1 Solution to Problem1

1.1 Dataset Description

The **Iris dataset** consists of 150 samples from three different species of iris flowers (*Setosa, Versicolor, and Virginica*), with four numerical features per sample:

- **Sepal Length (cm)**, **Sepal Width (cm)**, **Petal Length (cm)**, **Petal Width (cm)**,
- **Species**: The class label, indicating the species of the Iris flower.

The dataset contains both categorical and numerical attributes:

- **Nominal Attribute**: *Species*.
- **Continuous Attribute**: 4 numerical features, including Sepal Length, Sepal Width, Petal Length and Petal Width.

1.2 Data Preprocessing

We use the `load_iris()` dataset from `sklearn`, first loading it into a Pandas DataFrame and performing data preprocessing. The scatter matrix plot of the Iris dataset is shown in Figure 1.

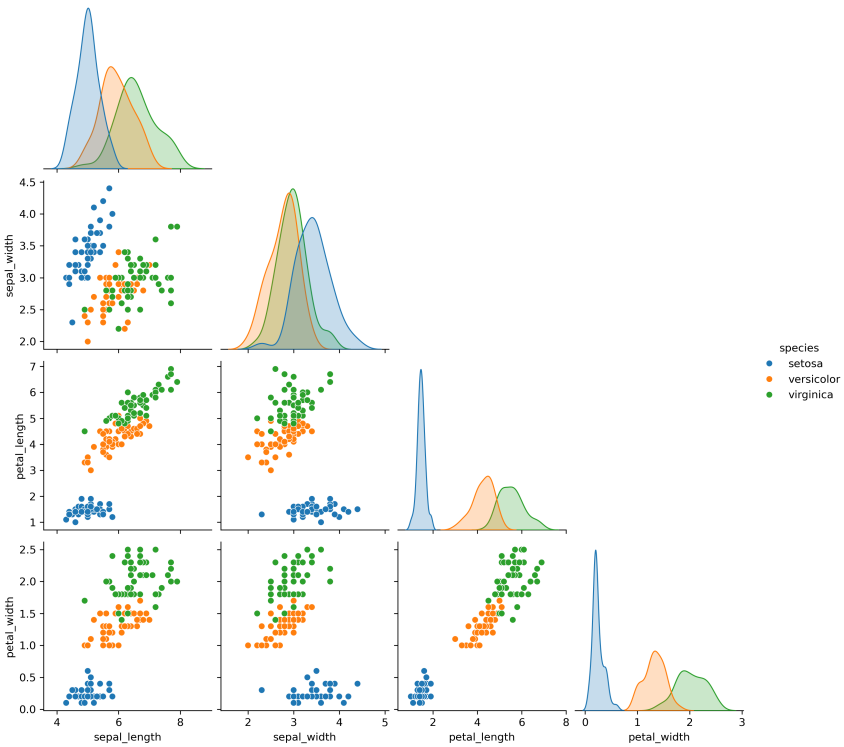


Figure 1: Scatter Matrix Plot of Iris Dataset

1.3 Decision Tree Construction with Different Depths

We perform multi-class classification using all classes in the Iris dataset. The `iris.target` contains three class labels: 0 (setosa), 1 (versicolor), and 2 (virginica).

We train a decision tree model using `DecisionTreeClassifier` from `sklearn.tree`. The dataset was split into **70% training** and **30% testing** using `train_test_split()`. The following parameter are set:

- `min_samples_leaf=2`: Each leaf node must contain at least 2 samples.

- `min_samples_split=5`: Each node must have at least 5 samples to split.

- `max_depth=1` to `5`: The maximum depth of the tree ranges from 1 to 5.

The decision tree diagram is shown in Figure 2.

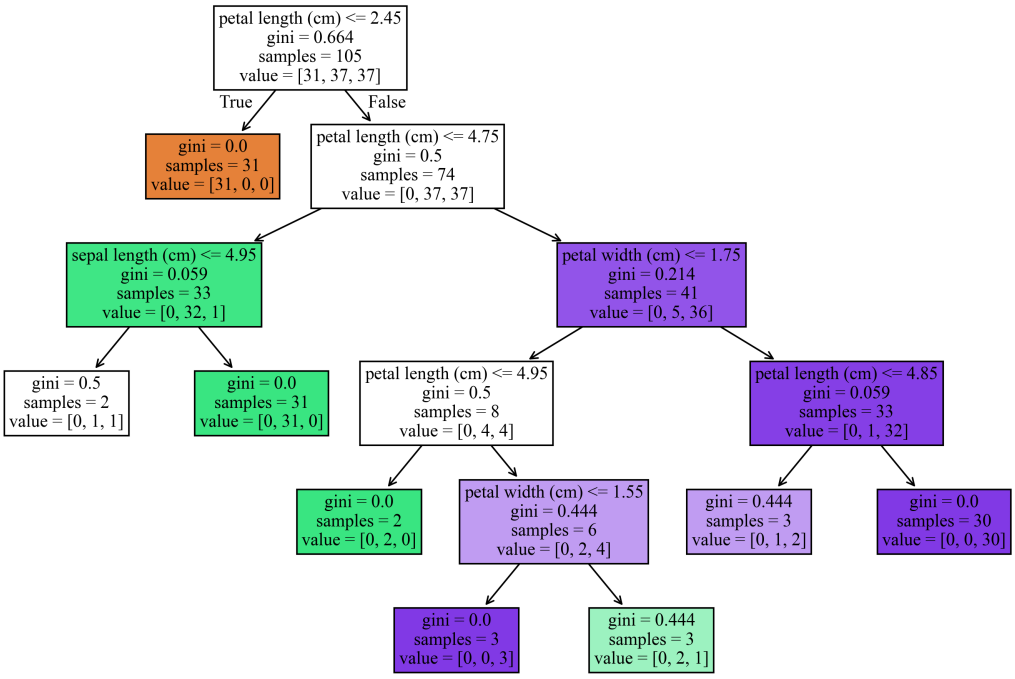


Figure 2: Tree Plot

1.4 Performance Metrics Evaluation

We compute the following metrics:

- Precision, Recall, and F1 Score for each Iris class at each tree depth.

- Macro-averaged, Weighted-averaged, and Micro-averaged scores for Precision, Recall, and F1 Score at each tree depth.

The confusion matrix is shown in Figure 3.

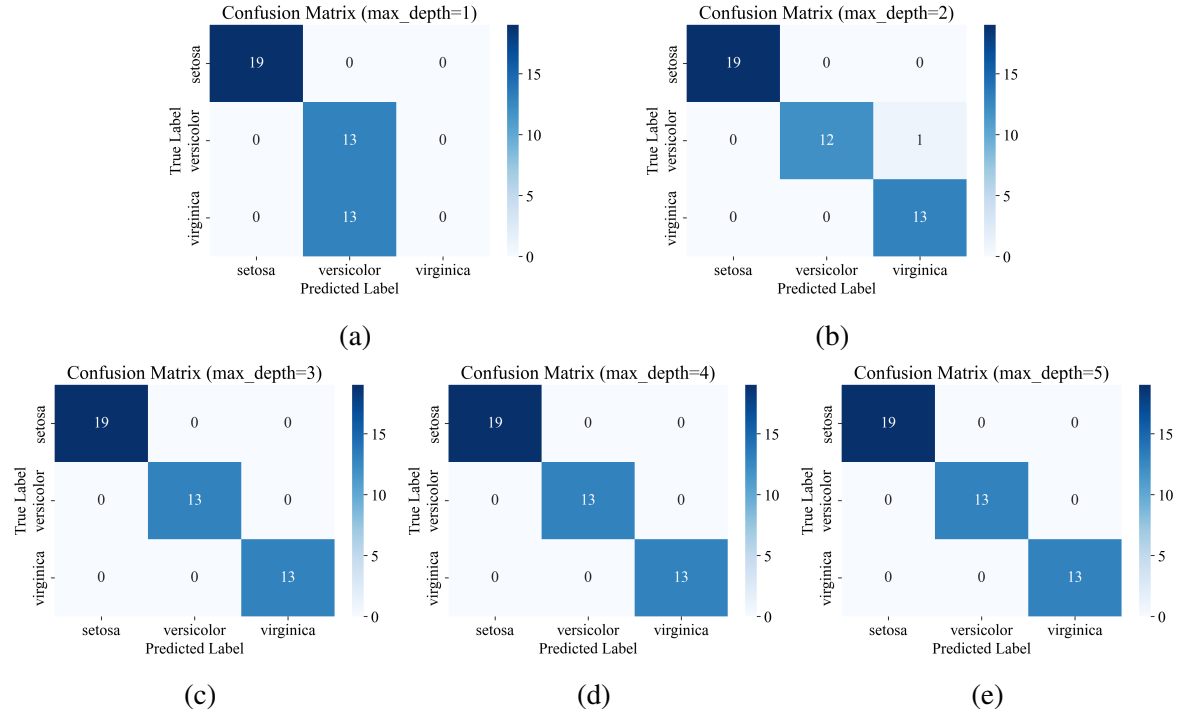


Figure 3: Confusion Matrices

From Figure 3, we can observe: As the tree depth increases from 1 to 3, the number of diagonal elements increases while the number of non-diagonal elements decreases, indicating that the number of correctly classified samples increases while the number of misclassified samples decreases. When the depth reaches 3, all non-diagonal elements become zero, suggesting that the model has achieved optimal classification performance. As the depth increases from 3 to 5, the classification performance remains unchanged, maintaining the optimal classification.

The computed metric results are shown in Table 1.

Table 1: Computed Metric Results

Max Depth	**1**			**2**			**3 to 5**		
Metrics	Precision	Recall	F1 Score	Precision	Recall	F1 Score	Precision	Recall	F1 Score
Setosa	1	1	1	1	1	1	1	1	1
Versicolor	0.5	1	0.67	1	0.92	0.96	1	1	1
Virginica	0	0	0	0.93	1	0.96	1	1	1
Macro	0.5	0.67	0.56	0.98	0.97	0.97	1	1	1
Weighted	0.57	0.71	0.61	0.98	0.98	0.98	1	1	1
Micro	0.71	0.71	0.71	0.98	0.98	0.98	1	1	1

From the table, we can see :

- At a depth of 1, all metrics for Virginica are zero because only one split has been performed, dividing the samples into two classes. At this depth, Setosa has already achieved a Precision, Recall, and F1 Score of 1, but Versicolor still has relatively low Precision and F1 Score. Meanwhile, the Macro-averaged, Weighted-averaged, and Micro-averaged scores for all three metrics remain low due to the poor classification performance resulting from a single split.
- When the depth increases to 2, the metrics for all three Iris classes improve significantly, and the Macro-averaged, Weighted-averaged, and Micro-averaged scores for Precision, Recall, and F1 Score also show a substantial increase.
- As the depth increases from 3 to 5, all metrics, including Precision, Recall, and F1 Score for each Iris class, as well as the Macro-averaged, Weighted-averaged, and Micro-averaged scores for these three metrics, reach the maximum value of 1 and remain unchanged. This indicates that the model has achieved optimal classification performance.

1.5 Result and Analysis

The classification performance of the decision tree model varies with different depths. The observations are as follows:

1. **A depth of 3 to 5 results in the highest Recall.**

   Recall is calculated as:

   $$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{1}$$

   At this depth, the decision tree correctly classifies all samples, leading to zero False Negatives and thus achieving the maximum Recall of 1. A deeper decision tree allows for more refined splits of the data, leading to improved classification performance and higher recall.

2. **A depth of 1 results in the lowest Precision.**

   Precision is calculated as:

   $$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

   With a shallow depth, the decision tree model is too simple to capture the complexity of the data, leading to a higher number of False Positives and consequently a lower Precision.

3. **A depth of 3 to 5 leads to the optimal F1 Score.**

   F1 Score is calculated as:

   $$F1\ \text{Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

   The F1 Score represents a balance between Precision and Recall. Since both Precision and Recall reach their maximum values of 1 at this depth, the F1 Score also achieves its highest possible value of 1, indicating optimal classification performance.

1.6 Explanation of Micro, Macro and Weighted Scores

The three different calculation methods - Micro, Macro, and Weighted - are defined as follows:

- **Macro Average**: Computes the evaluation metrics (such as Precision, Recall, and F1 Score) for each class separately and then takes the average of these metrics. Each class is given equal weight, regardless of class imbalance.
- **Weighted Average**: Computes the evaluation metrics (such as Precision, Recall, and F1 Score) for each class separately and then takes a weighted average, where the weight is determined by the number of samples in each class. This method is suitable for imbalanced datasets.
- **Micro Average**: Aggregates the total True Positives (TP), False Positives (FP), and False Negatives (FN) across all classes, without distinguishing between individual classes, to compute overall Precision, Recall, and F1 Score. This method is useful for handling class imbalance.

The formulas for these three averaging methods are given as follows:

$$\text{Macro Precision} = \frac{1}{N}\sum_{i=1}^{N}\text{Precision}_i \tag{4}$$

$$\text{Macro Recall} = \frac{1}{N}\sum_{i=1}^{N}\text{Recall}_i \tag{5}$$

$$\text{Macro F1} = \frac{1}{N}\sum_{i=1}^{N}\text{F1 Score}_i \tag{6}$$

$$\text{Weighted Precision} = \sum_{i=1}^{N} w_i \times \text{Precision}_i, \quad w_i = \frac{n_i}{\sum_{j=1}^{N} n_j} \tag{7}$$

$$\text{Weighted Recall} = \sum_{i=1}^{N} w_i \times \text{Recall}_i, \quad w_i = \frac{n_i}{\sum_{j=1}^{N} n_j} \tag{8}$$

$$\text{Weighted F1} = \sum_{i=1}^{N} w_i \times \text{F1 Score}_i, \quad w_i = \frac{n_i}{\sum_{j=1}^{N} n_j} \tag{9}$$

$$\text{Micro Precision} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N}(TP_i + FP_i)} \tag{10}$$

$$\text{Micro Recall} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N}(TP_i + FN_i)} \tag{11}$$

$$\text{Micro F1} = \frac{2 \times \text{Micro Precision} \times \text{Micro Recall}}{\text{Micro Precision} + \text{Micro Recall}} \tag{12}$$

where N is the total number of classes, n_i is the number of samples in class i, and TP_i, FP_i, and FN_i are the true positives, false positives, and false negatives for class i, respectively.

2 Solution to Problem2

2.1 Dataset Description

The Wisconsin Breast Cancer dataset contains information on 699 patients, aiming to classify tumors as either benign or malignant based on 11 attributes, which is described as follows:

- **ID**: Unique identifier for each patient (not used in classification).
- **Clump Thickness**: Measures the thickness of cell clumps.
- **Uniformity of Cell Size**: Evaluates how uniform the cell sizes are.
- **Uniformity of Cell Shape**: Measures the consistency of cell shapes.
- **Marginal Adhesion**: Assesses the ability of cells to adhere to each other.
- **Single Epithelial Cell Size**: Examines the size of single epithelial cells.
- **Bare Nuclei**: Indicates the presence of bare nuclei in the sample (may have missing values).
- **Bland Chromatin**: Evaluates chromatin texture in cell nuclei.
- **Normal Nucleoli**: Counts the number of nucleoli in cell nuclei.
- **Mitoses**: Measures the number of cell mitotic divisions.
- **Class**: Target variable, where 2 represents benign tumors and 4 represents malignant tumors.

The dataset contains both categorical and numerical attributes:

- **Nominal Attribute**: *ID*, *Class* (benign or malignant).
- **Discrete Attributes**: All numeric attributes except *Bare Nuclei* are discrete, taking integer values from 1 to 10.
- **Continuous Attribute**: *Bare Nuclei* is recorded as a floating-point number, with some missing values.

2.2 Data Preprocessing

Since the data file does not contain column names, we manually added them. Next, we converted the *bare_nuclei* column to numeric values and removed rows with missing values. The target variable, *class*, originally had values 2 (benign) and 4 (malignant), which we mapped to a binary classification: $2 \rightarrow 0$, $4 \rightarrow 1$. The scatter matrix plot of the dataset is shown in Figure 4.

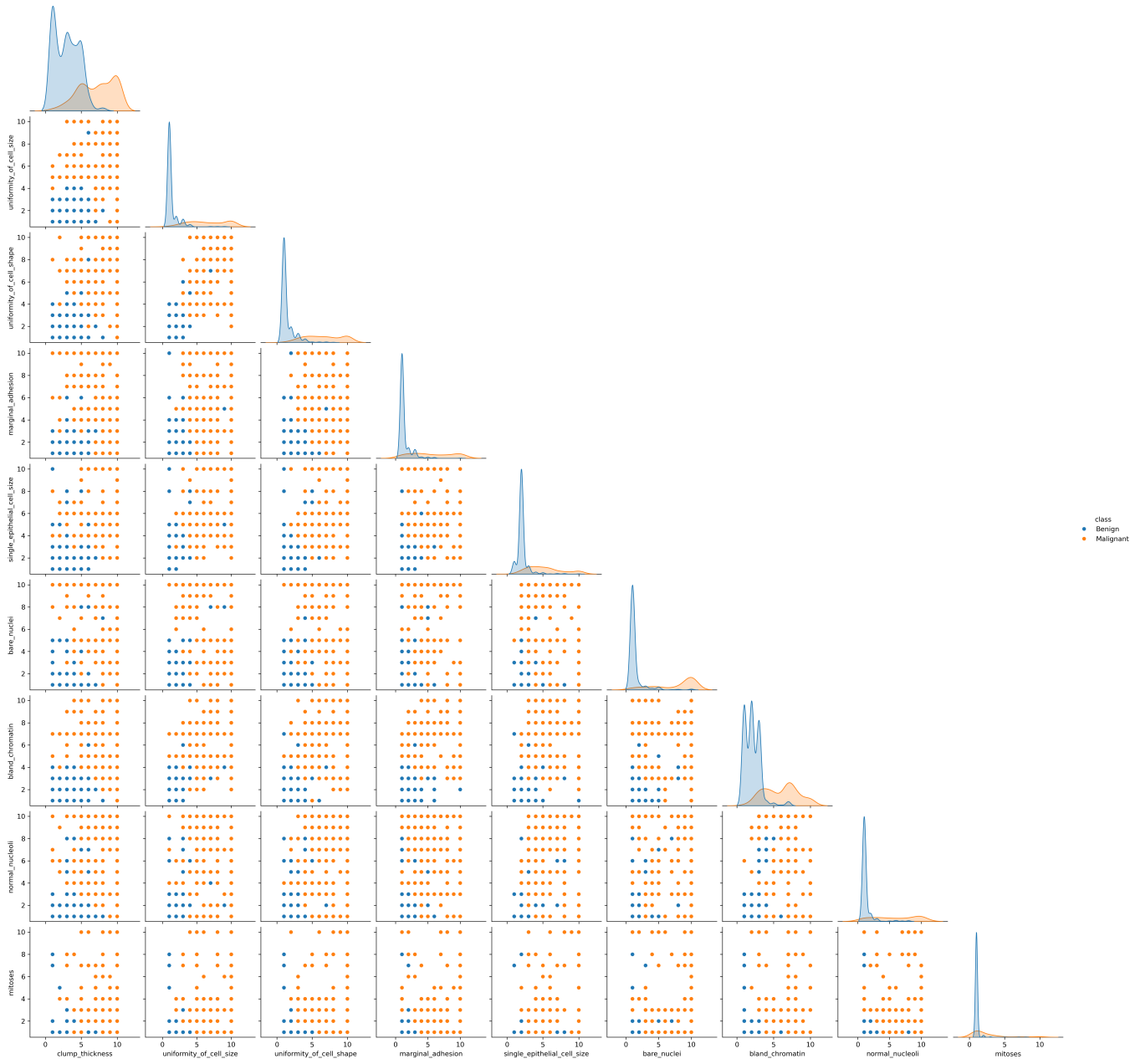


Figure 4: Scatter Matrix Plot of Breast Cancer (Discrete)

2.3 Decision Tree Construction

We used the *DecisionTreeClassifier* with the following parameter settings: *max_depth*=2, *min_samples_split*=5, and *min_samples_leaf*=2. Since we computed metrics for the dataset itself, we did not split it into training and testing sets but trained the decision tree on the entire dataset. After training, we accessed the tree structure using *clf.tree_* and extracted the feature index and threshold for the root node (the first split). The decision tree visualization is shown in Figure 5.

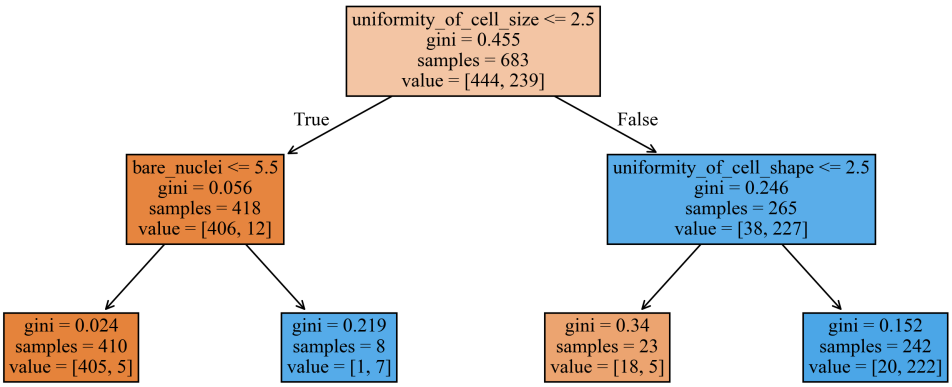


Figure 5: Tree Plot

2.4 Metric Computation

We first defined three functions to compute Entropy, Gini, and Misclassification Error. The formulas for these metrics are as follows:

Entropy measures the impurity of a dataset and is defined as:

$$H(S) = -\sum_{i=1}^{c} p_i \log_2 p_i \tag{13}$$

where:

- $H(S)$ is the entropy of the dataset S.
- c is the number of classes.
- p_i is the probability of class i in the dataset.

The Gini Index measures the probability of incorrectly classifying a randomly chosen element:

$$G(S) = 1 - \sum_{i=1}^{c} p_i^2 \tag{14}$$

where:

- $G(S)$ is the Gini Index of the dataset S.
- p_i is the probability of class i.

Misclassification Error represents the fraction of misclassified samples:

$$E(S) = 1 - \max(p_i) \tag{15}$$

where:

- $E(S)$ is the misclassification error.
- $\max(p_i)$ is the maximum probability among all classes.

Then, we calculated these metrics for the root node (the entire dataset). Next, based on the first split rule, we divided the dataset into left and right subsets and computed the metrics for each subset. The weighted average of the metrics was then computed based on the sample sizes of the left and right subsets.

Finally, we calculated the Information Gain for Entropy, Gini, and Misclassification Error, which is defined as the root node metric minus the weighted metric after the split. Information Gain (IG) measures the reduction in impurity after splitting the dataset:

$$IG = P(S) - \sum_{j=1}^{k} M(S_j) \tag{16}$$

where:

- IG is the information gain.
- $P(S)$ is the impurity measure before splitting.
- S_j represents the j-th subset after the split.
- k is the number of subsets.
- $M(S_j)$ is the impurity measure after splitting.

2.5 Result and Analysis

Feature selected for the first split: *uniformity_of_cell_size*
Decision boundary value: 2.5

Root Node Metrics

- Entropy: 0.9340
- Gini: 0.4550
- Misclassification Error: 0.3499

Weighted Average Metrics After the First Split

- Weighted Entropy: 0.3451

- Weighted Gini: 0.1294
- Weighted Misclassification Error: 0.0732

Information Gain

- Information Gain based on Entropy: 0.5889
- Information Gain based on Gini: 0.3255
- Information Gain based on Misclassification Error: 0.2767

3 Solution to Problem3

3.1 Dataset Description

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset contains a total of 569 instances with 32 attributes, including an ID, a diagnosis label, and 30 numerical features describing various characteristics of cell nuclei present in breast cancer tissue samples. The dataset includes the following attributes:

- **ID**: A unique identifier for each sample.
- **Diagnosis**: The class label, either Malignant (M) or Benign (B).
- **Mean Features**: 10 mean values of cellular characteristics, including radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension.
- **Standard Error (SE) Features**: 10 standard error values corresponding to the mean features.
- **Worst Features**: 10 worst-case values (largest mean values) for each of the cellular characteristics.

The dataset contains both categorical and numerical attributes:

- **Nominal Attribute**: *ID*, *Diagnosis* (Malignant (M) or Benign (B)).
- **Continuous Attribute**: 30 numerical features excluding ID and Diagnosis.

3.2 Data Preprocessing

We converted the Diagnosis column to binary (M=1, B=0) and drop the ID column and split the dataset into training and testing sets. The dataset has been split into:
Training set: 398 samples
Test set: 171 samples
The partial scatter matrix plot of the dataset is shown in Figure 6.

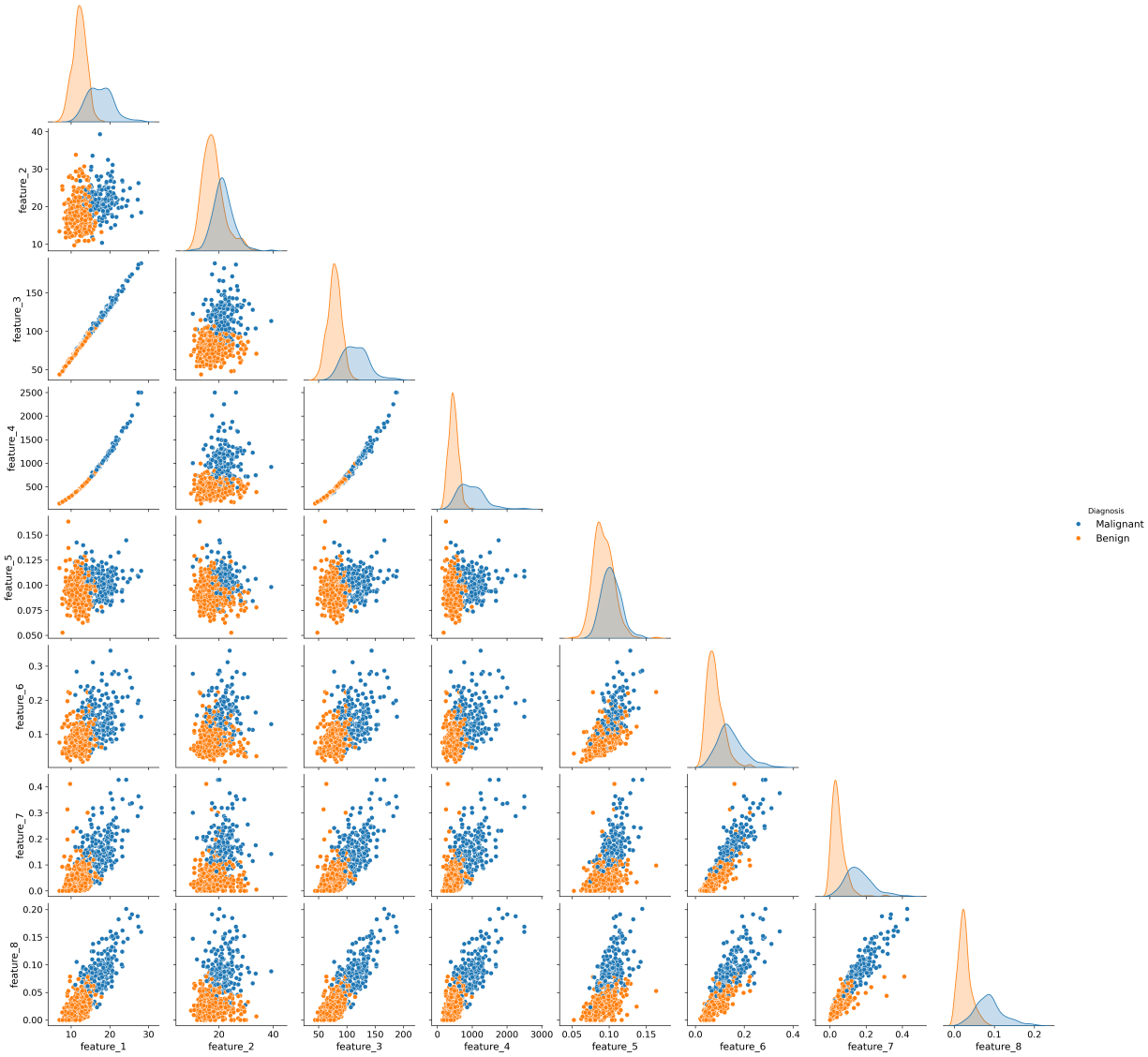


Figure 6: Scatter Matrix Plot of Breast Cancer (Continuous)

3.3 Decision Tree Construction

We train the same binary decision tree as in Problem 2 , but now using continuous data and computing evaluation metrics. The following parameter are set:

- `min_samples_leaf=2`
- `min_samples_split=5`
- `max_depth=2`

The decision tree trained on the original dataset is shown in Figure 7.

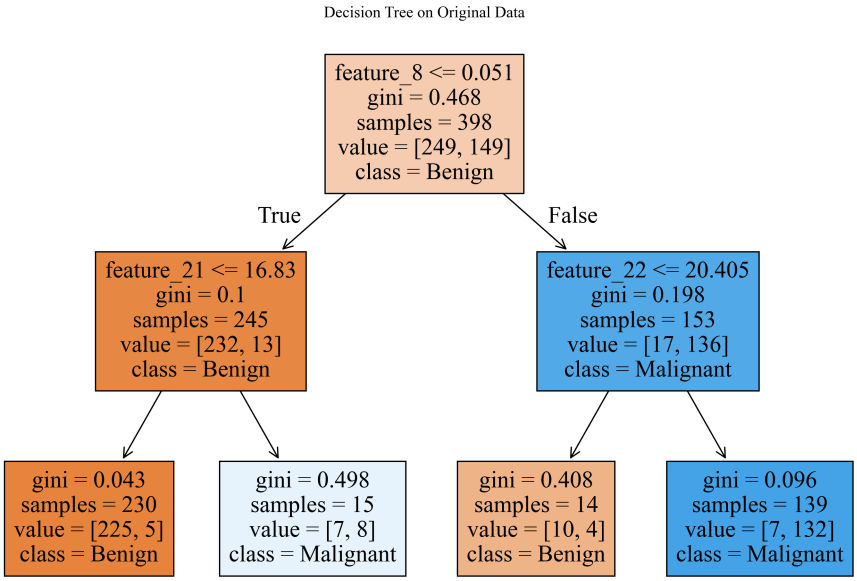
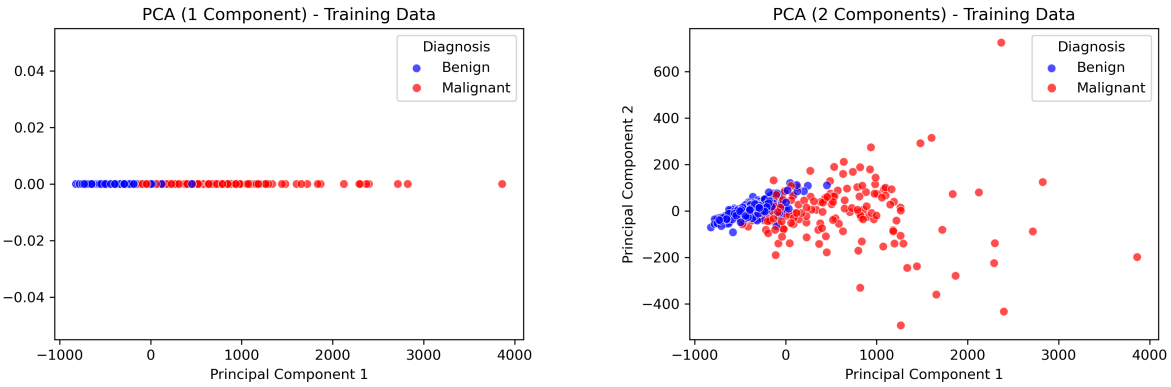


Figure 7: Tree Plot on Original Data

3.4 PCA Dimensionality Reduction

We apply Principal Component Analysis (PCA) to the original dataset and fit the model using the first principal component and the first two principal components, respectively, while computing the evaluation metrics. The visualization of the dataset after PCA transformation is shown in Figure 8.



(a) 1 Component

(b) 2 Component

Figure 8: Data distribution after PCA dimensionality reduction

The color represents the class distribution: blue for benign cases and red for malignant cases.

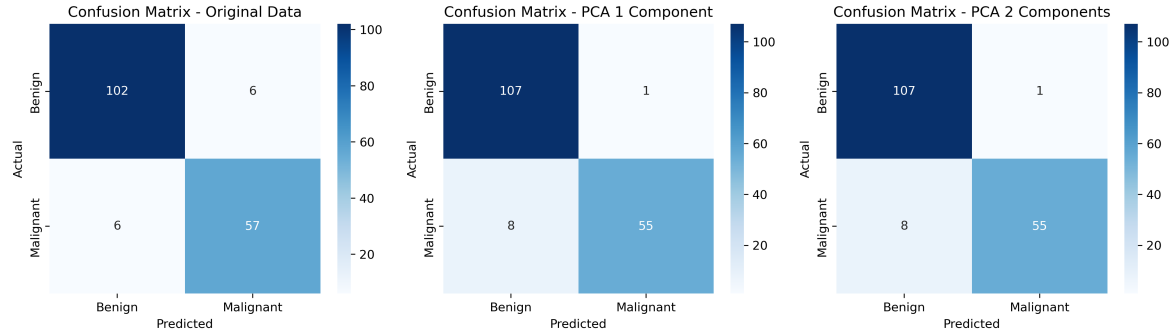
- In Figure 8a, since only one principal component is used, all data points are projected onto a single line, effectively reducing the original 30-dimensional data to a 1D space. The X-axis represents the first principal component (PC1), while the Y-axis is artificially set to 0 for

alignment. As seen in Figure 8a, the benign and malignant tumor samples are largely separated along the PC1 axis, indicating that the first principal component effectively differentiates the two classes.

- In Figure 8b, two principal components are used, and the data points are now distributed in a 2D plane. The X-axis represents the first principal component (PC1), and the Y-axis represents the second principal component (PC2). By observing Figure 8b, we can see that the two classes are well separated in this 2D space. The red and blue points are clearly distinguished, suggesting that the first two principal components successfully capture the major variance in the dataset and can be used for classification.

3.5 Result and Analysis

The confusion matrices for the decision tree trained on the original dataset, the dataset reduced to one principal component, and the dataset reduced to two principal components are shown in Figure 9.



(a) Original Data (b) PCA 1 Component (c) PCA 2 Component

Figure 9: Confusion Matrices

From Figure 9, it can be observed that compared to using the original continuous data, the number of diagonal elements increases while the number of off-diagonal elements decreases after applying PCA for dimensionality reduction. This indicates that the number of correctly classified samples has increased, and the number of misclassified samples has decreased. This demonstrates that PCA maintains good classification performance while reducing the dimensionality of the data. From Figures 9b and 9c, it can be seen that the results obtained using the first principal component and the first two principal components are identical. This suggests that most of the relevant information is contained within the first principal component.

The computed evaluation metrics for decision tree models trained on the original dataset and PCA-transformed datasets are summarized in Table 2.

Table 2: Metrics Calculation Results

	Precision	**Recall**	**F1 Score**	**FP**	**TP**	**FPR**	**TPR**
Original Data	0.905	0.905	0.905	6	57	0.056	0.905
PCA 1 Component	0.982	0.873	0.924	1	55	0.009	0.873
PCA 2 Components	0.982	0.873	0.924	1	55	0.009	0.873

By comparing the performance metrics in Table 2, we can observe:

1. Precision increased significantly after applying PCA (from 0.905 to 0.982), meaning the model made fewer false positive predictions.
2. Recall slightly decreased after PCA (from 0.905 to 0.873), indicating that the model missed a few more malignant cases.
3. F1 Score improved with PCA (from 0.905 to 0.924), showing a better balance between precision and recall.
4. False Positive Rate (FPR) decreased significantly after PCA (from 0.056 to 0.009), meaning the PCA-based models are much better at avoiding misclassifying benign cases as malignant.
5. True Positive Rate (TPR) dropped slightly after PCA (from 0.905 to 0.873), meaning fewer malignant cases were correctly identified.

3.6 Conclusion

1. Continuous data is beneficial if the goal is to maximize recall (detect all malignant cases).

   The original continuous data provided a higher recall (0.905) compared to the PCA-transformed data (0.873), meaning it was slightly better at identifying malignant cases.

   Continuous data retains all original features, allowing the decision tree to leverage the full information for classification.

2. If the goal is to minimize false positives and improve precision, PCA-based dimensionality reduction is beneficial.

   PCA improved precision (0.982 vs. 0.905) and significantly reduced the False Positive Rate (FPR: 0.009 vs. 0.056), making it more reliable in avoiding misclassification of benign cases as malignant.

   PCA reduces dimensionality, capturing the most important variance but potentially discarding some subtle patterns, leading to a slight recall drop.

   The high precision and low FPR in the PCA models suggest that PCA effectively filtered noise and redundant features, making the model more selective in identifying positive cases.

3. PCA reduced the dimensionality while still maintaining a strong classification performance.

   Using only one or two principal components provided nearly identical results, meaning most of the relevant information is captured in the first component.

4 Libraries Used

In this study, we utilized several Python libraries for data manipulation, visualization, machine learning model training, and evaluation. The key libraries used are:

- **NumPy** (`numpy`): Provides support for numerical operations and array manipulations.

- **Pandas** (`pandas`): Used for loading, processing, and analyzing structured data.
- **Matplotlib** (`matplotlib.pyplot`): Used for generating plots and visualizing data.
- **Seaborn** (`seaborn`): A statistical data visualization library that enhances Matplotlib plots.
- **Scikit-learn** (`sklearn`): A machine learning library used for:
  - `sklearn.datasets`: Loading standard datasets, such as the Iris dataset.
  - `sklearn.tree`: Constructing and visualizing decision trees.
  - `sklearn.model_selection`: Splitting datasets into training and testing sets.
  - `sklearn.metrics`: Evaluating model performance using accuracy, precision, recall, F1-score, and confusion matrices.
  - `sklearn.decomposition`: Performing Principal Component Analysis (PCA) for dimensionality reduction.
  - `sklearn.set_config`: Configuring settings for improved visualization.

All computations and visualizations in this study were performed using these libraries to ensure efficient data processing, model training, and performance evaluation.