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

# LAB 2: DECISION TREE

## Course: CSC14003 - Introduction to Artificial Intelligence

| **Student** | **Instructor** |
|---|---|
| 22127121 - Đào Việt Hoàng | Mr. Nguyễn Trần Duy Minh |

# Contents

# I Self-evaluation

| No. | Specification | Eval. |
|---|---|---|
| 1 | Preparing the datasets | 100 % |
| 2 | Building the decision tree classifiers | 100 % |
| 3 | Evaluating the decision tree classifiers | |
| | Classification report and confusion matrix | 100 % |
| | Comments | 100 % |
| 4 | The depth and accuracy of a decision tree | |
| | Trees, tables, and charts | 100 % |
| | Comments | 100 % |

# II Labwork Details

## II.1 Preparing the dataset

### II.1.1 Libraries required

Libraries required for this project include:

- `pandas`: for data representation
- `scikit-learn`: for these purposes
  - `sklearn.model_selection`: for splitting data into train data and test data
  - `sklearn.tree`: for decision tree instances and visualization export
  - `sklearn.metrics`: for evaluation routines
- `matplotlib.pyplot`, `matplotlib.image` and `pydotplus`: for beautiful visualization on Jupyter notebook
- `ucirepo`: for dataset used in this labwork

These libraries can be installed through `pip` by running

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

### II.1.2 Splitting dataset

The dataset is copied 4 times. Each copy includes `feature_train`, `label_train`, `feature_test` and `label_test`, and will be split as follows:

- First copy: 60% for train, 40% for test

- Second copy: 40% for train, 60% for test
- Third copy: 80% for train, 20% for test
- Fourth copy: 90% for train, 10% for test

Each copy is saved in a Python dictionary. To extract the dataset of specific train-test proportion (for example, 60/40), do the following:

```
    X_train, y_train = feature_train['60/40'], label_train['60/40']
    X_test, y_test = feature_test['60/40'], label_test['60/40']
```

### II.1.3 `train_test_split`

The parameters of `train_test_split` is set as follows:

- `test_size`: The proportion of test dataset to the whole. If `test_size` is specified, `train_size` doesn't need to be specified.
- `shuffle`: Whether to shuffle the data; default value is `True`.
- `random_state`: Seed value to randomize the order of the dataset. If not specified, the order is different every time the notebook is executed; otherwise, the order is consistent for each seed given. While the author's seed number is 42, it can be set to any integer.
- `stratify`: How to split the data. If set to `label`, the proportion of each class in the original dataset is kept when splitting to train and test ones. For example, if the proportion of Benign/Malignant in the original set is 60/40, the proportion in either the train or test set is also 60/40.

## II.2 Building the decision tree classifiers

All the models are saved inside a dictionary `models`. To access a specific model (for example, 60/40), do the following:

```
    model = models['60/40']
```

Function `export_model_image(model, filename)` receives the model to be visualized and the file name to save the picture inside `images/decision_tree_classifier`. The image is then loaded and displayed on the notebook. Instead of `graphviz` as per instruction, the author decided to use `matplotlib` for familiarity and the ability to fit the image with the notebook.

## II.3 Evaluating the decision tree classifiers

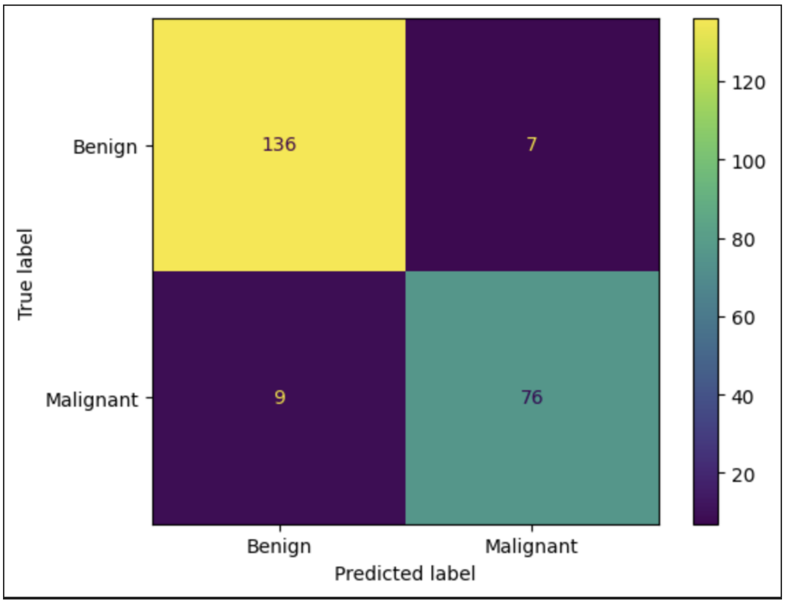
### II.3.1 Interpretation

`confusion_matrix`

In the confusion matrix $C$, the row represents the actual Benign or Malignant, while the column represents the prediction of the model.

- $C_{0,0}$: Both the truth and the prediction are Benign
- $C_{0,1}$: The truth is Benign, but the prediction is Malignant
- $C_{1,0}$: The truth is Malignant, but the prediction is Benign
- $C_{1,1}$: Both the truth and the prediction are Malignant

For example: Dataset of 60/40



Confusion Matrix on 60/40 Dataset

This confusion matrix means:

- There are 136 Benign cases that the model predicted correctly.
- There are 7 Benign cases that the model predicted wrong.
- There are 9 Malignant cases that the model predicted wrong.
- There are 76 Malignant cases that the model predicted correctly.

`classification_report`

The Precision, Recall and F1 Score can be calculated based on which type of class is chosen to be Positive. For example: if Malignant as Positive, Benign as Negative.

1. Precision: **What percentage of all the Positive predictions made by the model were accurate?** The formula to calculate Precision is:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

2. Recall: **What percentage of all the actual Positives were accurately predicted by the model?** The formula to calculate Recall is:

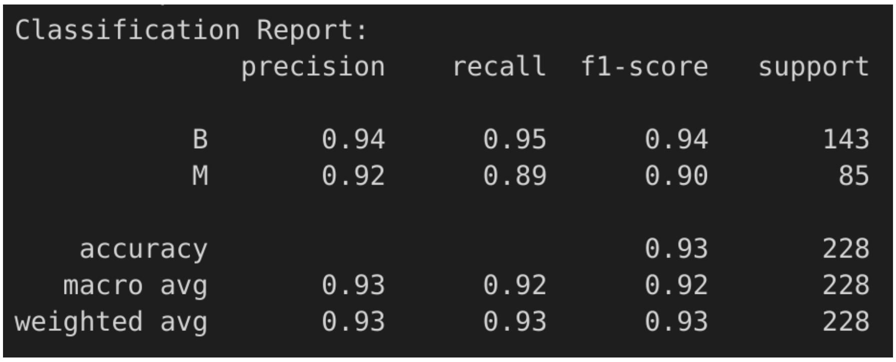
$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

3. F1 Score: **The harmonic mean of Precision and Recall.** If any of them becomes extremely low, F1 Score will also go down. Thus, F1 Score can help you find a good balance between Precision and Recall. The formula to calculate F1 Score is:

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

4. Support: **How many samples are in each class.** It uses the ground truth labels, which represent the actual class of each sample.

For example: Dataset of 60/40



```
Classification Report:
              precision    recall  f1-score   support

           B       0.94      0.95      0.94       143
           M       0.92      0.89      0.90        85

    accuracy                           0.93       228
   macro avg       0.93      0.92      0.92       228
weighted avg       0.93      0.93      0.93       228
```

Classification Report on 60/40 Dataset

The Malignant class is chosen to be Positive, while the Benign class is Negative. The statistics can be calculated as follows:

- Precision:
$$Precision = \frac{76}{76 + 7} = 0.92$$
- Recall:
$$Recall = \frac{76}{76 + 9} = 0.89$$
- F1 Score:
$$F1\ Score = \frac{2 \times 0.92 \times 0.89}{0.92 + 0.89} = 0.90$$

The Benign class is chosen to be Positive, while the Malignant class is Negative. The statistics can be calculated as follows:

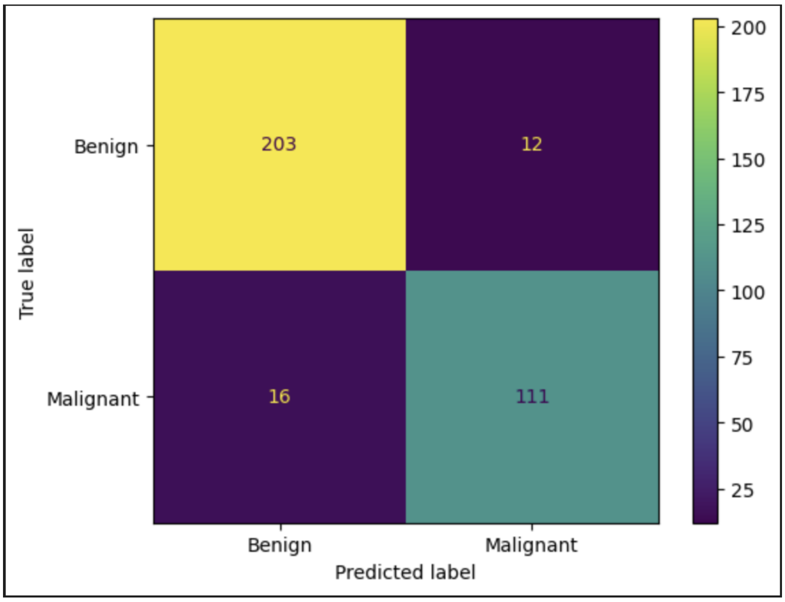
- Precision:
$$Precision = \frac{136}{136 + 9} = 0.94$$
- Recall:
$$Recall = \frac{136}{136 + 7} = 0.95$$
- F1 Score:
$$F1\ Score = \frac{2 \times 0.94 \times 0.95}{0.94 + 0.95} = 0.94$$

### II.3.2 Comments

***Analysis***

*Dataset of 40/60*

Confusion Matrix

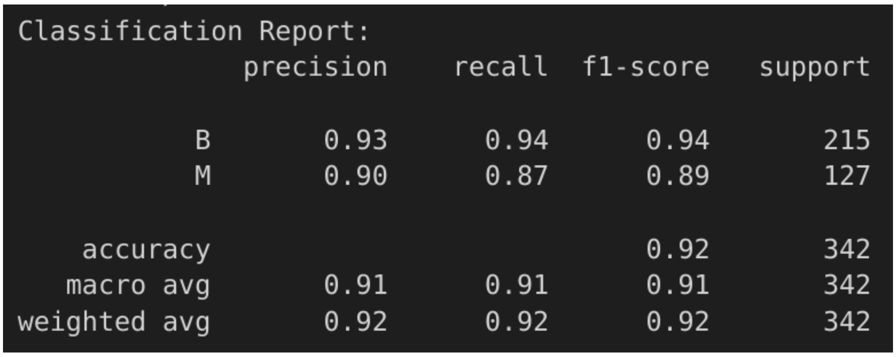


Confusion Matrix on 40/60 Dataset

This confusion matrix means:

- There are 203 Benign cases that the model predicted correctly.
- There are 12 Benign cases that the model predicted wrong.
- There are 16 Malignant cases that the model predicted wrong.
- There are 111 Malignant cases that the model predicted correctly.

Classification Report



```
Classification Report:
              precision    recall  f1-score   support

           B       0.93      0.94      0.94       215
           M       0.90      0.87      0.89       127

    accuracy                           0.92       342
   macro avg       0.91      0.91      0.91       342
weighted avg       0.92      0.92      0.92       342
```

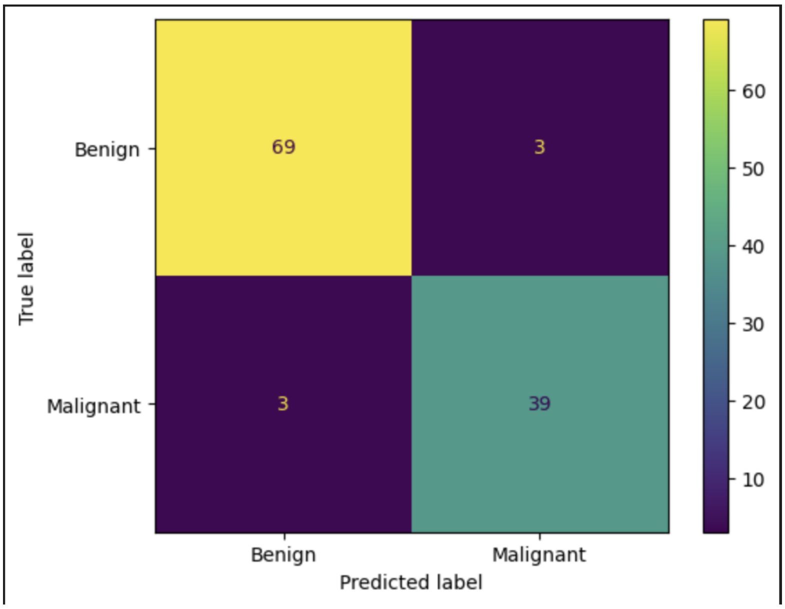
Classification Report on 40/60 Dataset

The Malignant class is chosen to be Positive, while the Benign class is Negative. The statistics can be calculated as follows:

- Precision:
$$Precision = \frac{111}{111+12} = 0.90$$
- Recall:
$$Recall = \frac{111}{111+16} = 0.87$$
- F1 Score:
$$F1\ Score = \frac{2 \times 0.90 \times 0.87}{0.90+0.87} = 0.89$$

*Dataset of 80/20*

Confusion Matrix

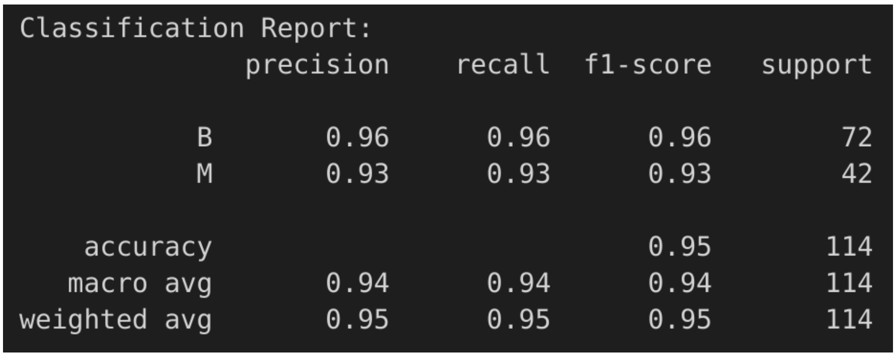


Confusion Matrix on 80/20 Dataset

This confusion matrix means:

- There are 69 Benign cases that the model predicted correctly.
- There are 3 Benign cases that the model predicted wrong.
- There are 3 Malignant cases that the model predicted wrong.
- There are 39 Malignant cases that the model predicted correctly.

Classification Report



```
Classification Report:
              precision    recall  f1-score   support

           B       0.96      0.96      0.96        72
           M       0.93      0.93      0.93        42

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

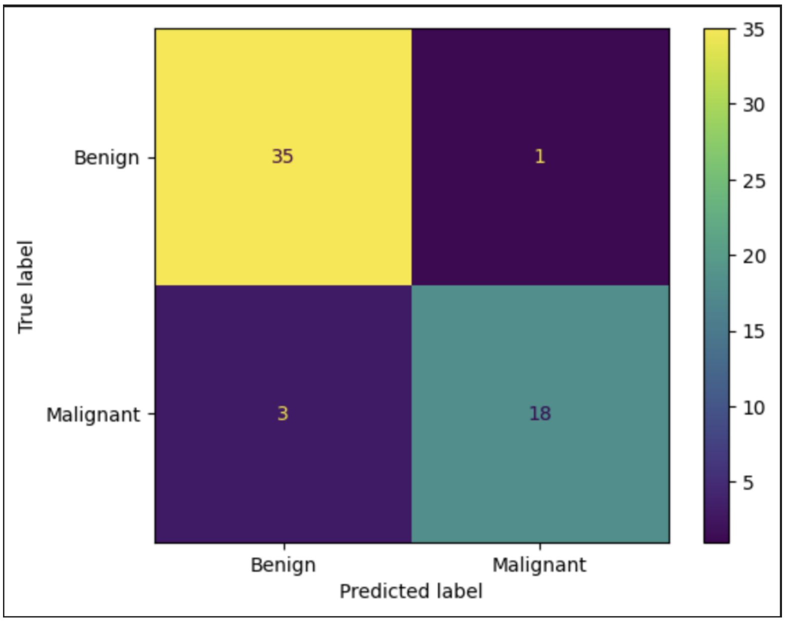
Classification Report on 80/20 Dataset

The Malignant class is chosen to be Positive, while the Benign class is Negative. The statistics can be calculated as follows:

- Precision:
$$Precision = \frac{39}{39+3} = 0.93$$
- Recall:
$$Recall = \frac{39}{39+3} = 0.93$$
- F1 Score:
$$F1\ Score = \frac{2 \times 0.93 \times 0.93}{0.93 + 0.93} = 0.93$$

*Dataset of 90/10*

Confusion Matrix

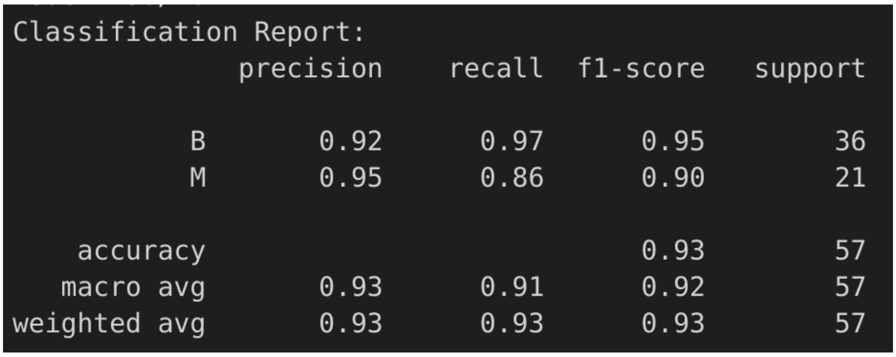


Confusion Matrix on 90/10 Dataset

This confusion matrix means:

- There are 35 Benign cases that the model predicted correctly.
- There are 1 Benign cases that the model predicted wrong.
- There are 3 Malignant cases that the model predicted wrong.
- There are 18 Malignant cases that the model predicted correctly.

Classification Report



```
Classification Report:
              precision    recall  f1-score   support

           B       0.92      0.97      0.95        36
           M       0.95      0.86      0.90        21

    accuracy                           0.93        57
   macro avg       0.93      0.91      0.92        57
weighted avg       0.93      0.93      0.93        57
```

Classification Report on 90/10 Dataset

The Malignant class is chosen to be Positive, while the Benign class is Negative. The statistics can be calculated as follows:

- Precision:
$$Precision = \frac{18}{18+1} = 0.95$$
- Recall:
$$Recall = \frac{18}{18+3} = 0.86$$
- F1 Score:
$$F1\ Score = \frac{2 \times 0.95 \times 0.86}{0.95 + 0.86} = 0.90$$

***Why do we have to use these metrics?***

When using these metrics, the first question lies in our head is: why do we have to use these two metrics instead of `accuracy_score` for simplicity? The answer is simple (but not simple): decision tree is an algorithm which is vulnerable to over-fitting, especially in the situation when there is **Survivorship Bias** in the distribution.

Usually, we calculate the accuracy of the model by just take the correct prediction divided by total number of samples. This is usually true, except for a situation: the percentage of Malignant cases are very low (such as less than 1%) (and we assume that all the breast cancer samples are drawn from the same distribution). The model doesn't have enough data of Malignant cases to know how to differentiate from the Benign. After training, when calculating accuracy by taking the number of correct predictions divided by number of all predictions, the accuracy is very high but there are still a lot of wrong labels ”literally thrown out of the window”. The classifier in this case is not much different than a line of code:

```
print("label=Benign")
```

The case in which the number of a class is overwhelming comparing to other classes is called **Imbalanced Classes**, and the problem with accuracy is called **Accuracy Paradox**. This is why we have to use Confusion Matrix, as well as Precision, Recall and F1 Score. Based on these metrics, one can choose the direction to modify and fine-tune the model in order to achieve the desired target.

## II.4 The depth and accuracy of a decision tree

### II.4.1 Model training and visualization

For this problem, the original 80/20 dataset from the beginning is used but different models are created to ensure the integrity. The models are saved inside a dictionary `model_max_depth`. To access a specific model (for example, maximum depth of 4), do the following:

```
model = models_max_depth[4]
```

Using function `export_model_image(model, filename)` to save and load model visualizations. Images are saved in `images/depth_accuracy`.

### II.4.2 Comments

The correlation between between different max_depth and accuracy_score of the decision tree classifier is shown in the table below.

| max_depth | None | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| Accuracy | 0.912281 | 0.921053 | 0.921053 | 0.912281 | 0.912281 | 0.95614 | 0.929825 |

As the statistics shown, the best accuracy_score belongs to the decision tree classifier with the maximum depth of 6. This is higher than both the **1.2.1 problem** 80/20 model (which is 0.94737) and the **1.2.4 problem** model (which is only 0.912281).

# III Reference

- https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
- https://scikit-learn.org/0.15/modules/generated/sklearn.metrics.classification_report.html
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html
- https://proclusacademy.com/blog/explainer/precision-recall-f1-score-classification-models/#precision
- https://stackoverflow.com/questions/76796808/what-is-support-in-classification-report-within-sklearn
- https://stackoverflow.com/questions/53391444/how-to-resize-the-image-of-the-tree-using-sklearn-tree-and-export-graph-viz-with