UNIVERSITY OF SCIENCE
FALCUTY OF INFORMATION TECHNOLOGY

COURSE: INTRODUCTION TO ARTIFICIAL INTELIGENCE

LAB 02: DECISION TREE WITH SCIKIT-LEARN
CLASS: 20CLC11

Student name: Mai Quý Trung
Student ID: 20127370

Lecturers: **NGUYỄN NGỌC THẢO, LÊ NGỌC THÀNH, HỒ THỊ THANH TUYẾN**

fit@hcmus
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG - HCM
KHOA CÔNG NGHỆ THÔNG TIN

Table of Contents

I) Check List

No	Specifications	Done	Not Done
1	Preparing the datasets	X	
2	Building the decision tree classifiers	X	
3	Evaluating the decision tree classifiers	X	
	Classification report and confusion matrix	X	
	Comments	X	
4	The depth and accuracy of a decision tree	X	
	Trees, tables, and charts	X	
	Comments	X	
	Total	X	

II) Tasks Overview

- **Source Code**: SOURCE folder (main.py and connect-4.data)
- In this Lab, we will use 100% Python3 for all of the tasks. You can turn on the Terminal and type "*python3 --version*" to check your version of Python (any version above Python 3.9 is accepted)
- **Library**: Besides that, we also need to download some compulsory libraries to your computer (if you're using Google Colab, that's fine):
  + *sklearn*: It is a common Python library using for machine-learning. Type "pip install sklearn" on Terminal to download the library.
  + *graphviz*: It is a graph visualization library. You also need to download it to your computer if you're not using Google Colab. For Mac users, download homebrew and then type "brew install graphviz" on Terminal.
  + *matplotlib*: It is a mathematical library used to visualize data.
- **Data:** We will mainly work on a csv-alike file called "*connect-4.data" which can be downloaded at Connect-4. The dataset has 67557 lines, which is 67557 data samples, with 43 columns. The first 42 columns is the attributes of a connect-4 board description, and the last column is the tag/label of the board (win, loss or draw). We will need to shuffle and split this dataset into smaller subsets and do the following lab tasks to reach Lab Goal.*
- **Lab Goal:** We are going to shuffle and split the dataset into smaller subsets, in order to train and test the dataset. There are several proportions between train/test for us to work on. Based on the above subsets, we need to build a decision tree with the support from sklearn library and then visualize the

decision tree graph using graphviz, a Python visualization library for graph. Furthermore, we need to present a classification report and a confusion matrix from the predictions of the test subsets

III) Source Code & Details

1. Preparing the datasets

- For this task, I created 2 arrays, *feature* and *label* array, where feature is a 2D array having 67557 rows (boards) and 42 columns which is the 42 attributes of the connect-4 board, and label is a 1D array having 67557 elements which is the label/result of the boards (win, loss, draw).
- Since the dataset file "*connect-4.data*" is a csv-alike file, for each line, I used .split function to seperate elements from the commas (,) in order to get the attributes/result only.
- Furthermore, I also wanted to change the data from characters to numbers since sklearn and graphviz library can only train the data with numbers. So, I represented the attributes of the board: 'b' as 0, 'x' as 1, 'o' as -1 and label/result of the game: 'win' as 1, 'loss' as -1, 'draw' as 0 as convention.
- After changing characters to numbers and appending them to the arrays, feature and label are returned.
- Next step, we wanted to shuffle the dataset and then split the dataset into 4 types of subsets: *feature_train , label_train, feature_test, label_test* with 4 different proportions of train/test: 4/6, 6/4, 8/2 and 9/1. Therefore, we needed 16 subsets in total.
- From *sklearn.model_selection*, we used *.train_test_split(X,Y)* function where X is feature and Y is label. Ther are also some parameters that I parsed in such as *train_size* is the train/data proportion, *shuffle=True* and *stratify=label*, which is the target names (win/loss/draw).

2. Building the decision tree classifiers

- After preparing the dataset and splitting them into 16 smaller subsets, we wanted to build the decision tree classifiers using *DecisionTreeClassifier* function from *sklearn.tree* library.
- Secondly, since the task requires information gain from the tree, in the *DecisionTreeClassifier* function's parameters, I set the *criterion='entropy'* and *max_depth=None* (this argument will be mentioned later) since *'entropy'* is the information gain.

- After that, we use *.fit(X,Y)* function where X is *feature_train* and Y is *label_train*, since I wanted to train the data from the given feature and label subset, therfore, it will create the decision tree based on the input train data.
- After constructing the decision tree, I used the *export_graphviz* function from *graphviz* library to visualize the decision tree as a pdf file, go along with *.Source* and *.render* function to render the graph into output file, which took quite amount of time (~approx 5 mins/graph)

3. Evaluating the decision tree classfiers

- Next up, I created a classification report and a confusion matrix from the decision tree classifiers using a *predict_test* array, which is *.predict* function taking input argument is the tree classifiers to predict from the *feature_test*.
- After that, we will use *.classification_report* and *.confusion_matrix* function to compare the predict_test with the label_test, since predict_test is the array predicted from the tree classifiers using feature_test as input.
- Finally, I visualized the confusion matrix by using *ConfusionMatrixDisplay* from *sklearn.metrics* library, taking the predictions as input and use *.suptitle* function to print out a visualization picture as output of the respective confusion matrix from respective *predict_test* and *label_test.*

4. The depth and accuracy of a decision tree

max_depth	None	2	3	4	5	6	7
Accuracy	0.7733	0.6583	0.6675	0.6751	0.6852	0.6965	0.7038

Comment:

- The max_depth of the decision tree is directly proportional to the accuracy, which means the more deeper of the decision tree, the more accurate of the prediction accuracy.
- From the table we can see that at max_depth=None, the accuracy reach highest at approximately 77,33 %.
- Meanwhile, the accuracy at max_depth=7 is only 70,38 % and at max_depth=2 the accuracy is lowest at only 65,83%

IV) Experiment & Statistics

1. Decision tree

- The decision tree description and graph visualization is include in the SOURCE folder.

- The data files and graphs includes decision tree with train/test proportion of [4/6, 6/4, 8/2, 9/1] respectively. Furthermore, at 8/2 train/test proportion, we also have 6 more graphs at max_depth from 2 to 7

Comment:

- With train/test proportion at 4/6, the decision tree visualization graph has the least components compared to proportion 6/4 or 8/2 or 9/1. Therefore, it has less data to train and classify leading to have a lower accuracy rate of decision tree than the other 3.
- On the other hand, train/test proportion at 9/1 has the most tree branches and nodes, therefore, it has higher accuracy rate.

2. Classification report

- At 4/6 train/test proportion: According to the classification report, the decision tree classifier has a medium accuracy rate prediction since it has less data to train than data to test. Therefore, its macro average is only 0.59 and weighted average is 0.75

```
              precision    recall  f1-score   support

          -1       0.66      0.65      0.65      9981
           0       0.27      0.27      0.27      3870
           1       0.85      0.85      0.85     26684

    accuracy                           0.75     40535
   macro avg       0.59      0.59      0.59     40535
weighted avg       0.75      0.75      0.75     40535

[[ 6493  1088  2400]
 [ 1084  1056  1730]
 [ 2274  1741 22669]]

Accuracy at max_depth = None:  0.7454792154927841

Confusion matrix:
[[ 6493  1088  2400]
 [ 1084  1056  1730]
 [ 2274  1741 22669]]
```

(Train/test proportion of 4/6 classification report and accuracy)

- At 6/4 train/test proportion: It has a more decent proportion between train and test where train ratio is a little bit higher. Therefore, clearly that we can see from the classification report and the confusion matrix, the macro average is raised to 0.62 and weighted average is raised to 0.77, a little bit higher.

```
              precision    recall  f1-score   support

          -1       0.69      0.69      0.69      6654
           0       0.30      0.29      0.30      2580
           1       0.86      0.87      0.86     17789

    accuracy                           0.77     27023
   macro avg       0.62      0.61      0.62     27023
weighted avg       0.77      0.77      0.77     27023

[[ 4566   708  1380]
 [  709   749  1122]
 [ 1315  1035 15439]]

Accuracy at max_depth = None:  0.7680124338526441

Confusion matrix:
[[ 4566   708  1380]
 [  709   749  1122]
 [ 1315  1035 15439]]
```

(Train/test proportion of 6/4 classification report and accuracy)

- At 8/2 train/test proportion: It remains the same macro average and weighted average compared to the 6/4 train/test ratio proportion. And as we can predict, the accuracy of max_depth=2 is the lowest with only 65,83% and max_depth = None reach highest with 77,33% at 8/2 train/test proportion.

```
              precision    recall  f1-score   support

          -1       0.71      0.69      0.70      3327
           0       0.29      0.29      0.29      1290
           1       0.86      0.87      0.87      8895

    accuracy                           0.77     13512
   macro avg       0.62      0.62      0.62     13512
weighted avg       0.77      0.77      0.77     13512

[[2311  378  638]
 [ 325  378  587]
 [ 602  533 7760]]

Accuracy at max_depth = None:  0.7733126110124334

Confusion matrix:
[[2311  378  638]
 [ 325  378  587]
 [ 602  533 7760]]

Accuracy at max_depth =  2  :  0.658303730017762


Accuracy at max_depth =  3  :  0.6674807578448786


Accuracy at max_depth =  4  :  0.6750296033155714


Accuracy at max_depth =  5  :  0.6851687388987566


Accuracy at max_depth =  6  :  0.6964920071047958


Accuracy at max_depth =  7  :  0.7038188277087034
```

(Train/test proportion of 8/2 classification report and accuracy)

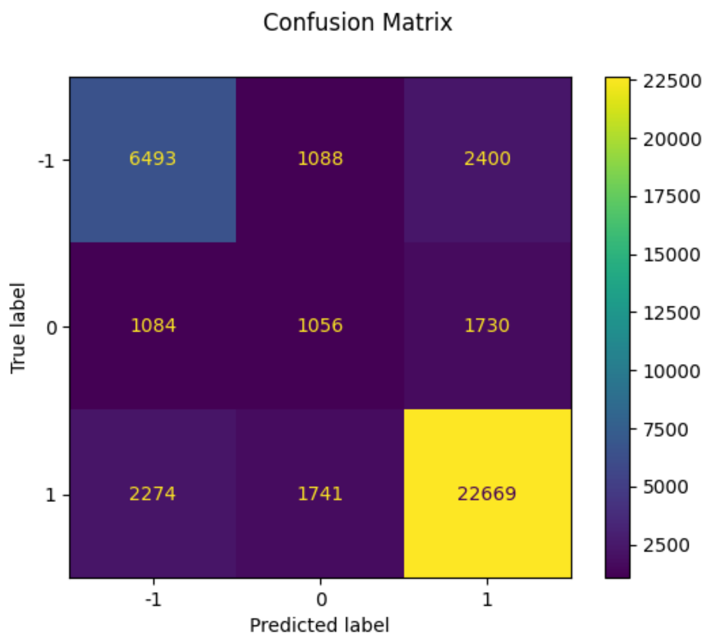
- At 9/1 train/test proportion: From the classification report and confusion matrix, the macro average is 0.61 and weighted average is 0.77

```
              precision    recall  f1-score   support

          -1       0.69      0.68      0.69      1664
           0       0.27      0.28      0.28       645
           1       0.87      0.87      0.87      4447

    accuracy                           0.77      6756
   macro avg       0.61      0.61      0.61      6756
weighted avg       0.77      0.77      0.77      6756

[[1139  220  305]
 [ 182  180  283]
 [ 326  255 3866]]

Accuracy at max_depth = None:  0.7674659561870929

Confusion matrix:
[[1139  220  305]
 [ 182  180  283]
 [ 326  255 3866]]
```

(Train/test proportion of 9/1 classification report and accuracy)

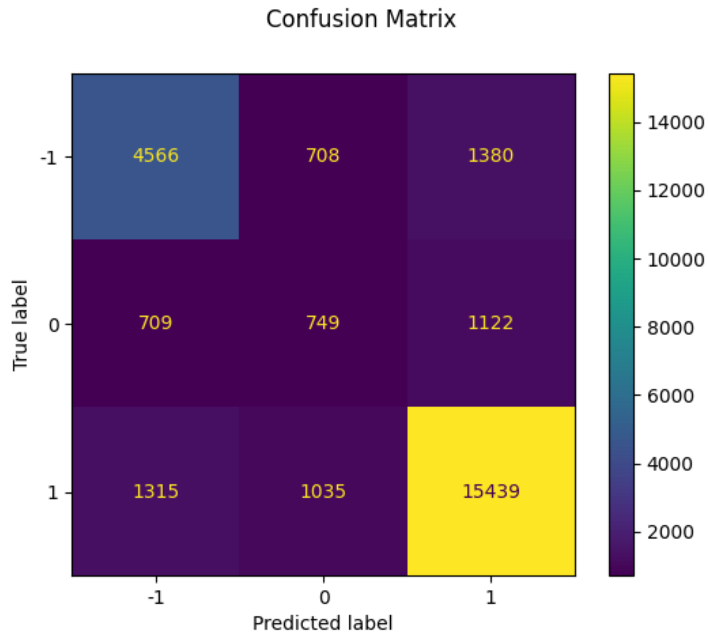
3. Confusion matrix

- At 4/6 train/test proportion:



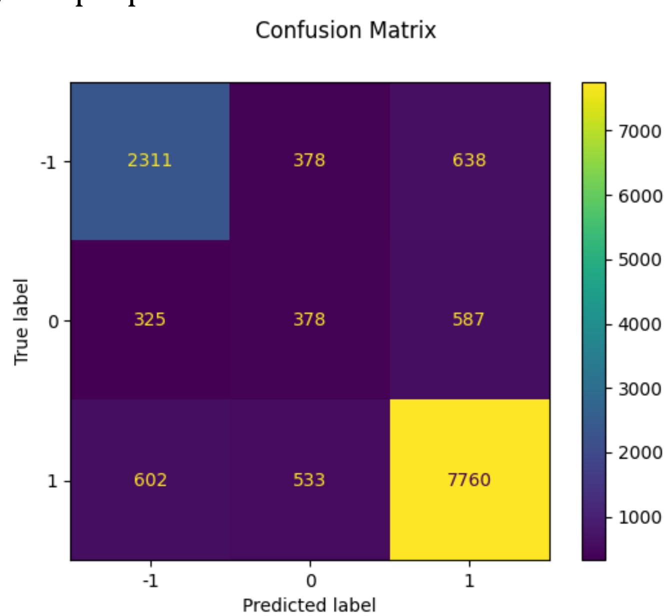
(Train/test proportion of 4/6 confusion matrix)

- At 6/4 train/test proportion:



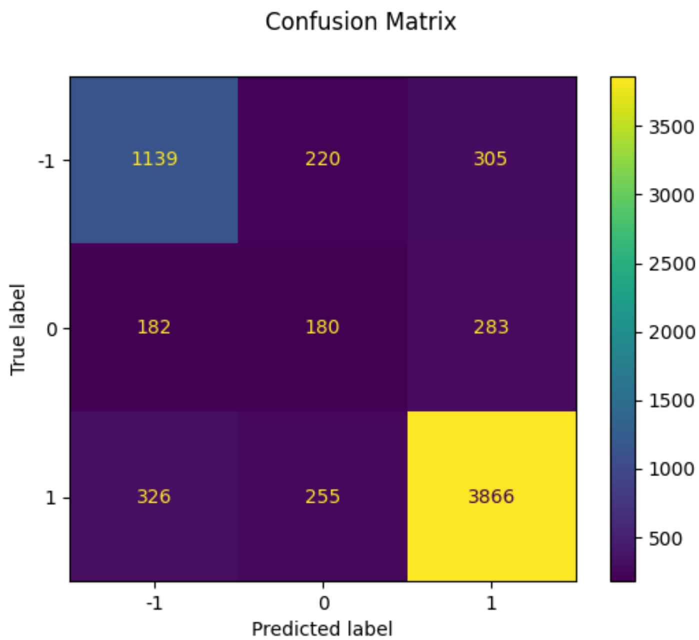
(Train/test proportion of 4/6 confusion matrix)

- At 8/2 train/test proportion:



(Train/test proportion of 4/6 confusion matrix)

- At 9/1 train/test proportion:



(Train/test proportion of 4/6 confusion matrix)

V) References

- **Scikit-learn decision trees:** scikit-learn.org/tree
- **Analysis and classification of Mushrooms:** kaggle.com/analysis-and-classification-of-mushrooms
- **Scikit-learn DecisionTreeClassifier:** scikit-learn.org/DecisionTreeClassifier
- **Scikit-learn classification_report:** scikit-learn.org/classification_report
- **Scikit-learn confusion_matrix:** scikit-learn.org/confusion_matrix