Lab 2: Decision Tree

22127266

14th April 2024

Contents

1 Information

Name	Student ID	Email
Nguyễn Bình Minh	22127266	nbminh22@clc.fitus.edu.vn

2 Completion status

No.	Specifications	Status
1	Preparing the data sets	100%
2	Building the decision tree classifiers	100%
3	Evaluating the decision tree classifiers	100%
	Classification report and confusion matrix	100%
	Comments	100%
4	The depth and accuracy of a decision tree	100%
	Trees, tables, and charts	100%
	Comments	100%

3 Environment and library

Running environment: Kaggle Notebook

List of library used:

- pandas:
- sklearn
- graphviz
- matplotlib
- seaborn
- IPython

4 Preparing the data sets

Firstly, data will be exported from "nursery.data.csv" file. The first 8 columns will be exported as "Feature" and the last column is the "Label" ("Class").

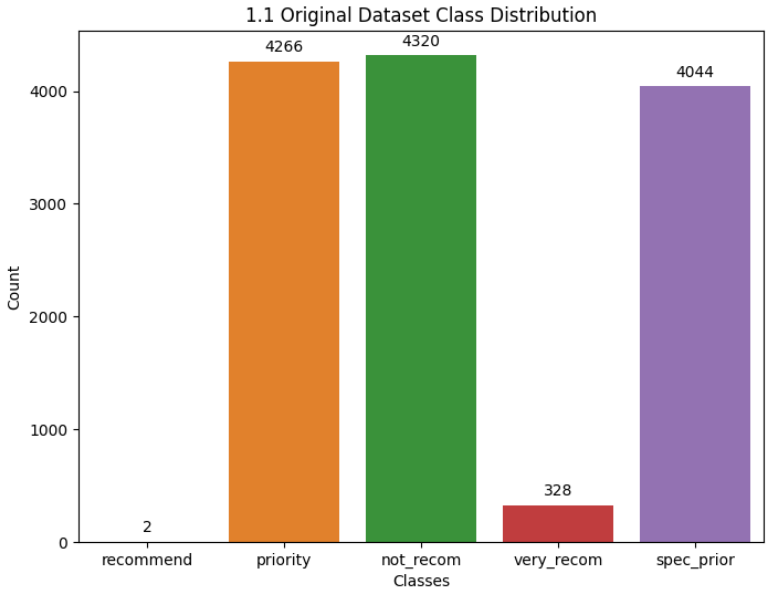
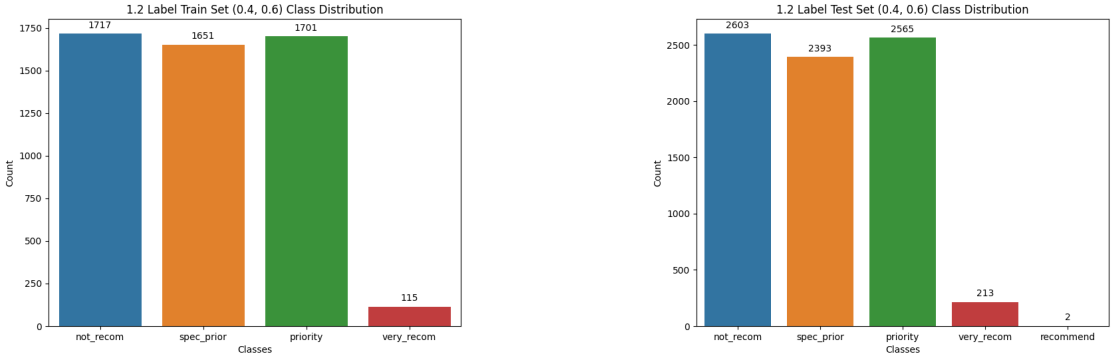


Figure 1: The original dataset class distribution

Using the `train_test_split()` function from sklearn library to shuffle and split the exported data into 4 sets: feature train sets, feature test sets, label train sets and label test sets. Each set has 4 subsets corresponded to the given ratio ((0.4, 0.6), (0.6, 0.4), (0.8, 0.2), (0.9, 0.1)).

The below images are the class distribution after the shuffle and split:

Train set and test set for (0.4, 0.6) ratio:



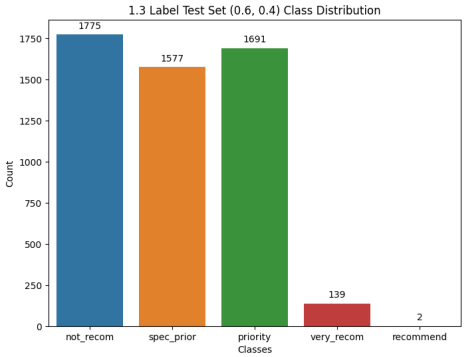
(a) Train set

(b) Test set

Figure 2: Ratio (0.4, 0.6)

Train set and test set for (0.6, 0.4) ratio:

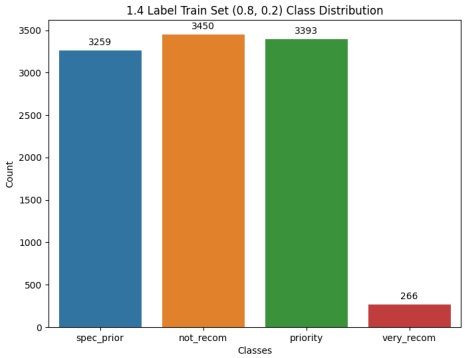
(a) Train set



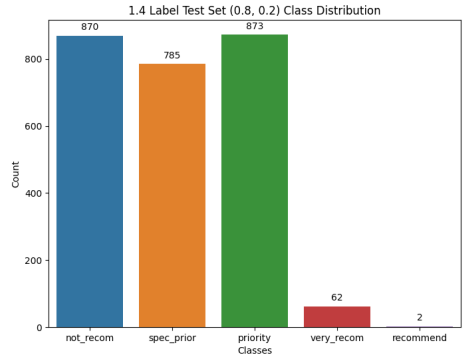
(b) Test set

Figure 3: Ratio (0.6, 0.4)

Train set and test set for (0.8, 0.2) ratio:



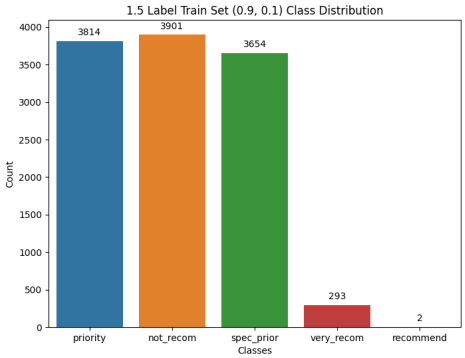
(a) Train set



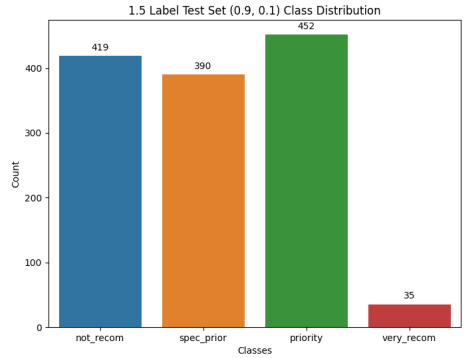
(b) Test set

Figure 4: Ratio (0.8, 0.2)

Train set and test set for (0.9, 0.1) ratio:



(a) Train set



(b) Test set

Figure 5: Ratio (0.9, 0.1)

5 Building the decision tree classifiers

The function `train_decision_tree(X_train, y_train, max_depth=None)` will be used to create train decision tree. We start training the decision tree by using the built in function `tree.DecisionTreeClassifier`. In parameter criterion, input `criterion="entropy"` to train the decision tree based on Information Gain. Because the data is not in numeric, the feature train sets need to be encoded first, then use `fit(X_train_encoded, y_train)` to start the training process.

There will be 4 decision trees created after the training. Because the complexity of these decision trees, clearer and more details images will be included in "Img" folder or in the ipynb file.

Figure 6: Decision Tree (0.4, 0.6)

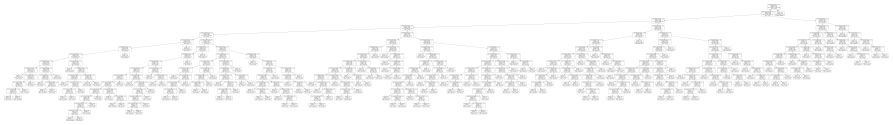


Figure 7: Decision Tree (0.6, 0.4)

Figure 8: Decision Tree (0.8, 0.2)

Figure 9: Decision$_{T}ree(0.9, 0.1)$

6 Evaluating the decision tree classifiers

The confusion matrix is a represent of the actual data turn out vs how the model predict the data. The classification report is the calculated values taken from the data in the confusion matrix.

Precision is the percentage of true positive value in all the predicted positive value.

Recall is the percentage of true positive value of actual positive value.

F1 score is a combination of both Precision and Recall. This value is only reached 1 when both Precision and Recall reached 1. If either Precision or Recall is lower than the other, the F1-score will tilt to the lower one.

Support represents the number of occurrence of all classes in the sets.

Macro and Weighted Average: These metrics provide an overall evaluation of the model's performance across all classes.

6.1 Ratio (0.4, 0.6)

```
Classification report: 40/60:
              precision    recall  f1-score   support

   not_recom       1.00      1.00      1.00      2603
```

priority	0.98	0.98	0.98	2565
recommend	1.00	0.00	0.00	2
spec_prior	0.98	0.98	0.98	2393
very_recom	0.96	0.98	0.97	213
accuracy			0.99	7776
macro avg	0.98	0.79	0.79	7776
weighted avg	0.99	0.99	0.98	7776

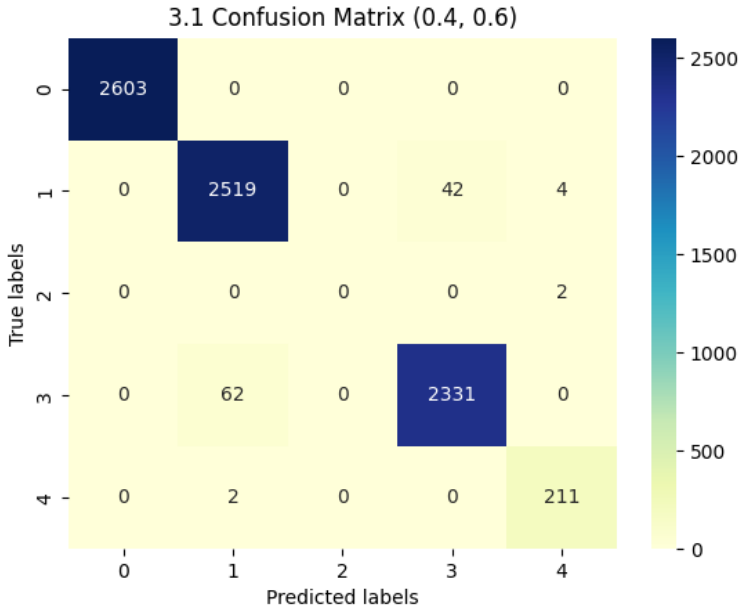


Figure 10: Confusion matrix ratio 40/60

The 40/60 ratio decision tree prediction is quite correct with only some error. In the Recommend class, because of the low instances, this decision tree is only able to predict all the Recommend class but fail at identifying any of the class.

6.2 Ratio (0.6, 0.4)

Classification report: 60/40:

	precision	recall	f1-score	support
not_recom	1.00	1.00	1.00	1775
priority	0.98	0.99	0.99	1691
recommend	1.00	0.00	0.00	2
spec_prior	0.99	0.98	0.99	1577
very_recom	0.99	0.99	0.99	139
accuracy			0.99	5184
macro avg	0.99	0.79	0.79	5184
weighted avg	0.99	0.99	0.99	5184

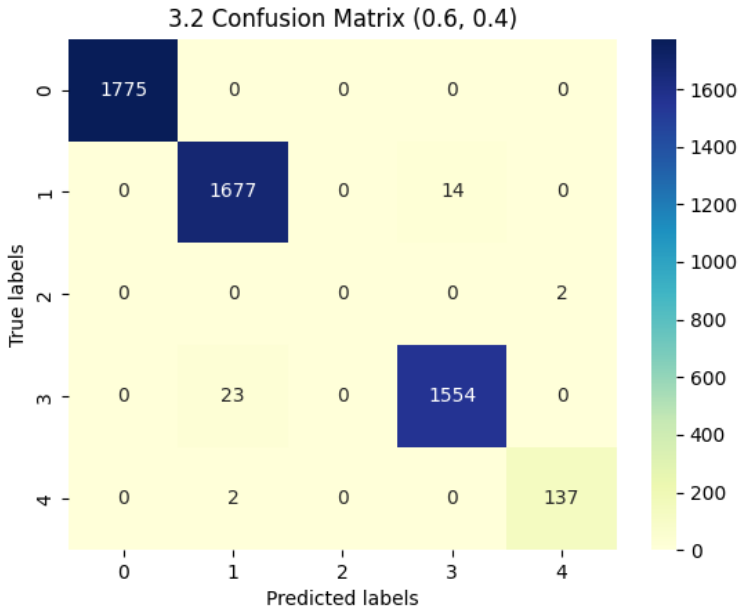


Figure 11: Confusion matrix ratio 60/40

These results are quite similar to the 40/60 decision tree results. With all the stats have high performance expect for the recommend class stats because of poor data pool.

6.3 Ratio (0.8, 0.2)

```
Classification report: 80/20:
              precision    recall  f1-score   support

   not_recom       1.00      1.00      1.00       870
    priority       1.00      0.99      0.99       873
   recommend       1.00      0.00      0.00         2
  spec_prior       0.99      0.99      0.99       785
  very_recom       0.94      1.00      0.97        62

    accuracy                           0.99      2592
   macro avg       0.98      0.80      0.79      2592
weighted avg       0.99      0.99      0.99      2592
```

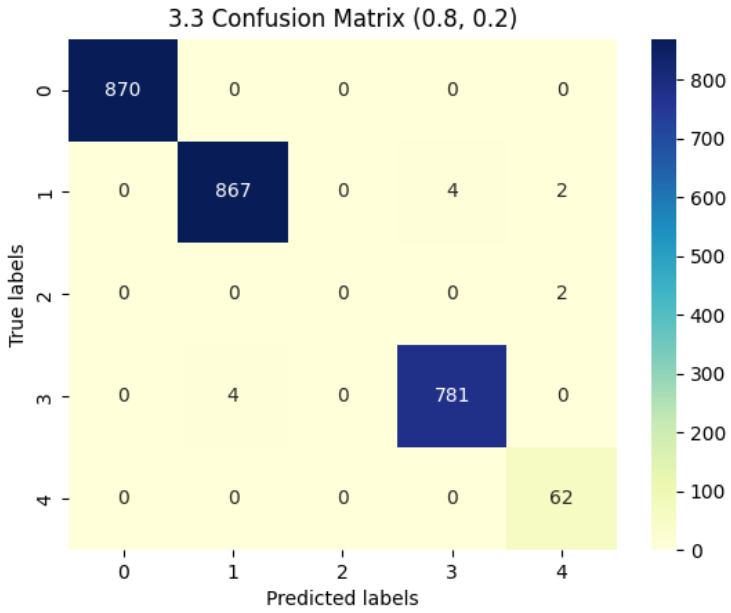


Figure 12: Confusion matrix ratio 80/20

Most of the stat performing great. Class very_recommend precision only at 0.94, this means that there are some prediction made by the decision tree are not true positive. But the recall stat, which is 1.00, shows that the model can easily identify all the very_recommend class in the set.

6.4 Ratio (0.9, 0.1)

```
Classification report: 90/10:
              precision    recall  f1-score   support

   not_recom       1.00      1.00      1.00       419
    priority       0.99      1.00      1.00       452
  spec_prior       1.00      0.99      0.99       390
  very_recom       1.00      1.00      1.00        35

    accuracy                           1.00      1296
   macro avg       1.00      1.00      1.00      1296
weighted avg       1.00      1.00      1.00      1296
```

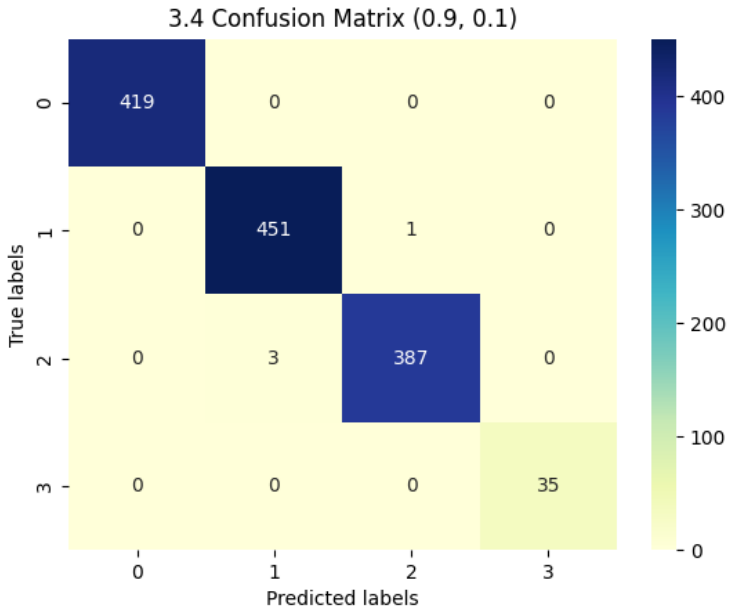


Figure 13: Confusion matrix ratio 90/10

In the test set there is no recommend class, so the class is missing from the matrix and report. Overall, we can see that the prediction and identification the decision tree made is quite accurate.

Ratio 60/40 and 90/10 seem to give the best performance out of any ratio.

7 The depth and accuracy of a decision tree

In this section, we create a list of value that stores all the desired depth. Then we put the depth into the `train_decision_tree(X_train, y_train, max_depth=None)` function to create decision tree.

max_depth	None	2	3	4	5	6	7
Accuracy	0.995	0.770	0.816	0.844	0.873	0.882	0.913

Table 1: Table of the accuracy score

The following images are decision trees corresponding to the max depth:

Figure 14: Decision Tree - Max depth: None

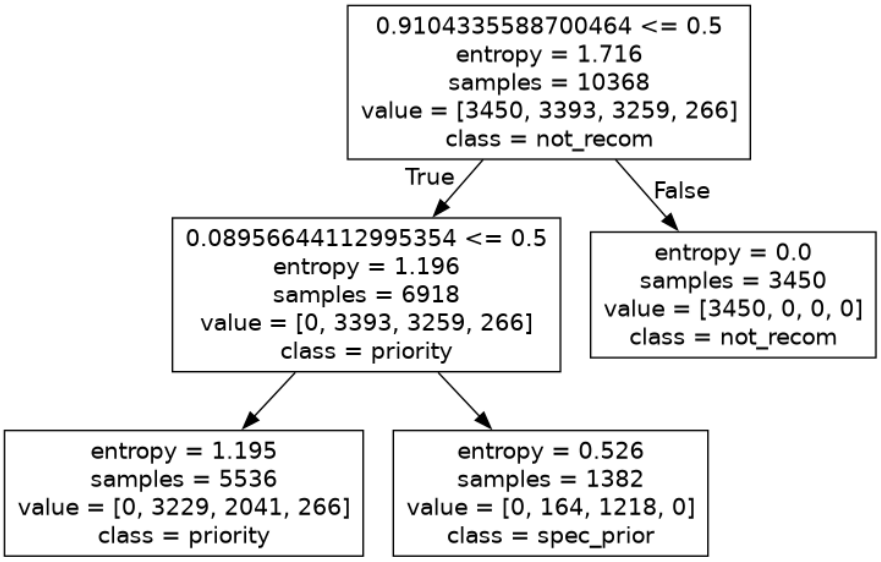


Figure 15: Decision Tree - Max depth: 2

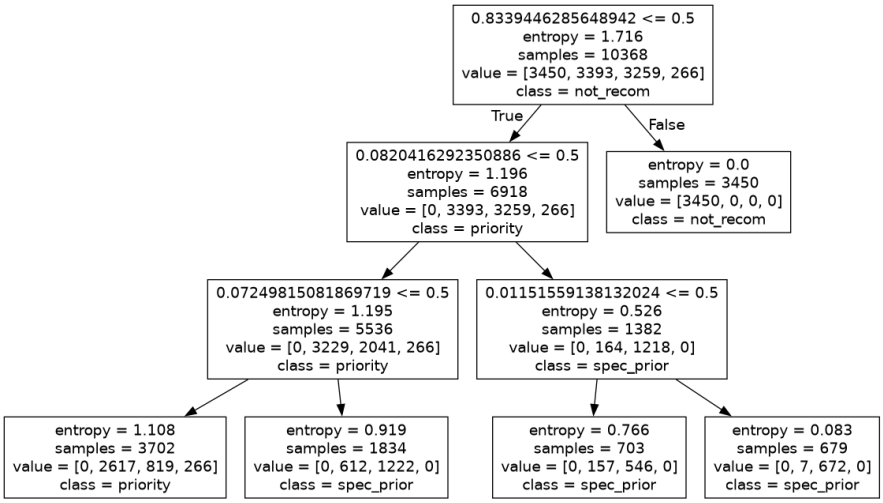


Figure 16: Decision Tree - Max depth: 3

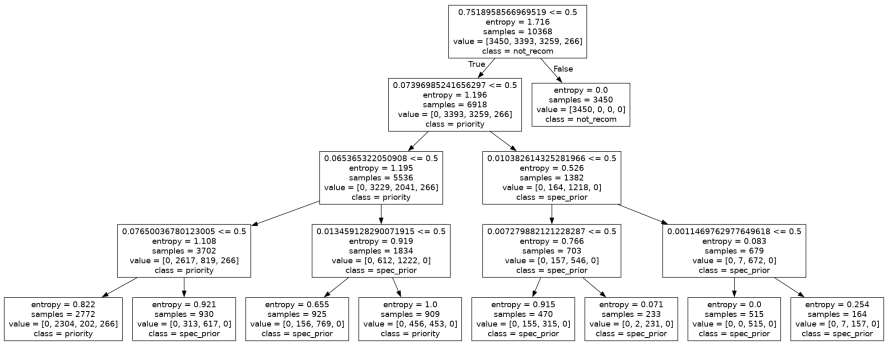


Figure 17: Decision Tree - Max depth: 4

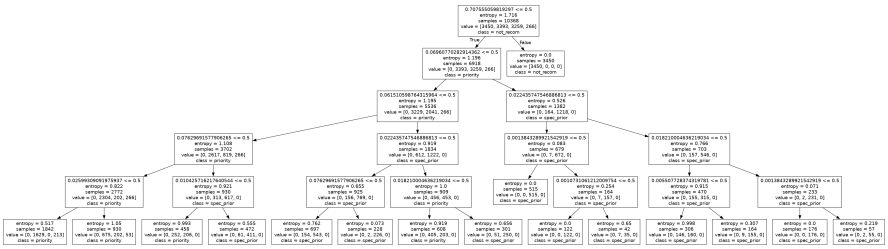


Figure 18: Decision Tree - Max depth: 5

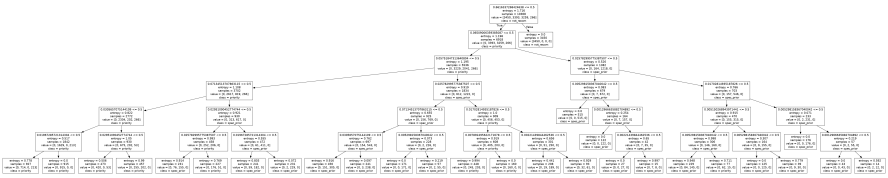


Figure 19: Decision Tree - Max depth: 6

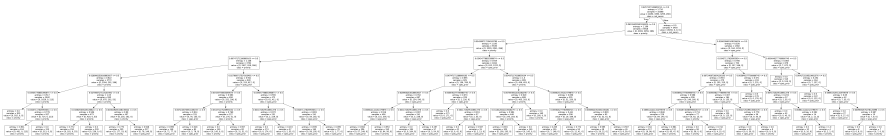


Figure 20: Decision Tree - Max depth: 7

7.1 Comments

On the same dataset, with different depth result in different accuracy for the model. As we allow the decision tree to branch more and deeper, the accuracy of the model increase, each outcome is a new decision tree learning problem with fewer examples and one less attribute.