2110446 DATA SCIENCE AND DATA ENGINEERING

Homework 3: Decision Tree

Computer Engineering, Chulalongkorn Univerity

Worralop Srichainont

January 25, 2026

Contents

1 Problem Description

1.1 Kyphosis

Kyphosis is a condition where your spine curves outward more than it should. This causes your upper back around the thoracic region (the part of your spine between your neck and ribs) to bend forward.

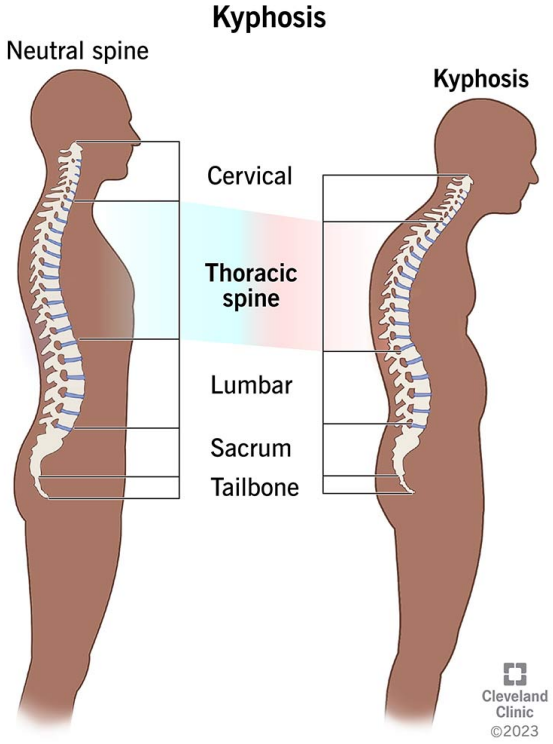

Figure 1: Kyphosis Symptoms

1.2 Dataset

The dataset has 81 rows and 4 columns which have the following details:

1. `Kyphosis` is the target column.
   (a) `absent` means the spine has not deformed.
   (b) `present` means the spine has deformed.
2. `Age` is the age of the patient in months unit.
3. `Number` is the total number of vertebrae (spine bones) that were involved in the surgical procedure.
4. `Start` is the index of the topmost vertebra (spine bone) that was operated on.

	Kyphosis	Age	Number	Start
0	absent	71	3	5
1	absent	158	3	14
2	present	128	4	5
3	absent	2	5	1
4	absent	1	4	15

Figure 2: Dataset Example

For `Start` column in the dataset, it treats human vertebrae as a continuous sequence from 1 to 24.

- **Cervical vertebrae (C1-C7)** is numbered as 1 to 7.
- **Thoracic vertebrae (T1-T12)** is numbered as 8 to 19.
- **Lumbar vertebrae (L1-L5)** is numbered as 20 to 24.

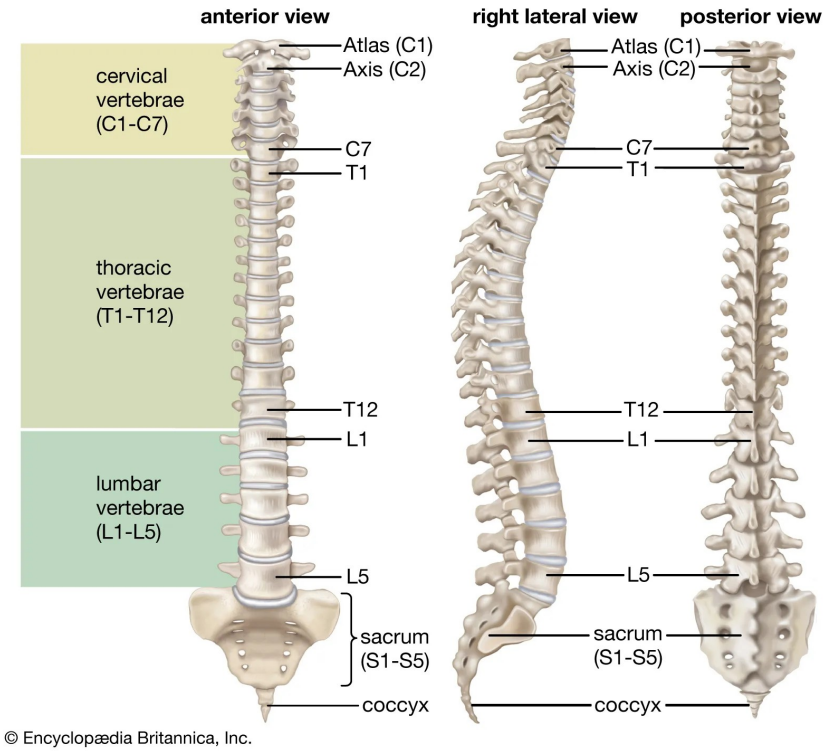

Figure 3: Vertebrae Structure

1.3 Objective

Models

1. **Decision Tree Model** is a flowchart-like tree used for making decisions or predictions. It breaks down a complex decision into a series of questions eventually leading to a final answer.
2. **Random Forest Model** is a technique that create multiple decision trees from a different random sample of the original dataset with random features.
   (a) **Classification** uses the majority of votes from the forest of decision trees.
   (b) **Regression** uses the average value from the forest of decision trees.

Assignment

Please study the demo notebook and create a report and export it as a PDF file, comparing:

- Decision Tree model with adjusted parameters.
- Random Forest model with adjusted parameters.

Objective: Achieve the highest possible macro-F1 score with proper interpretation.

2 Data Gathering

2.1 Colab Notebook

To access the colab notebook, please follow this link: `https://colab.research.google.com/drive/19gwAoeK6NIPp-c9uK-8UBJO1VauI06PL`

2.2 Dependencies

Before we start, please import these libraries.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import ConfusionMatrixDisplay,
    classification_report
```

Code 1: Dependencies

2.3 Load Data

Next, we are going to load CSV dataset file by using `pandas`.

```
KYPHOSIS_URL = "https://raw.githubusercontent.com/reisenx/2110446-DATA-
    SCI-ENG/refs/heads/main/03-Traditional-ML/Homework/kyphosis.csv"

KYPHOSIS_DF = pd.read_csv(KYPHOSIS_URL)
```

Code 2: Load Dataset

3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is the data analysis to identifies general patterns in the data.

3.1 Class Distribution

Class Distribution shows amount of each target classes in the dataset.

```
sns.countplot(data=KYPHOSIS_DF, x="Kyphosis", hue="Kyphosis", palette="Set1")
plt.title("Class Distribution")
plt.show()
```

Code 3: Class Distribution

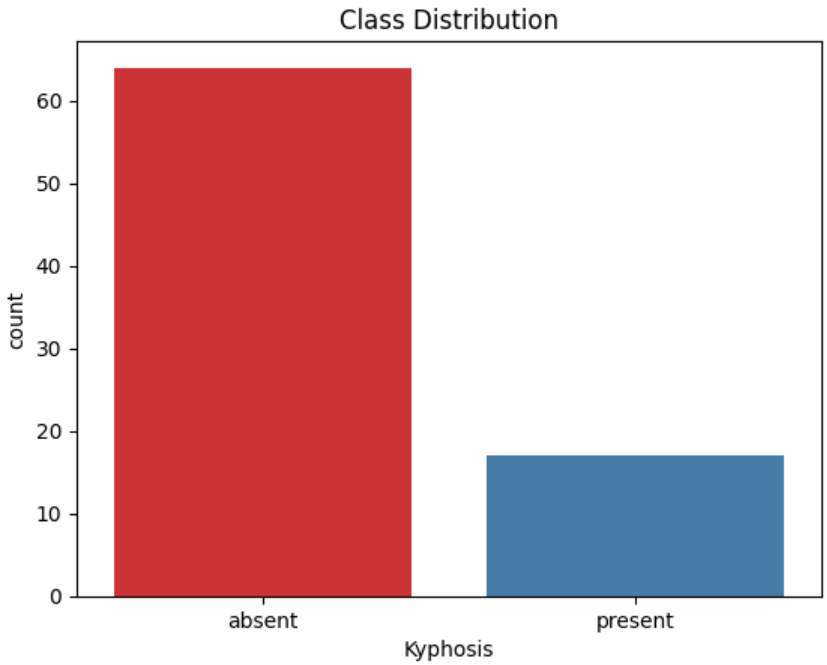

Figure 4: Class Distribution

3.2 Correlation Heatmap

Correlation Heatmap is the visualization the strength and direction of relationships between features.

- **Correlation Value** is in range $[-1.0, 1.0]$ indicating strength of the correlation.
- **Positive Correlation**: As one variable increases, the other also increases.
- **Negative Correlation**: As one variable increases, the other decreases.
- **No Correlation**: There is no linear relationship between the variables.

```
features = KYPHOSIS_DF.drop(columns=["Kyphosis"])
sns.heatmap(features.corr(), annot=True, cmap="coolwarm")
plt.show()
```

Code 4: Correlation Heatmap

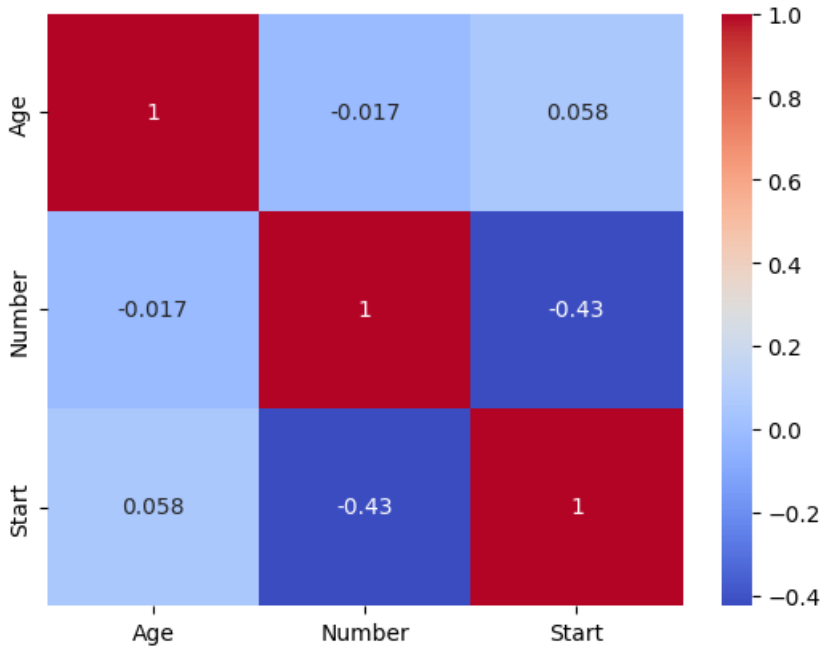

Figure 5: Correlation Heatmap

3.3 Pair Plot

Pair plot is a matrix of graphs that enables the visualization of the relationship between each pair of variables in a dataset.

- **Histogram Plot** is on the diagonal line. It shows the distribution of a single features separated by target classes.
- **Scatter Plot** shows the relationship of two variables.

```
sns.pairplot(data=KYPHOSIS_DF, hue="Kyphosis", palette="Set1")
plt.show()
```

Code 5: Pair Plot

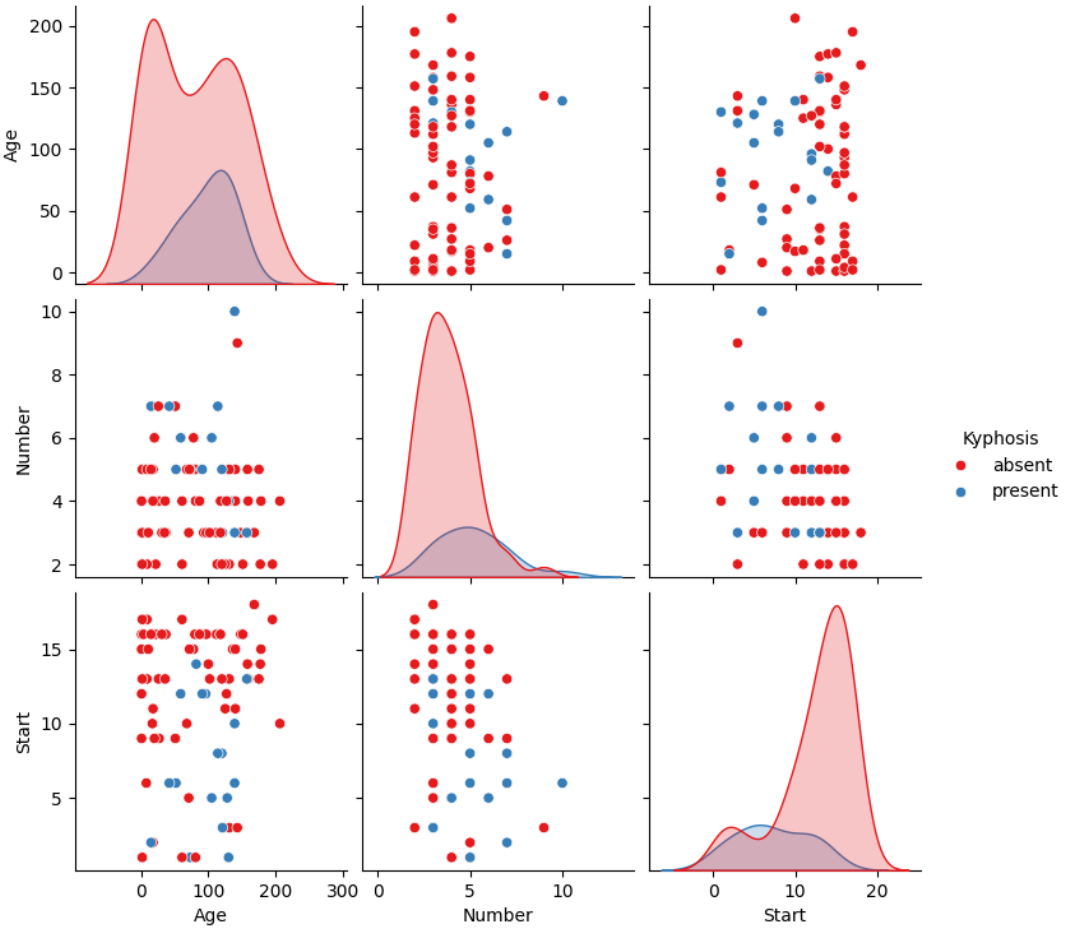

Figure 6: Pair Plot

3.4 Interpretation

1. **Class Imbalance**: The dataset has 64 `absent` class (79%), but only 17 `present` class (21%).
2. **Features Relationship**: According to the correlation heatmap and pair plot
   - `Age` has no linear correlation to any features.
   - Only `Number` and `Start` features have notable negative linear correlation.
3. **Overlapping**: The classes overlap significantly in the feature space.

4 Data Preparation

4.1 Feature and Target Separation

1. **Features** (`X`) are all columns except the `Kyphosis` column.
2. **Target** (`y`) is the `Kyphosis` column.

```
X = KYPHOSIS_DF.drop(columns=["Kyphosis"])
y = KYPHOSIS_DF["Kyphosis"]
```

Code 6: Feature and Target Separation

4.2 Train and Test Dataset

Stratification is a dataset separation technique can ensure that both train datasets and test dataset have the same `absent` and `present` proportions.

In this problem, we are going to split the dataset into train dataset and test dataset by ratio 7 : 3 using stratification technique.

- **Train dataset** (70% of the dataset) uses for training the model.
- **Test dataset** (30% of the dataset) uses for testing the model.

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=30, stratify=y
)
```

Code 7: Train and Test Dataset Separation

5 Decision Tree

5.1 Adjusting Parameters

1. Choose `criterion` to `gini` instead of `entropy`:
    - Choosing **Gini Impurity** to decide how to split the data.
    - **Gini Impurity** often works better on imbalanced data.
    - **Gini Impurity** tends to isolating the most frequent class first.
2. Choose `max_depth` to 3:
    - Limits the decision tree to growing only 3 levels deep.
    - Prevents over-fitting which occurred when the decision tree has too many level.
3. Choose `max_samples_split` to 16:
    - Requires 16 samples to split a node.
    - Prevents the decision tree model for making rules based only small clusters.
4. Choose `max_samples_leaf` to 8:
    - Every leaf node must have at least 8 samples.
    - Prevents the decision tree model for making rules just for a tiny cluster.
5. Choose `ccp_alpha` to 0.01:
    - **Cost Complexity Pruning** uses for pruning too-complex decision tree.
    - Prevents branching when it is just slightly improve the purity.

```
CRITERION = "gini"
MAX_DEPTH = 3
MIN_SAMPLES_SPLIT = 16
MIN_SAMPLES_LEAF = 8
CCP_ALPHA = 0.01
```

Code 8: Decision Tree Parameters

5.2 Decision Tree Model

Initialize and train a decision tree model using adjusted parameters.

```
model = DecisionTreeClassifier(
    criterion=CRITERION,
    max_depth=MAX_DEPTH,
    min_samples_split=MIN_SAMPLES_SPLIT,
    min_samples_leaf=MIN_SAMPLES_LEAF,
    ccp_alpha=CCP_ALPHA,
)
model.fit(X_train, y_train)
```

Code 9: Decision Tree Parameters

To visualize the decision tree, you can use `plot_tree()`.

```
plt.figure(figsize=(6, 10))
plot_tree(
    model,
    feature_names=X_train.columns,
    class_names=y_train.unique().tolist(),
    filled=True,
    rounded=True,
)
plt.show()
```

Code 10: Decision Tree Visualization

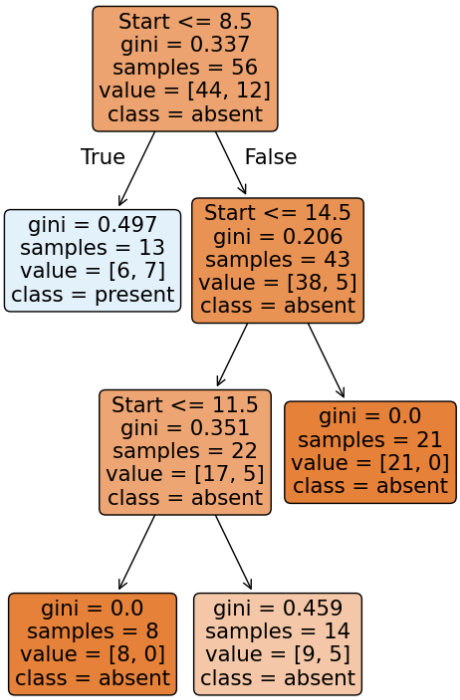

Figure 7: Decision Tree Visualization

5.3 Confusion Matrix

Confusion Matrix is a simple table used to measure how well a classification model is performing. It compares the predictions made by the model with the actual results and shows where the model was right or wrong.

Given `absent` is the `positive` class, and `present` is `negative` class.

- **True Positive (TP)** means predicted `positive`, actual `positive`.
- **False Positive (FP)** means predicted `positive`, actual `negative`.
- **True Negative (TN)** means predicted `negative`, actual `negative`.
- **False Negative (FN)** means predicted `negative`, actual `positive`.

For text visualization use `confusion_matrix()`, but for image visualization use `ConfusionMatrixDisplay.from_predictions()`.

```
confusion_matrix = ConfusionMatrixDisplay.from_predictions(
    y_test, predictions, cmap=plt.cm.Greens
)
plt.title("Confusion Matrix")
plt.show()
```

Code 11: Confusion Matrix of Decision Tree

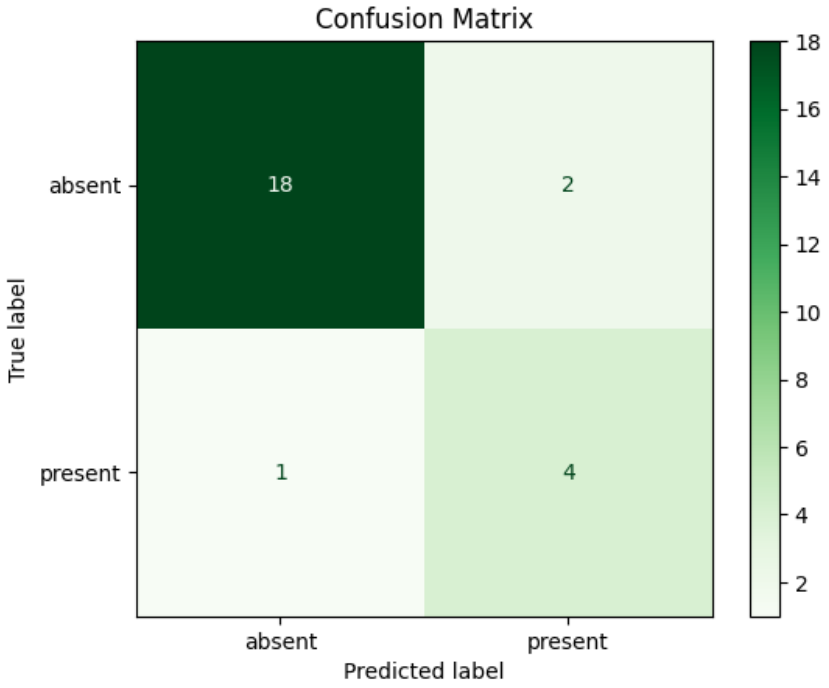

Figure 8: Confusion Matrix of Decision Tree

5.4 Classification Report

Class Metrics

- **Precision** is the proportions of true positive (TP) by total positive predictions.
- **Recall** is the proportions of true positive (TP) by total actual positive.
- **Support** is the amount of actual data in the class.
- **F1 Score** is the harmonic mean of **Precision** and **Recall**.

Overall Metrics

- **Accuracy** is the proportions of correct predictions by total predictions.
- **Macro F1 Score** is mean of per-class F1 score.
- **Weight Average F1 Score** is mean of per-class F1 score weighted by each class's support.

```
report = classification_report(y_test, predictions, digits=4)
print(report)
```

Code 12: Classification Report of Decision Tree

Class	Precision	Recall	F1 Score	Support
absent	0.9474	0.9000	0.9231	20
present	0.6667	0.8000	0.7273	5

Table 1: Class Metrics

Accuracy	Macro F1 Score	Weighted F1 Score
0.8800	0.8252	0.8839

Table 2: Overall Metrics

Result: This decision tree model got **0.8252 F1 Macro Score**.

6 Random Forest

6.1 Adjusting Parameters

1. Choose `n_estimators` to 100:
   - Create 100 different decision tree to vote a result.
2. Choose `criterion` to `gini` instead of `entropy`:
   - Choosing **Gini Impurity** to decide how to split the data.
   - **Gini Impurity** often works better on imbalanced data.
   - **Gini Impurity** tends to isolating the most frequent class first.
3. Choose `max_depth` to 3:
   - Limits the decision tree to growing only 3 levels deep.
   - Prevents over-fitting which occurred when the decision tree has too many level.
4. Choose `max_samples_split` to 8:
   - Requires 8 samples to split a node.
   - Prevents the decision tree model for making rules based only small clusters.
5. Choose `max_samples_leaf` to 4:
   - Every leaf node must have at least 4 samples.
   - Prevents the decision tree model for making rules just for a tiny cluster.
6. Choose `max_features` to 2:
   - Choose only 2 features when sampling the data.
   - This makes the dataset more vary which makes the each decision tree different.
7. Choose `random_state` to 32830:
   - Fix the random seed to ensure the same result when executing the code.
8. Choose `ccp_alpha` to 0.01:
   - **Cost Complexity Pruning** uses for pruning too-complex decision tree.
   - Prevents branching when it is just slightly improve the purity.
9. Choose `max_samples` to 50:
   - Choose only 50 rows when sampling the data.
   - This makes the dataset more vary which makes the each decision tree different.

```
N_ESTIMATORS = 100
CRITERION = "gini"
MAX_DEPTH = 3
MIN_SAMPLES_SPLIT = 8
MIN_SAMPLES_LEAF = 4
MAX_FEATURES = 2
RANDOM_STATE = 32820
CCP_ALPHA = 0.01
MAX_SAMPLES = 40
```

Code 13: Random Forest Model Parameters

6.2 Random Forest Model

Initialize and train a random forest model using adjusted parameters.

```
model = RandomForestClassifier(
    n_estimators=N_ESTIMATORS,
    criterion=CRITERION,
    max_depth=MAX_DEPTH,
    min_samples_split=MIN_SAMPLES_SPLIT,
    min_samples_leaf=MIN_SAMPLES_LEAF,
    max_features=MAX_FEATURES,
    random_state=RANDOM_STATE,
    ccp_alpha=CCP_ALPHA,
    max_samples=MAX_SAMPLES,
)
model.fit(X_train, y_train)
```

Code 14: Random Forest Model Parameters

Random Forest Model consists of multiple decision trees which can access by `.estimators_` properties. To access a single decision tree, use indexing.

```
example_tree_01 = model.estimators_[0]
example_tree_02 = model.estimators_[4]
```

Code 15: Random Forest Model Parameters

Here are the visualization of the first and the fifth decision tree from a random forest model.

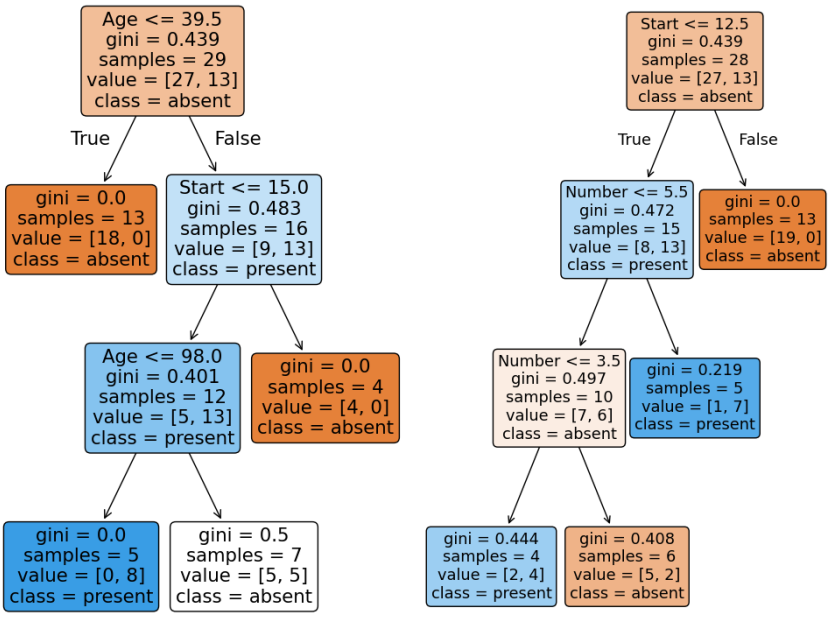

Figure 9: The First Decision Tree (left) The Fifth Decision Tree (right)

6.3 Confusion Matrix

Confusion Matrix is a simple table used to measure how well a classification model is performing. It compares the predictions made by the model with the actual results and shows where the model was right or wrong.

Given `absent` is the `positive` class, and `present` is `negative` class.

- **True Positive (TP)** means predicted `positive`, actual `positive`.
- **False Positive (FP)** means predicted `positive`, actual `negative`.
- **True Negative (TN)** means predicted `negative`, actual `negative`.
- **False Negative (FN)** means predicted `negative`, actual `positive`.

For text visualization use `confusion_matrix()`, but for image visualization use `ConfusionMatrixDisplay.from_predictions()`.

```
confusion_matrix = ConfusionMatrixDisplay.from_predictions(
    y_test, predictions, cmap=plt.cm.Greens
)
plt.title("Confusion Matrix")
plt.show()
```

Code 16: Confusion Matrix of Decision Tree

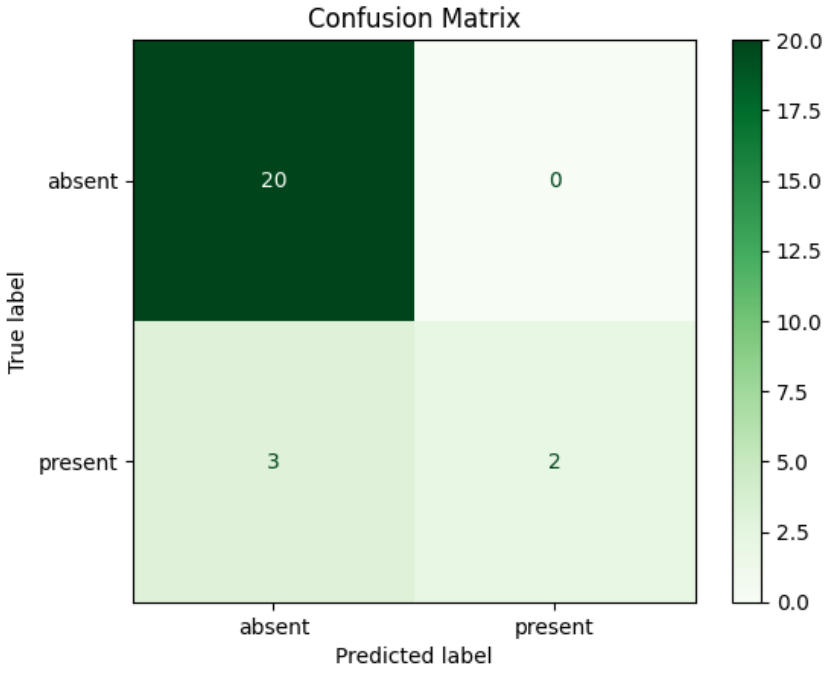

Figure 10: Confusion Matrix of Random Forest Model

6.4 Classification Report

Class Metrics

- **Precision** is the proportions of true positive (TP) by total positive predictions.
- **Recall** is the proportions of true positive (TP) by total actual positive.
- **Support** is the amount of actual data in the class.
- **F1 Score** is the harmonic mean of **Precision** and **Recall**.

Overall Metrics

- **Accuracy** is the proportions of correct predictions by total predictions.
- **Macro F1 Score** is mean of per-class F1 score.
- **Weight Average F1 Score** is mean of per-class F1 score weighted by each class's support.

```
report = classification_report(y_test, predictions, digits=4)
print(report)
```

Code 17: Classification Report of Random Forest Model

Class	Precision	Recall	F1 Score	Support
absent	0.8696	1.0000	0.9302	20
present	1.0000	0.4000	0.5714	5

Table 3: Class Metrics

Accuracy	Macro F1 Score	Weighted F1 Score
0.8800	0.7508	0.8585

Table 4: Overall Metrics

Result: This random forest model got **0.7508 F1 Macro Score**.