VIETNAM NATIONAL UNIVERSITY
HO CHI MINH UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

LAB 02: DECISION TREE

Course: Artificial Intelligence

Student:

Hoàng Bảo Khanh (22127183)

Lecturers:

PhD Nguyễn Ngọc Thảo

M.S Lê Ngọc Thành

M.S Nguyễn Hải Đăng

M.S Nguyễn Trần Duy Minh

Ho Chi Minh city, April 16th, 2024

CONTENTS:

I. PROJECT DESCRIPTION

In this Lab, I am going to build a decision tree on the UCI Nursery Data Set, with support from the scikit-learn library.

The link of the dataset I downloaded is here: link dataset.

To implement this Lab, I used Python programming language and used scikit-learn functions to complete the tasks.

In total, there are many significant tasks that I had to complete this Lab:

- Preparing the data sets
- Building the decision tree classifiers
- Evaluating the decision tree classifiers
- The depth and accuracy of a decision tree

II. *PREPARING* THE DATA SETS

1. *Reading the file:*

According to the requirements, initially, I must add the ".csv" extension to the file "nursery.data" to read the file. In order to read the file, I used "pandas" library.

```
#·Read·file
data·=·pd.read_csv("nursery/nursery.data.csv",·header=None)
```

Image 1: Reading file to object "data".

2. *Splitting data sets:*

Before splitting data, I shuffled the data and split it into two strategies: features and labels. Features are the values of the first seven columns, labels are the the values in the last column.

```
#Separate·features·and·labels
X·=·data.iloc[:,·:-1]·#·All·columms·except·the·last·one
y·=·data.iloc[:,·-1]·#·The·last·column
```

Image 2: Splitting data into 2 strategies.

Subsequently, I split the data sets into four subsets:

- feature_train: a set of training examples, each of which is a tuple of 8 attribute values (target attribute excluded).
- label_train: a set of labels corresponding to the examples in feature_train.
- feature_test: a set of test examples, it is of similar structure of feature_train.
- label_test: a set of labels corresponding to the examples In feature_test.

To create four subsets, I used train_test_split function. To implement this function, we designed the splits_proportions list which includes train/test ratio: 40/60, 60/40, 80/20, 90/10.

```python
splits_proportions = [(0.4, 0.6), (0.6, 0.4), (0.8, 0.2), (0.9, 0.1)]

feature_train_sets, feature_test_sets, label_train_sets, label_test_sets = [], [], [], []

def splitData():
    for split in splits_proportions:
        X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=split[0],
                                                            test_size=split[1], random_state=None, stratify=y)

        feature_train_sets.append(X_train)
        feature_test_sets.append(X_test)
        label_train_sets.append(y_train)
        label_test_sets.append(y_test)
```

Image 3: Splitting into four subsets.

3. Visualize the distributions of classes:

The program will visualize 3 main data sets: Original sets, Training sets and Testing sets by using bar charts.

To complete it, the program uses matplotlib library and there are 9 charts shown in the output (Original sets, 4 Training sets and 4 Testing sets).

- Original sets:

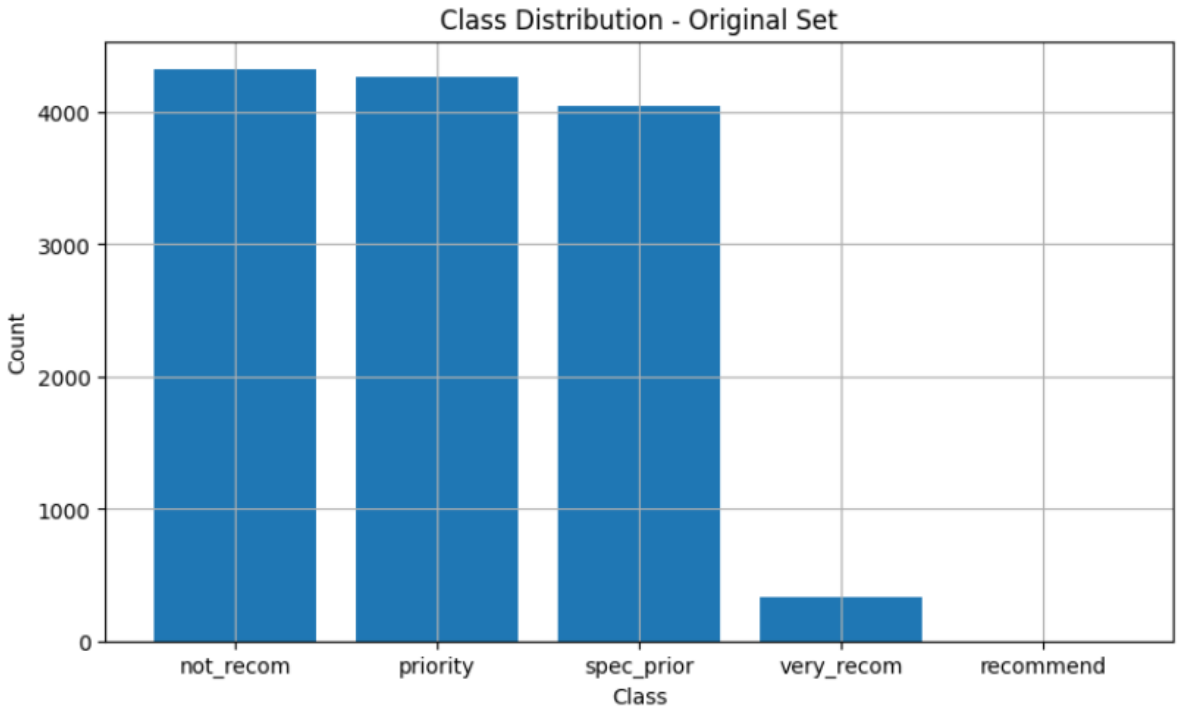

Image 4: The bar chart of Original set

- Training sets:

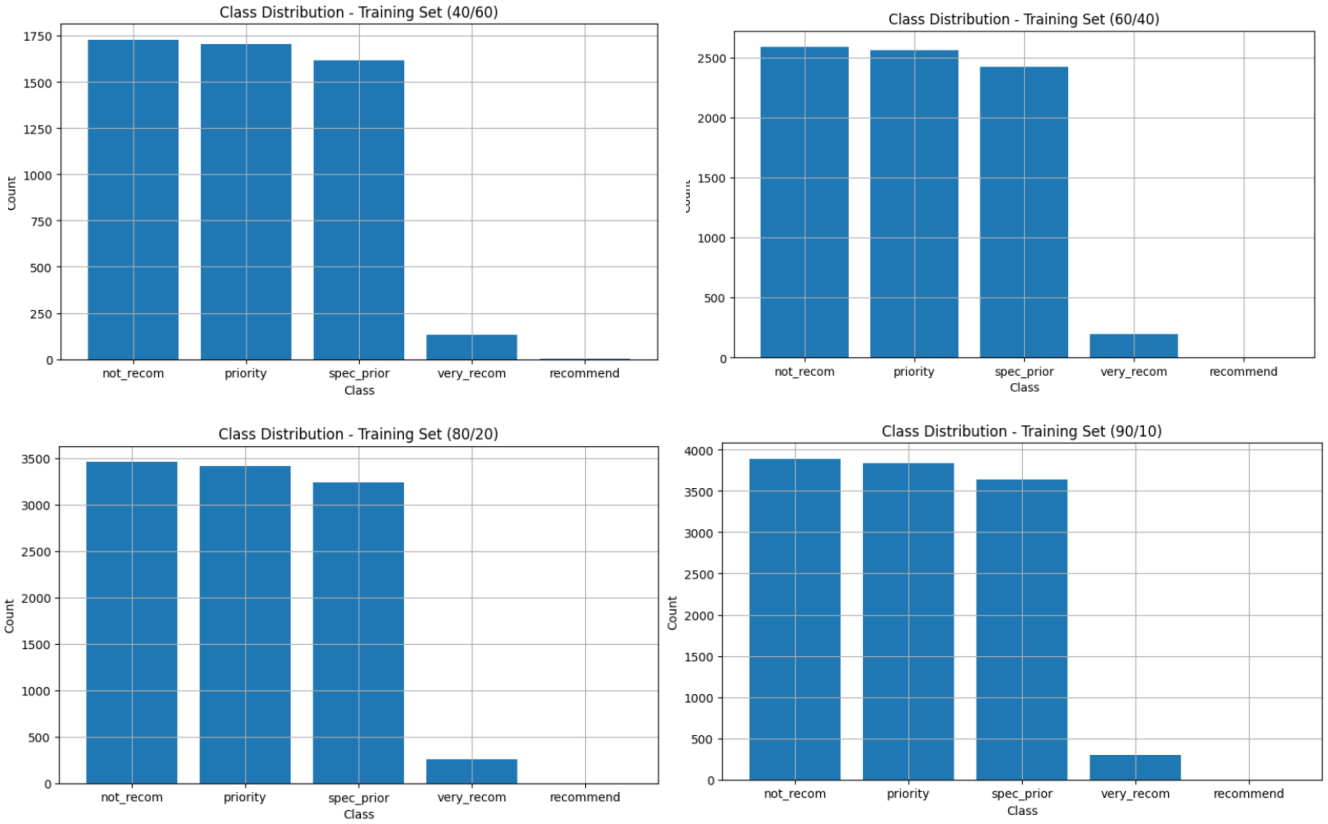

Image 5: The bar chart of Training sets

- Testing sets:

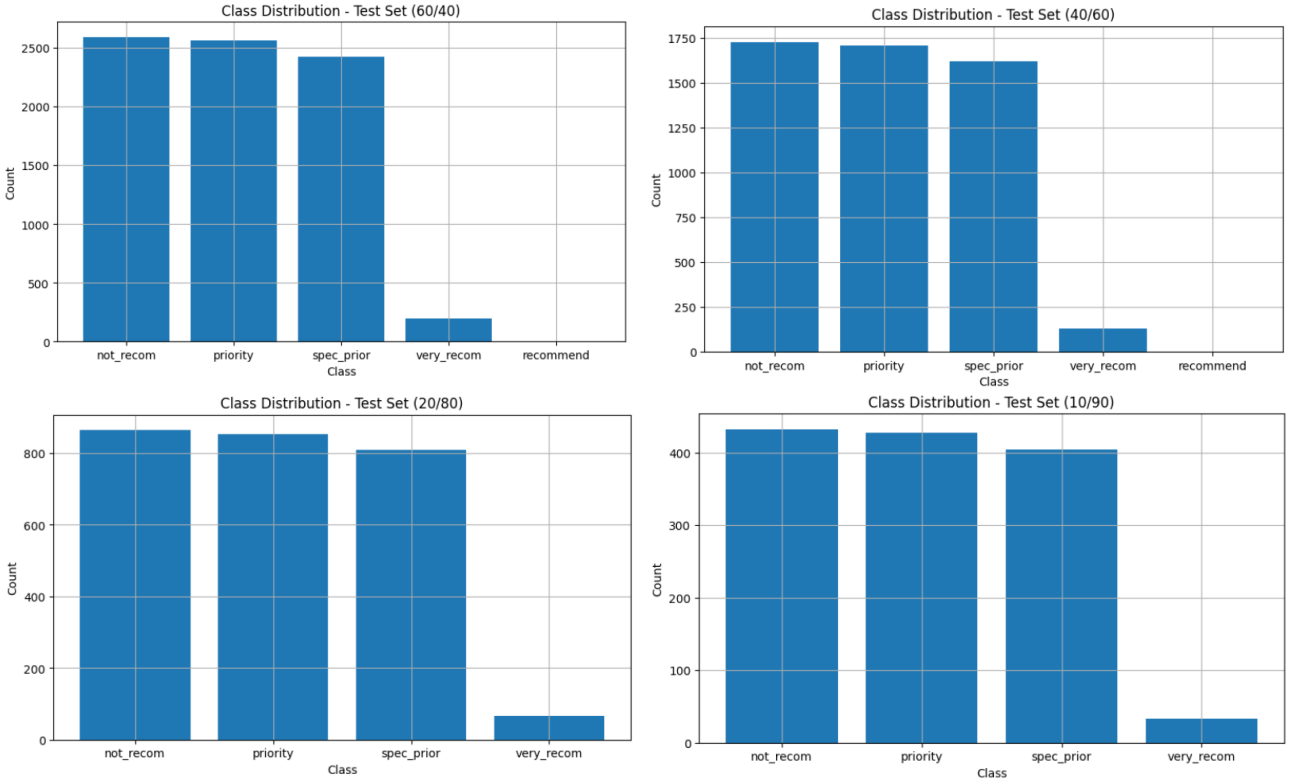

Image 6: The bar chart of Testing sets

III. BUILDING THE DECISION TREE CLASSIFIERS

In purpose of bulding the decision trees classifiers, I fit an instance of sklearn.tree.DecisionTreeClassifier to each training set with information gain (using entropy).

I used the fit function to train the training data sets. Then, to visualize the decision tree, I installed the graphviz library through this link: link download.

The decision trees will be saved as .pdf file and in the folder named Decision_Tree. In the folder, there are 4 files for 4 decision trees with 4 subsets I had split above.

IV. EVALUATING THE DECISION TREE CLASSIFIERS

In this task, the program will predict the examples in the corresponding test sets. And then, the program creates the classification report and confusion matrix.

1. Classification report

The program used sklearn.metrics.classification_report to implement this task.

The classification report illustrates the information of labels in the decision tree. The report has four main strategies: precision, recall, f1-score and support.

- precision: it can be seen as a measure of a classifier's exactness. It is defined as the ratio of true positives to the sum of true and false positives.
- recall: it is a measure of the classifier's completeness, the ability of a classifier to correctly find all positive instances.
- f1-score: is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0.
- support: is the number of actual occurences of the class in the specified dataset.

2. Confusion matrix

The function sklearn.metrics.confusion_matrix is used for this task. The confusion matrix demonstrates the numbers of labels in the matrix and the accuracy of a classification.

Moreover, to display the matrix, I used ConfusionMatrixDisplay and used “plt” to design the size.

To get the color of the matrix, I used Matplotlib library and according to the colormap in Matplotlib, I chose the color code “YlOrRd” to implement the confusion matrix.

3. Comments of the performance:

a. Train / Test: 40/60

Classification report 40/60:

	precision	recall	f1-score	support
not_recom	1.00	1.00	1.00	2592
priority	0.98	0.98	0.98	2560
recommend	0.00	0.00	0.00	1
spec_prior	0.98	0.98	0.98	2426
very_recom	0.91	0.94	0.93	197
accuracy			0.99	7776
macro avg	0.77	0.78	0.78	7776
weighted avg	0.99	0.99	0.99	7776

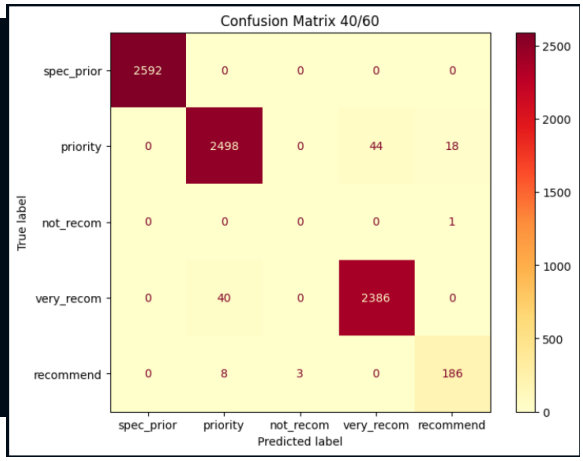

Image 7: Classification report and confusion matrix, ratio is 40/60

From the report, it is clear that the number of class “not_recom” is the biggest, which is 2592, and the class having the smallest number is “recommend”.

The precisions of 5 classes are considerably high except recommend. They are almost over 0.9. However, the number of class “recommend” c is very small, so the precision is 0.

The recall scores and f1-scores are also high, indicating a good balance between precision and recall.

b. Train / Test: 60/40

Classification report 60/40:

	precision	recall	f1-score	support
not_recom	1.00	1.00	1.00	1728
priority	0.99	0.99	0.99	1706
recommend	1.00	1.00	1.00	1
spec_prior	0.99	0.99	0.99	1618
very_recom	0.98	0.98	0.98	131
accuracy			0.99	5184
macro avg	0.99	0.99	0.99	5184
weighted avg	0.99	0.99	0.99	5184

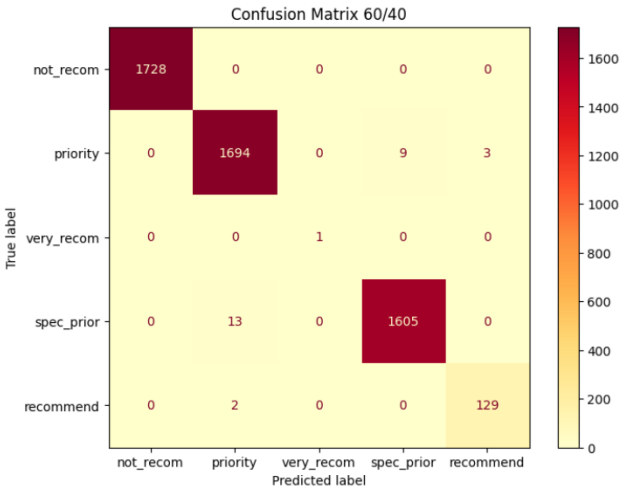

Image 8: The classification report, confusion matrix, ratio is 60/40

The number of testing objects is smaller than the ones in the report 40/60 above. The maximum number of these classes is 1728.

It is clear that the precision scores, recall scores and f1-scores are very high, which is higher than the classifier in the report 40/60.

c. Train / Test: 80/20

Classification report 80/20:

	precision	recall	f1-score	support
not_recom	1.00	1.00	1.00	864
priority	1.00	1.00	1.00	853
spec_prior	1.00	1.00	1.00	809
very_recom	1.00	1.00	1.00	66
accuracy			1.00	2592
macro avg	1.00	1.00	1.00	2592
weighted avg	1.00	1.00	1.00	2592

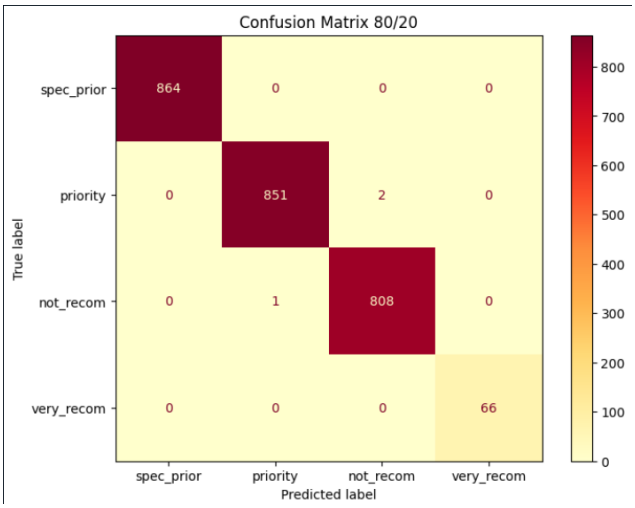

Image 9: The classfication report, confusion matrix, ratio 80/20

The number of classes in this ratio is smaller remarkably than the 2 reports above. However, the precision, recall and f1-scores reach absolutely the best score, whose value is 1.0.

d. Train / Test: 90/10

Classification report 90/10:

	precision	recall	f1-score	support
not_recom	1.00	1.00	1.00	432
priority	0.99	1.00	1.00	427
spec_prior	1.00	0.99	1.00	404
very_recom	1.00	1.00	1.00	33
accuracy			1.00	1296
macro avg	1.00	1.00	1.00	1296
weighted avg	1.00	1.00	1.00	1296

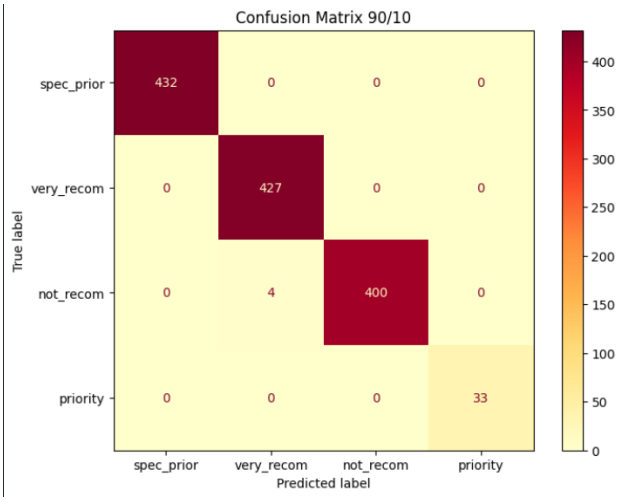

Image 10: The classification report, confusion matrix, ratio 90/10

Similarly, the number of classes in this case is the smallest. However, the scores are very high.

Conclusion:

- From 4 classification reports shown above, the classifiers in 80/20 and 90/10 are higher than the classifiers in 40/60 and 60/40.
- In general, the classifier in 80/20 ratio performs most significant in terms of precision, recall and f1-scores. Therefore, it demonstrates a higher ability to correctly classify samples.

V. THE DEPTH AND ACCURACY

This task works on the 80/20 training set and test set. At first, I initialize the list of max depth of the decision tree. The list of max depth includes None, 2, 3, 4, 5, 6, 7.

The function of creating decision tree is similar to the function when the program builds the decision tree using graphviz. In total, there are 7 decision trees with different depths and those trees are saved as pdf file in Depth folder.

Besides designing decision trees, the program also calculates the accuracy score after prediction. For each decision tree, there is a accuracy score and I collected it into the table below.

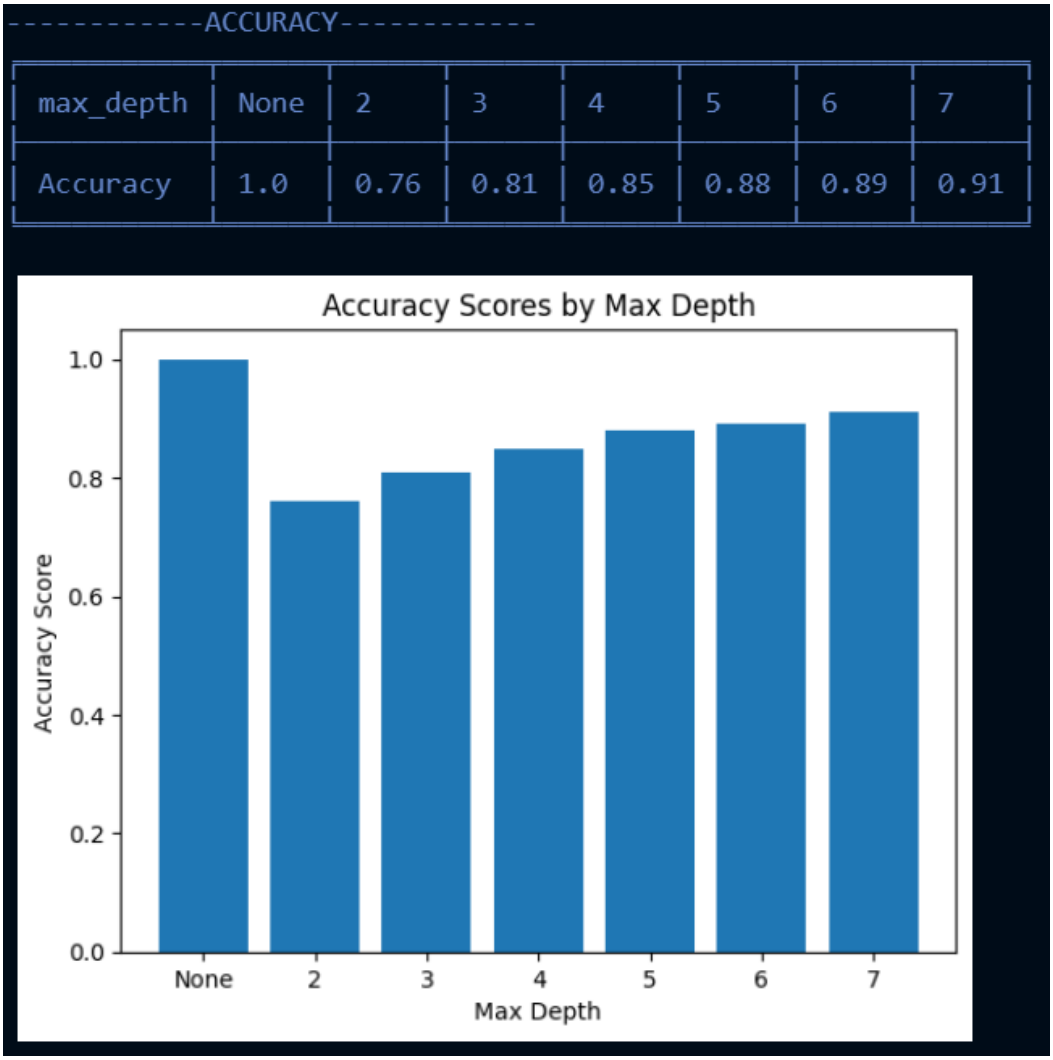

Image 11: The table and chart illustrate the accuracies of decision trees.

Comments of the performance:

- According to the table and chart of accuracy, it is clear that when the max depth increases, the accuracy of prediction rises significantly.
- When the max depth is None, the decision tree will expand as much as possible. Therefore, the accuracy of this case is approximately the best score.

Conclusion: The decision tree's depth affects the classification accuracy. The accuracy is higher when the decision tree's depth increases.

VI. REFERENCES

[1]. Decision Tree Classification in Python Tutorial, datacamp.

[2]. Python | Decision tree implementation, 07/12/2023, geeksforgeeks.org.

[3]. How to Create a Table with Matplotlib, 27/01/2022, geeksforgeeks.org.

[4]. Confusion matrix, scikit-learn.org.

[5]. Choosing Colormaps in Matplotlib, matplotlib.org.

[6]. Download graphviz, graphviz.org

[7]. Decision Tree, 20/08/2023, geeksforgeeks.org.