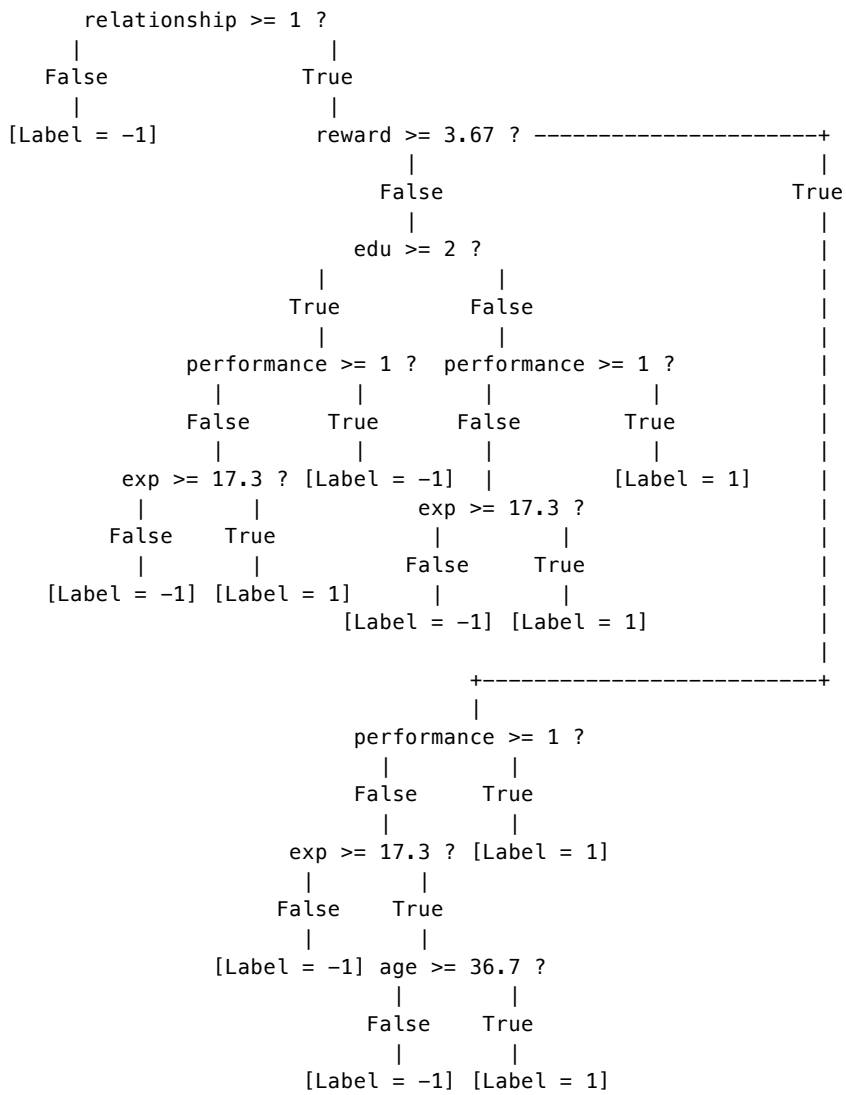
- absolutely right 定義如下：



- Training data: 5000筆
- Testing data: 100筆

Result

- 經過訓練後Decision Tree如下圖:



- Training

Training	Value
Accuracy	0.97140
Precision	1.00000
Recall	0.93515

- Testing

Testing	Value
Accuracy	0.97000
Precision	1.00000
Recall	0.93617

Comparison

- 經比較absolutely right與學習出來的Tree，發現只有Root的規則一致，再往後的分支就會與原本設定的absolutely right分支順序有些差異，甚至多出幾個absolutely right沒出現的判斷分支，我想這應該就是因為隨機產生的資料，其分佈剛好在一個原本absolutely right不存在的條件分支形成分離狀況，讓Tree誤解以為有其他的條件分支，進而衍生出原本設定中沒有的分支。
- 觀察精準度的部份，發現Decision Tree並無法學習到100%的精準度，我想這是因為對於連續數值型的資料(如年紀、工作經驗等)，模型建置時我們將這連續數值切割成數個離散的區間，這可能使得原本的條件分支落在在某個離散區間內，而這離散區間便會存在無法分割之"+1"類別及"-1"類別，故精準度無法達到100%。

Other Model

- 這實驗除了嘗試Decision Tree以外，另外我還使用了SVM來作對照，我們是採用scikit-learn所提供之SVC進行訓練其中 $kenel = 'rbf'$、$\gamma = 'scale'$，訓練出來的精準度如下:

- Training

Training	Value
Accuracy	0.90420
Precision	0.91570
Recall	0.86213

- Testing

Testing	Value
Accuracy	0.87000
Precision	0.86957
Recall	0.85106

Conclusion

- 比較SVM以及Decision Tree訓練成果之後，觀察其精準度發現SVM學習本次資料效果相較於Decision Tree為差，分析其原因應為本資料之生成為產生自一系列的if-else並且每一次皆只進行單一屬性判斷，其型式與Decision Tree較為相近，因此Decision Tree會學習到比較相近的結果，而SVM會試圖把原本資料投射到高維度空間進行分割，因此會作出較為複雜的分類，使得雖然在訓練時有90%精準度，卻在測試資料表現只有87%。

- 最後，不管是SVM、Decision Tree或是其他的分類器，都有著各自不同的分類方法，對於不同分佈資料採取不同的模型都可能會有不同的結果，因此當想訓練一個未知類別的資料進行分類，應該要嘗試各種不同的模型綜合考量後，再做出模型選擇的決定。像是Decision Tree雖然方法簡單，但卻能在這次實驗上有好的結果，而且其分支出來的判斷又能比較貼近人類的理解，是一種還不錯的模型選擇考量。

Authors

Yu-Tong Shen (https://github.com/yutongshen/)