ML Project #1 Report

Name: 曾偉杰

ID: 109550156

Data Preprocessing and Feature extraction

For the training and testing data of two given datasets,

#1: run `python dataset1_process.py -m "train"` or

`python dataset1_process.py -m "test"`

#2: run `python dataset2_process.py -m "train"` or

`python dataset2_process.py -m "test"`

If successes, you will see `Saving processed dataset done!`

Explanation:

從原始 dataset #1 中，可以觀察到全部的 feature 類型都是連續的且有許多的 missing value，處理的第一步驟，我先把具有太多 missing value 的 instance 捨去掉，捨去掉缺失 feature 數超過 feature 總數的 60%的 instance。再來，我利用每個 feature 的中位數來填補剩下來 instance 的 missing value。最後，我對每個 feature 處理 outlier，利用 clamp transformation 的方式替換掉 outlier。以上是我處理 dataset #1 的方式。

從原始 dataset #2 中，可以觀察到每句話的長度不一，不適合直接當作 feature，我們必須對它做 feature extraction，過程中，我使用了 NLTK(自然語言處理套件)來幫助我處理文本，首先我把每個 instance 句子中的名詞、動詞、形容詞和副詞找出來形成新的 list，然後我對這個 list 中的每個詞都各別找出它們的同義詞(synset)，用這些同義詞再形成新的 list。因為預測目標是電影好不好看的程度，所以我決定利用每句話和{ "perfective", "good", "bad", "atrocious"}這四個詞彙的相似度來當作我的 4 個新 feature，feature 計算的方式是用 synset list 中每個 synset 和上述四個詞各自計算 wup similarity(語義相似度)相加之後平均，如下圖，這樣就完成了 dataset#2 的處理。

f1	f2	f3	f4
0.242576	0.264307	0.316162	0.301314
0.181818	0.2	0.25	0.5

Build Decision Tree Model

For dataset #1: run `python build_decision_tree.py -d "dataset1" -md 20 -p 70`

For dataset #2: run `python build_decision_tree.py -d "dataset2" -md 25 -p 200`

If building model successes, you will see `Building decision tree model done!` and the model will be saved under directory ./model.

Explanation:

我建構決策樹的 Criteria 是用 GR(Gain ratio)，在每個 node 要找用於分割 dataset 的 feature 時，對於計算每個 feature 的 GR 時，我會先排序 feature 的值，然後在利用二分搜尋法找到會有最大 IG 的 feature 分割值，這樣就可以得出每個 feature 會有最大 GR 的分割值，再從這些 feature 中選出有最大的即為某次分割的 feature。從根開始，不斷遞迴上述的方法直到無法在分割為止(達到最大深度、數量小於最小分割需求數、label 都屬於同一種)，這樣決策樹模型就成功建立了。

Model Evaluation

For dataset #1: run `python evaluation.py -d "dataset1"`

For dataset #2: run `python evaluation.py -d "dataset2"`

If successful, you will see the accuracy, confusion matrix, F1-score (including Precision and Recall) of the decision tree model, such as:

#1:

```
Accuracy: 0.54

Confusion Matrix:
[[ 0  2  3  0  0]
 [ 0 24  8  0  0]
 [ 0 15 28  5  0]
 [ 0  2  9  2  0]
 [ 0  0  1  1  0]]

Classification Report:
              precision    recall  f1-score   support

           4       1.00      0.00      0.00         5
           5       0.56      0.75      0.64        32
           6       0.57      0.58      0.58        48
           7       0.25      0.15      0.19        13
           8       1.00      0.00      0.00         2

    accuracy                           0.54       100
   macro avg       0.68      0.30      0.28       100
weighted avg       0.56      0.54      0.51       100
```

#2:

```
Accuracy: 0.55

Confusion Matrix:
[[ 0  0  3  0  0]
 [ 0  0 18  1  0]
 [ 0  0 55  0  0]
 [ 0  0 19  0  0]
 [ 0  0  4  0  0]]

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.00      0.00         3
           1       1.00      0.00      0.00        19
           2       0.56      1.00      0.71        55
           3       0.00      0.00      0.00        19
           4       1.00      0.00      0.00         4

    accuracy                           0.55       100
   macro avg       0.71      0.20      0.14       100
weighted avg       0.57      0.55      0.39       100
```

Testing

For dataset #1: run `python test.py -d "dataset1"`

For dataset #2: run `python test.py -d "dataset2"`

If successes, you will see `Saving prediction done!` and the prediction result will be saved under directory ./prediction/dataset1/y_predict.xlsx

or ./prediction/dataset2/y_predict.xlsx