VNUHCM

University of Science

Department of Information Technology

Project 2: Report

Topic: Decision Tree

Subject: Artificial Intelligence

Students:

Trần Phương Anh (22125004)

November 10th, 2024

Contents

1 Binary class dataset

1.1 Introduction

1.1.1 Objective

The purpose of the project is to classify breast tumors as malignant or benign using decision tree classifiers.

1.1.2 Dataset Overview

The Breast Cancer Wisconsin (Diagnostic) dataset [5] is a collection of medical data used for classification tasks, specifically to distinguish between **malignant** and **benign breast cancer tumors**. It contains **569 instances** (patients) with **30 features**. Class distribution: 357 benign, 212 malignant.

Features are derived from digitized images of fine needle aspirates (FNA) of breast mass samples. For each of the **10 primary features** describing the nucleus (e.g., radius, texture, perimeter, smoothness), there are three corresponding values: **the mean**, **the standard error**, and **the worst (largest) value** as mentioned in the dataset description file "wdbc.names".

Variables:

- ID Number (1): Unique identifier for each instance.
- Diagnosis (2): Indicates whether the tumor is malignant (M) or benign (B).
- Features (3-32): Ten real-valued features are computed for each cell nucleus from the FNA image:
  - Radius: Mean distance from the center to points on the perimeter.
  - Texture: Standard deviation of gray-scale values.
  - Perimeter
  - Area
  - Smoothness: Local variation in the radius lengths.
  - Compactness: Ratio of the perimeter squared to the area minus 1.

- Concavity: Severity of concave portions of the contour.
- Concave Points: Number of concave portions of the contour.
- Symmetry
- Fractal Dimension: A measure of the complexity of the contour.

1.2 Data Preparation

1.2.1 Loading and Preprocessing the Data

After loading the dataset, we perform data preprocessing:

1. **Remove the ID column**
2. **Encode the Diagnosis column**
   - Malignant (`M`) is mapped to 1.
   - Benign (`B`) is mapped to 0.

   This binary encoding allows us to treat the Diagnosis as a target variable for classification.
3. **Separate features and labels**:
   - Features: All columns except the Diagnosis column.
   - Labels: The values in the Diagnosis column, which act as the target variable for classification.

These preprocessing steps are implemented in Listing 1.

1.2.2 Description of Train/Test Splits

The dataset was shuffled and split into four proportions: 40/60, 60/40, 80/20, and 90/10.
To ensure that the class distribution in the target variable (Diagnosis) is maintained across the training and testing sets, we used **stratification** to ensure that the proportion of malignant and benign instances is approximately the same in both the training and testing sets.
Listing 2 demonstrates the splitting process.

1.2.3 Class distributions

Figure 1 summarizes class distributions for training and test sets across the original dataset and the four splits. We can see that the proportion of malignant and benign samples remains the same in both the training and test sets as in the original dataset.

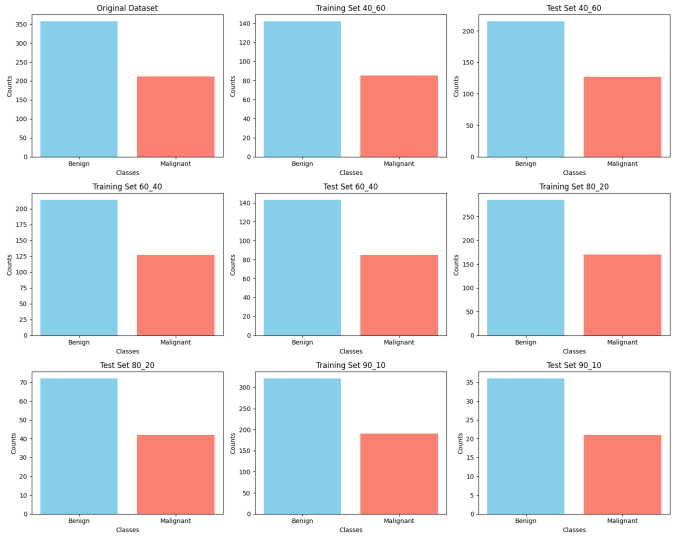

Figure 1: Distribution of each dataset split

1.3 Decision Tree Classifiers

1.3.1 Model Training

We used the **scikit-learn** library [1] to implement the model **DecisionTreeClassifier** with the "entropy" criterion for information gain. We set the random seed to 42 [4] to ensure the reproducibility of the results across different runs (there's no significance to this number). All other parameters are left at their default settings.

We iterated the training on each of the splits (40/60, 60/40, 80/20, 90/10) to evaluate the model's performance across different training sizes. The process implemented in Listing 3 includes:

- Training the classifier using the training set (X_{train}, y_{train}) for each split.
- Fitting the decision tree on the training data, which builds the model based on the feature set and target labels.
- Finally we evaluate the classifier on the test set (X_{test}, y_{test}) for each split.

1.3.2 Decision Tree Visualization

After training the decision tree classifier, we visualize the resulting tree structure for each of the training/test splits as shown in Listing 4.

- The **export_graphviz** function is used to export the trained decision tree into a DOT format, which is a text representation of the tree structure.
- **Graphviz** is used to render the tree into a visual format, which is saved as a PNG file for each split.

Each node in the tree represents a decision based on a specific feature, with edges representing the possible outcomes. The tree is filled with colors that correspond to the predicted class (malignant or benign).

By comparing the trees across different splits, we can observe changes in the model's complexity and its decision boundaries.

The Decision Tree shown in Figure 2 was trained on 40% of the data and tested on 60%.

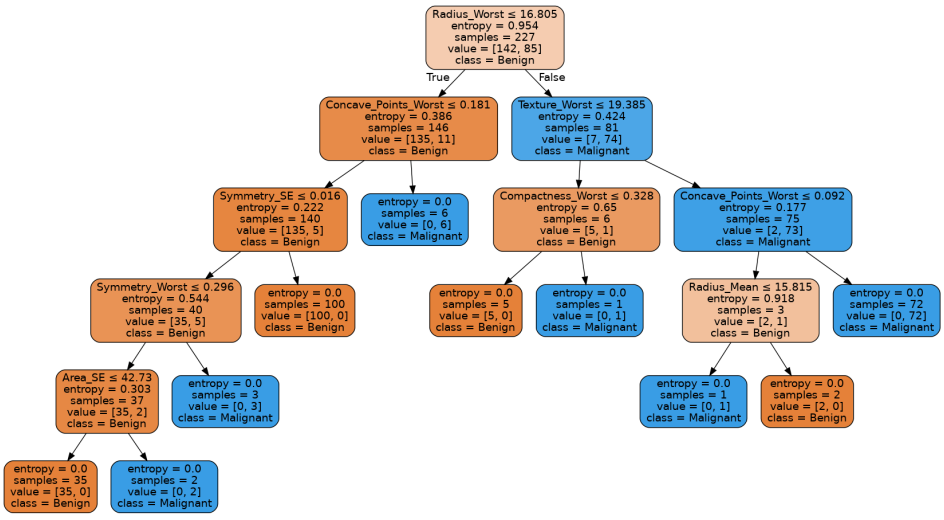

Figure 2: Decision Tree for the 40/60 Split

1.3.3 Statistical Results

The performance of the Decision Tree classifier was evaluated on four different train/test splits (40/60, 60/40, 80/20, and 90/10). The classification metrics, including **precision**, **recall**, **F1-score**, and **accuracy**, are presented for each split in Table 1 and the confusion matrices are presented in Graph 3.

Metric	40/60	60/40	80/20	90/10
Precision (Benign)	0.91	0.94	0.95	0.95
Precision (Malignant)	0.92	0.93	0.97	0.95
Recall (Benign)	0.96	0.96	0.99	0.97
Recall (Malignant)	0.83	0.91	0.90	0.90
F1-Score (Benign)	0.93	0.95	0.97	0.96
F1-Score (Malignant)	0.88	0.92	0.94	0.93
Accuracy	0.91	0.94	0.96	0.95
Macro Average F1	0.90	0.93	0.95	0.94
Weighted Average F1	0.91	0.94	0.96	0.95

Table 1: Performance Metrics for Decision Tree Classifier across Different Train/Test Splits

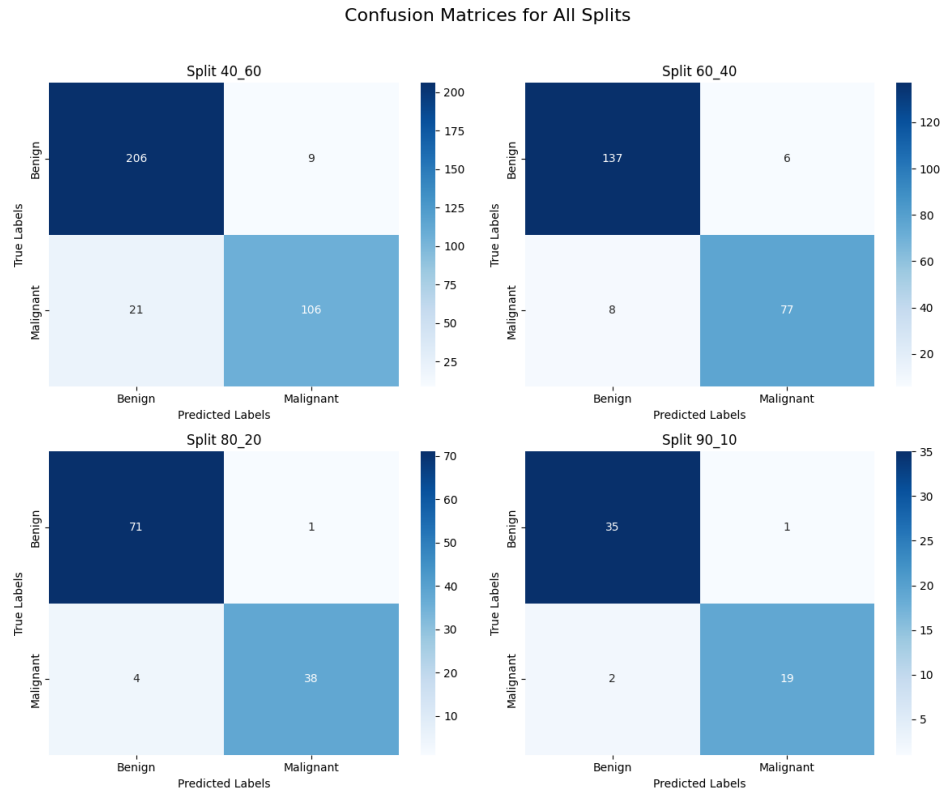

Figure 3: Confusion Matrices

Key Observations

- **Accuracy** improves with the size of the training set. The model achieves 91% accuracy in the 40/60 split, and this increases to 95% in the 90/10 split.
- **Precision** and **recall** for both classes are very high for all splits, with precision and recall for **Benign** being particularly high (around 0.95 to 0.97) in both the 80/20 and 90/10 splits. In the 40/60 split, the recall is lower for **Malignant**, 0.83, probably because there were fewer Malignant cases in the training data.
- **F1-Score**: The F1-score for **Benign** is higher than for **Malignant** in all splits.
- **Macro Average**: The **macro average** F1-score also increases with the size of the training set, indicating that the model becomes more consistent in performance between both classes as more data is available.
- **Confusion Matrices**: The confusion matrices show that there are fewer misclassifications as the model is trained on more data, especially in the **Malignant** class.

1.3.4 Insights

The decision tree classifier demonstrates high performance across all splits, but we need to keep these in mind:

- **Class Distribution:** The dataset is imbalanced, with a higher prevalence of benign cases (357) compared to malignant cases (212). This imbalance influences performance metrics, particularly precision and recall for the minority class (malignant).
- **Recall vs Precision:** In medical diagnosis, recall (sensitivity) for malignant cases is often prioritized. False negatives (misclassifying malignant cases as benign) are more critical than false positives, as they can delay necessary treatment.

Discussion on Overfitting and Generalization

As the training set size grows (e.g., in the 90/10 split), the classifier begins to show signs of overfitting. It achieves high precision but at the cost of slightly reduced recall for malignant cases. This behavior aligns with the characteristics of decision trees, which have a tendency to overfit in the

absence of pruning or regularization techniques. One of the methods we could look into is limiting tree depth to address overfitting.

1.4 Impact of Tree Depth

1.4.1 Experiment Setup

For this experiment, the dataset was split into an 80/20 ratio for training and testing. A range of `max_depth` values, including `None` (no restriction on depth) and integers from 2 to 7, was tested. The following steps were taken in Listing 5:

- **Model Training:** Decision trees were trained with the entropy criterion.
- **Evaluation:** Accuracy scores were computed for each depth using the test set.
- **Visualization:** Accuracy results were reported in both a table and a line plot for easy comparison.

The results are presented below.

1.4.2 Statistical Results

Table 2: Accuracy Scores for Different Max Depths

Max Depth	None	2	3	4	5	6	7
Accuracy	0.9561	0.8860	0.9386	0.9298	0.9561	0.9298	0.9561

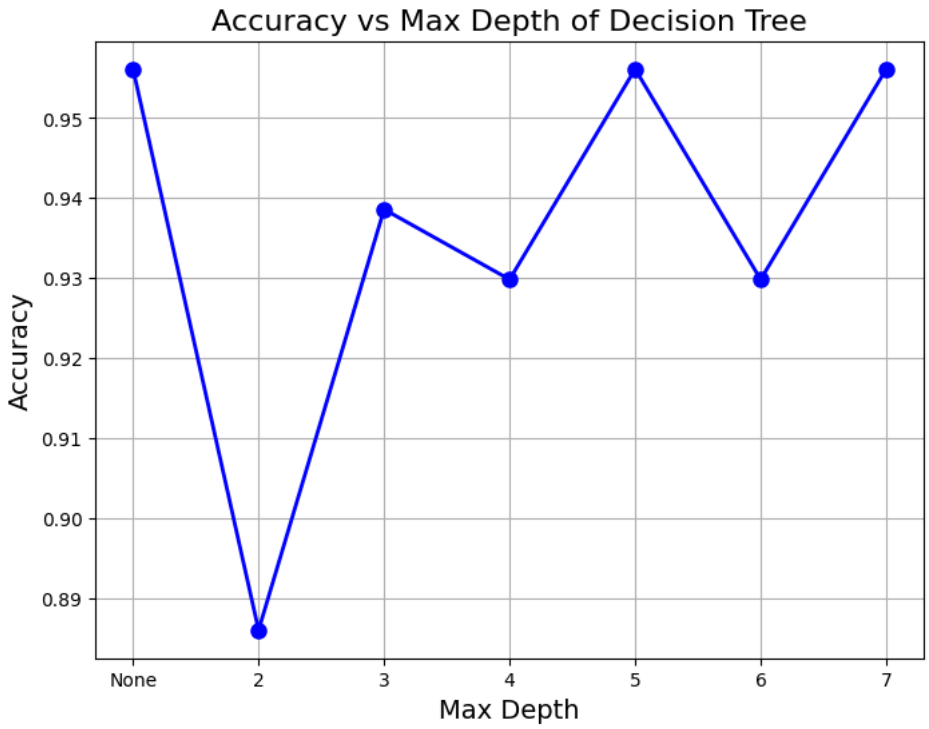

Figure 4: Accuracy vs Max Depth of Decision Tree

1.4.3 Insights

As the depth increases, the model captures more details of the training data, leading to improved accuracy. For example, the accuracy improves from 0.8860 at a depth of 2 to a maximum of 0.9561 at depths of None, 5, and 7. However, there are some trade-offs:

- **Underfitting:** At lower depths (e.g., 2), the decision tree lacks the complexity to model the data's patterns, resulting in reduced accuracy (0.8860).
- **Overfitting:** At higher depths, such as when the max depth is unrestricted (`None`), models usually risk overfitting. However, in this dataset, the performance remains high (0.9561).
- **Performance Plateau:** Depths of 5, 6, and 7 produce similar accuracies (0.9561 for depths 5 and 7, and 0.9298 for depth 6), indicating diminishing returns when further increasing depth.

1.4.4 Optimization

Based on the results, a max depth of **5** as shown in Figure 5 is the optimal choice. It achieves high accuracy (0.9561) while maintaining a balance between model complexity and generalization.

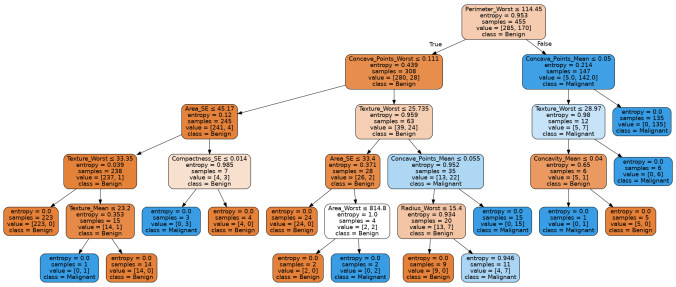

Figure 5: Decision Tree on Max Depth 5

Shallower depths, such as 2 or 3, result in underfitting, whereas deeper trees (e.g., 6 and beyond) provide no significant improvement in accuracy while increasing the likelihood of overfitting and computational inefficiency. Therefore, a depth of 5 is recommended for optimal performance on this dataset.

2 Multi-class dataset

2.1 Introduction

2.1.1 Objective

The purpose of this project is to classify wine samples into three quality categories (**Low**, **Standard**, and **High**) based on the physicochemical properties of white wine samples using decision tree classifiers.

2.1.2 Dataset Overview

Wine Quality dataset [3] is a collection of medical data used for classification tasks, specifically to distinguish between **malignant** and **benign breast cancer tumors**. It contains **4,898 white wine samples** with **11 features**. Wine quality scores are grouped into three categories:

- **Low quality:** Scores 0–4
- **Standard quality:** Scores 5–6
- **High quality:** Scores 7–10

The classes are imbalanced, with more samples in the **Standard** category than the **Low** or **High** categories:

- **Low Quality (0)**: 183 instances (3.7%)
- **Standard Quality (1)**: 3655 instances (74.8%)
- **High Quality (2)**: 1060 instances (21.7%)

Variables:

- **fixed_acidity:** Continuous
- **volatile_acidity:** Continuous
- **citric_acid:** Continuous
- **residual_sugar:** Continuous

- **chlorides:** Continuous
- **free_sulfur_dioxide:** Continuous
- **total_sulfur_dioxide:** Continuous
- **density:** Continuous
- **pH:** Continuous
- **sulphates:** Continuous
- **alcohol:** Continuous
- **quality:** Integer (score between 0 and 10)

2.2 Data Preparation

2.2.1 Loading and Preprocessing the Data

After loading the dataset, the following preprocessing steps were applied:

1. The dataset does not contain any missing values, so no cleaning was required.
2. The "quality" column was grouped into three categories:
    - **Low quality (0):** Scores 0–4
    - **Standard quality (1):** Scores 5–6
    - **High quality (2):** Scores 7–10
3. Features and labels were separated:
    - **Features:** All columns except "quality"
    - **Labels:** The "quality" column

These preprocessing steps are implemented in Listing 6.

2.2.2 Description of Train/Test Splits

The dataset was shuffled and split into four proportions: 40/60, 60/40, 80/20, and 90/10.
To ensure that the class distribution in the target variable (Quality) is maintained across the training and testing sets, we used **stratification** to ensure that the proportion of Low, Standard, and High instances is approximately the same in both the training and testing sets.
The splitting process is the same as what we do for dataset 1 so we reuse the code in Listing 2.

2.2.3 Class distributions

Figure 6 summarizes class distributions for training and test sets across the original dataset and the four splits. We can see that the proportion of Low, Standard, and High samples remains the same in both the training and test sets as in the original dataset.

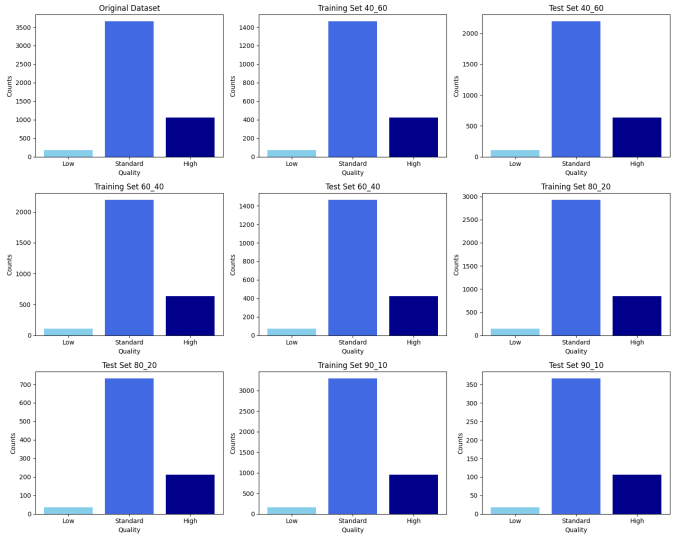

Figure 6: Distribution of each dataset split

2.3 Decision Tree Classifiers

2.3.1 Model Training

The model training process is similar to the approach used for the binary dataset, with adjustments for the multiclass nature of the target variable. So we also use the same code in Listing 3 to fit the model and predict the result.

2.3.2 Decision Tree Visualization

Similar to the previous dataset, after training the decision tree classifier, we visualize the resulting tree structure for each of the training/test splits as shown in Listing 7.

Since the Decision Tree for this multiclass dataset without constraints is too large, in this report we'll limit the tree's depth to 4 for visualization purposes, as shown in Figure 7.

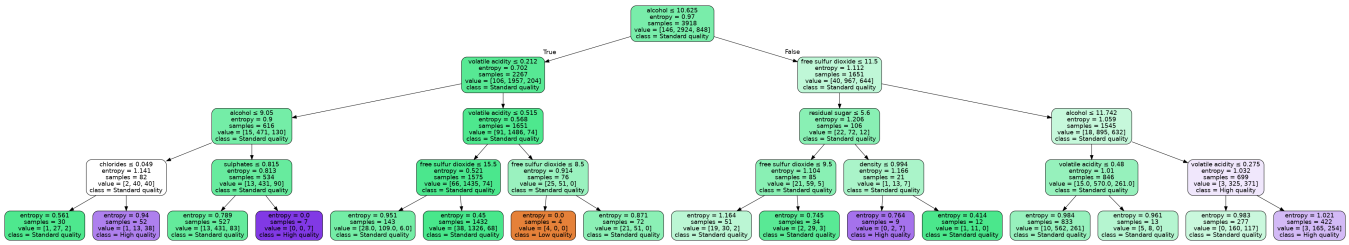

Figure 7: Decision Tree for the 80/20 Split, Max depth 4

2.3.3 Statistical Results

The performance of the Decision Tree classifier was evaluated on four different train/test splits (40/60, 60/40, 80/20, and 90/10). The classification metrics, including **precision**, **recall**, **F1-score**, and **accuracy**, are presented for each split in Table 3 and the confusion matrices are presented in Graph 8.

Metric	40/60	60/40	80/20	90/10
Precision (Low Quality)	0.26	0.25	0.31	0.32
Precision (Standard Quality)	0.84	0.85	0.87	0.89
Precision (High Quality)	0.55	0.58	0.62	0.60
Recall (Low Quality)	0.22	0.27	0.38	0.44
Recall (Standard Quality)	0.84	0.84	0.85	0.83
Recall (High Quality)	0.58	0.59	0.63	0.71
F1-Score (Low Quality)	0.24	0.26	0.34	0.37
F1-Score (Standard Quality)	0.84	0.85	0.86	0.85
F1-Score (High Quality)	0.57	0.59	0.63	0.65
Accuracy	0.76	0.77	0.79	0.79
Macro Average F1	0.55	0.56	0.61	0.63
Weighted Average F1	0.76	0.77	0.79	0.79

Table 3: Performance Metrics for Decision Tree Classifier on Multiclass Dataset across Different Train/Test Splits

Key Observations

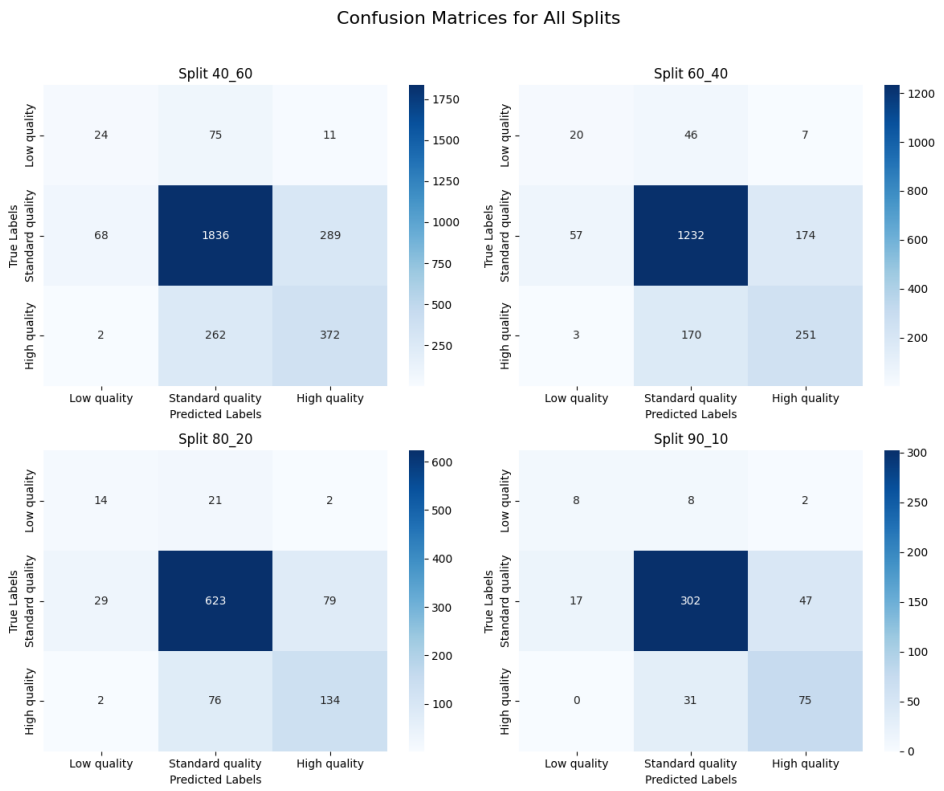

Figure 8: Confusion Matrices

- **Accuracy:** Accuracy improves as the training set size increases. It starts at 76% for the 40/60 split and increases to 79% in the 90/10 split. This suggests that more training data improves the model's overall performance.
- **Precision and Recall:**
  - **Low Quality (0):** Precision and recall are consistently low across all splits, with the best performance in the 90/10 split (precision: 0.32, recall: 0.44).
  - **Standard Quality (1):** Precision and recall are high across all splits, around 0.84 to 0.89.
  - **High Quality (2):** Precision and recall improve slightly with larger training sizes, with the best performance in the 90/10 split (precision: 0.60, recall: 0.71).
- **F1-Score:**
  - The F1-score for **Low Quality (0)** is the lowest across all splits.

  - For **Standard Quality (1)**, the F1-score is consistently high, around 0.84 to 0.85, due to balanced precision and recall.
  - The **High Quality (2)** class shows moderate improvement in F1-score as training data increases, peaking at 0.65 in the 90/10 split.

- **Macro Average:** The macro average F1-score improves as training size increases.
- **Weighted Average:** The weighted average F1-score aligns closely with accuracy, as the majority class dominates the performance metrics.
- **Confusion matrices:** We could see the model has a problem with differentiating between Standard quality and High quality, it could be due to how the classes don't have clear boundaries with each other.

2.3.4 Insights

The decision tree classifier demonstrates moderate performance across all splits, with overall accuracy ranging from 76% to 79%. However, the dataset is imbalanced, with the **Low Quality (0)** class significantly underrepresented. This imbalance skews the model towards favoring the majority class (**Standard Quality (1)**), resulting in poor precision and recall for the minority class.
A possible solution is pruning the decision tree or limiting its depth to prevent overfitting to the majority class.

2.4 Impact of Tree Depth

2.4.1 Experiment Setup

For this experiment, the dataset was split into an 80/20 ratio for training and testing. A range of `max_depth` values, including `None` (no restriction on depth) and integers from 2 to 7, was tested. The following steps were undertaken:

- **Model Training:** Decision trees were trained with the entropy criterion.
- **Evaluation:** Accuracy scores were computed for each depth using the test set.
- **Visualization:** Accuracy results were reported in both a table and a line plot for easy comparison.

The results are presented below.

2.4.2 Statistical Results

Table 4: Accuracy Scores for Different Max Depths

Max Depth	None	2	3	4	5	6	7
Accuracy	0.7867	0.7459	0.7582	0.7571	0.7786	0.7653	0.7745

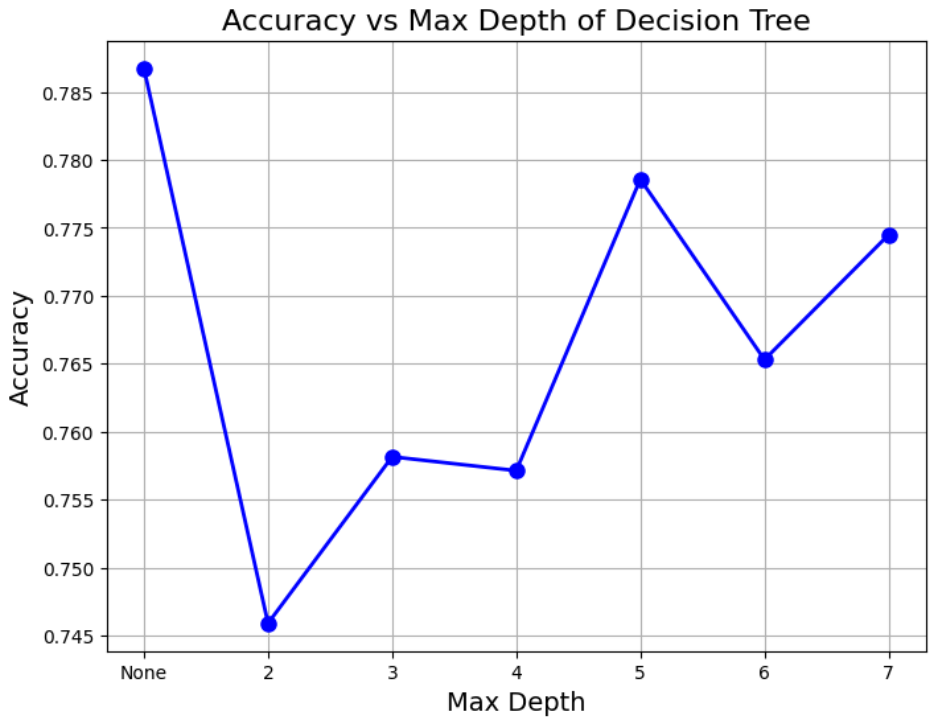

Figure 9: Accuracy vs Max Depth of Decision Tree

2.4.3 Insights

Impact of Increasing Depth on Accuracy: As the depth of the decision tree increases, the accuracy initially improves, reaching a peak at a certain depth before it starts to decline or stabilize.

Overfitting or Underfitting:

- **Underfitting**: At lower depths (e.g., depth 2), the model may underfit the data, resulting in lower accuracy.
- **Overfitting**: At higher depths (e.g., depth 7), the model may overfit the training data, capturing noise and leading to a slight decrease in accuracy on the test set.

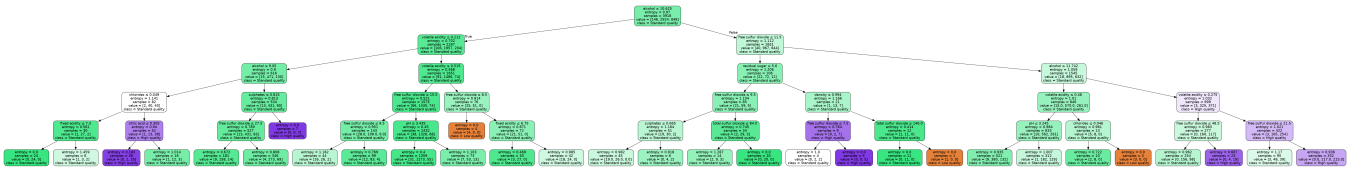

Figure 10: Decision Tree on Max Depth 5

2.4.4 Optimization

Based on the results, a depth of 5 as shown in Figure 10 appears to provide the best balance between bias and variance, achieving the highest accuracy without significant overfitting.

3 Additional dataset

3.1 Introduction

3.1.1 Objective

The objective of this project is to classify anemia types based on Complete Blood Count (CBC) data.

3.1.2 Dataset Overview

The **Anemia Classification dataset** [2] is a collection of medical data used to diagnose different types of anemia based on **Complete Blood Count (CBC)** parameters. The dataset contains **1,281 samples** with **14 features** and is labeled into eight initial classes.
Grouping of Classes: To simplify the classification process, the original eight classes were grouped into four broader categories:

- **Normal:** Healthy
- **Iron-Related Anemia:** Iron Deficiency Anemia, Other Microcytic Anemia
- **Non-Iron-Related Anemia:** Normocytic Hypochromic Anemia, Normocytic Normochromic Anemia, Macrocytic Anemia
- **Blood Disorders:** Thrombocytopenia, Leukemia, Leukemia with Thrombocytopenia

This grouping helps reduce the complexity of multi-class classification while retaining medical relevance.

variables:

- **HGB (Hemoglobin):** The amount of hemoglobin in the blood, crucial for oxygen transport.
- **PLT (Platelet Count):** The number of platelets in the blood, involved in blood clotting.
- **WBC (White Blood Cell Count):** The count of white blood cells, vital for immune response.
- **RBC (Red Blood Cell Count):** The count of red blood cells, responsible for oxygen transport.
- **MCV (Mean Corpuscular Volume):** Average volume of a single red blood cell.
- **MCH (Mean Corpuscular Hemoglobin):** Average amount of hemoglobin per red blood cell.
- **MCHC (Mean Corpuscular Hemoglobin Concentration):** Average concentration of hemoglobin in red blood cells.
- **PDW (Platelet Distribution Width):** A measurement of the variability in platelet size distribution in the blood.
- **PCT (Procalcitonin):** Helps diagnose sepsis or assess the risk of developing sepsis.
- **LYMp (Lymphocytes Percentage):** Percentage of lymphocytes in the blood.
- **NEUTp (Neutrophils Percentage):** Percentage of neutrophils in the blood.
- **LYMn (Lymphocytes Count):** Absolute count of lymphocytes in the blood.
- **NEUTn (Neutrophils Count):** Absolute count of neutrophils in the blood.
- **Diagnosis:** The anemia type or blood disorder based on the CBC parameters.

3.2 Data Preparation

3.2.1 Loading and Preprocessing the Data

After loading the anemia dataset, the following preprocessing steps were applied:

1. The dataset does not contain any missing values, so no cleaning was required.
2. The original eight classes in the "Diagnosis" column were grouped into four broader categories:
    - **Normal (0):** Healthy
    - **Iron-Related Anemia (1):** Iron Deficiency Anemia, Other Microcytic Anemia
    - **Non-Iron-Related Anemia (2):** Normocytic Hypochromic Anemia, Normocytic Normochromic Anemia, Macrocytic Anemia
    - **Blood Disorders (3):** Thrombocytopenia, Leukemia, Leukemia with Thrombocytopenia
3. Features and labels were separated:
    - **Features:** All CBC parameters.
    - **Labels:** The grouped "Diagnosis" column.

These preprocessing steps are implemented in Listing 8.

3.2.2 Description of Train/Test Splits

The dataset was shuffled and split into four proportions: 40/60, 60/40, 80/20, and 90/10. The splitting process follows the same procedure as previously implemented for other datasets, and the corresponding code is reused from Listing 2.

3.2.3 Class distributions

Figure 6 summarizes class distributions for training and test sets across the original dataset and the four splits. We can see that the proportion of Normal, Iron-Related Anemia, Non-Iron-Related Anemia, and Blood Disorder samples remains the same in both the training and test sets as in the original dataset.

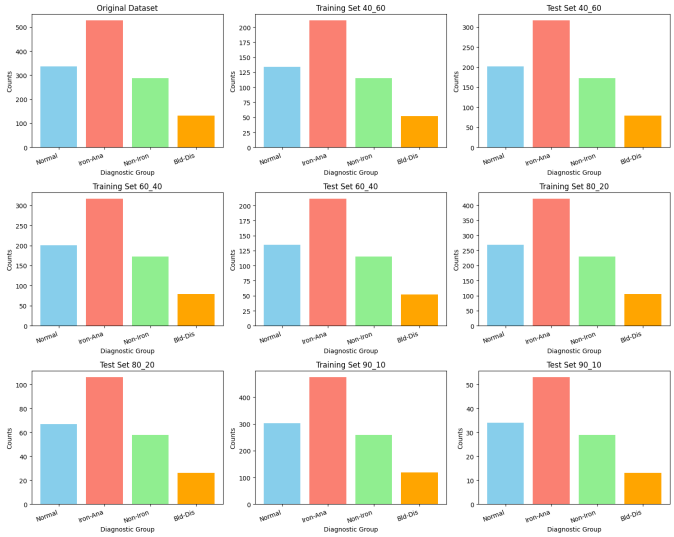

Figure 11: Distribution of each dataset split

3.3 Decision Tree Classifiers

3.3.1 Model Training

For the anemia dataset, we also use the same code in Listing 3 to fit the model and predict the result.

3.3.2 Decision Tree Visualization

Unlike the previous datasets, the full tree for this dataset is compact and easy to interpret. Figure 12 illustrates the complete decision tree for the 80/20 split, with a maximum depth of only 4 (no pruning or depth limit applied).

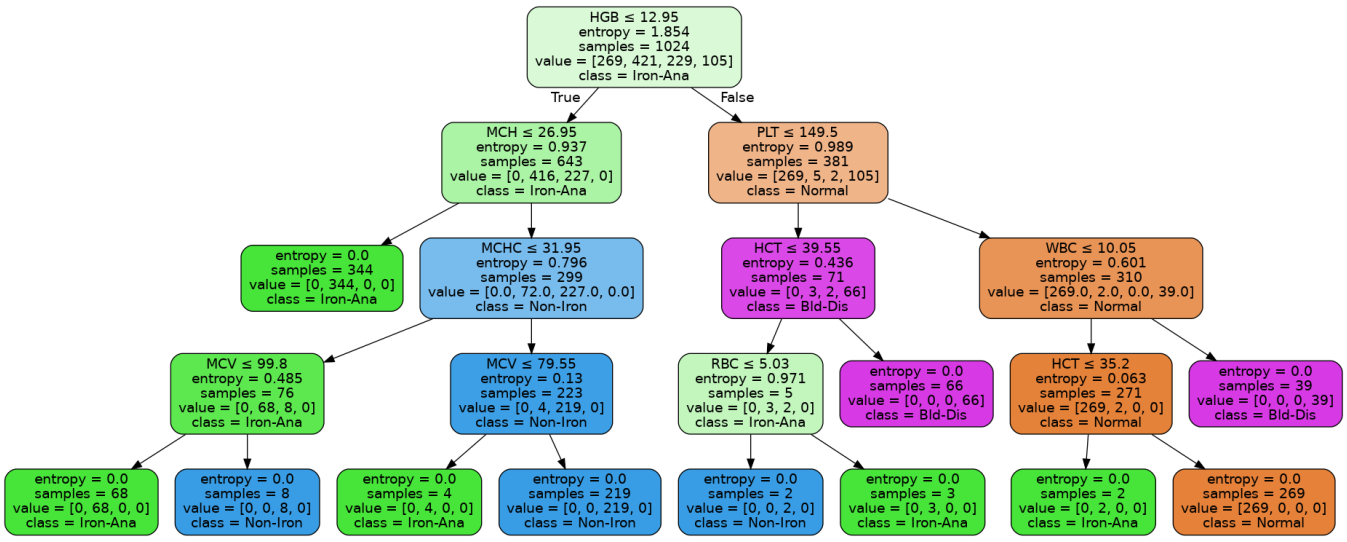

Figure 12: Decision Tree for the 80/20 Split

3.3.3 Statistical Results

The performance of the Decision Tree classifier was evaluated on three different train/test splits (60/40, 80/20, and 90/10). The classification metrics, including **precision**, **recall**, **F1-score**, and **accuracy**, are presented for each split in Table 5, and the confusion matrices are presented in Figure 13.

Metric	40/60	60/40	80/20	90/10
Precision (Normal)	1.00	0.99	1.00	1.00
Precision (Iron-Ana)	0.98	1.00	1.00	1.00
Precision (Non-Iron)	0.99	0.99	0.98	0.97
Precision (Bld-Dis)	1.00	1.00	1.00	1.00
Recall (Normal)	1.00	1.00	1.00	1.00
Recall (Iron-Ana)	0.99	0.99	0.99	0.98
Recall (Non-Iron)	0.97	1.00	1.00	1.00
Recall (Bld-Dis)	1.00	1.00	1.00	1.00
F1-Score (Normal)	1.00	1.00	1.00	1.00
F1-Score (Iron-Ana)	0.99	1.00	1.00	0.99
F1-Score (Non-Iron)	0.98	1.00	0.99	0.98
F1-Score (Bld-Dis)	1.00	1.00	1.00	1.00
Accuracy	0.99	1.00	1.00	0.99
Macro Average F1	0.99	1.00	1.00	0.99
Weighted Average F1	0.99	1.00	1.00	0.99

Table 5: Performance Metrics for Decision Tree Classifier on Anemia Dataset across Different Train/Test Splits

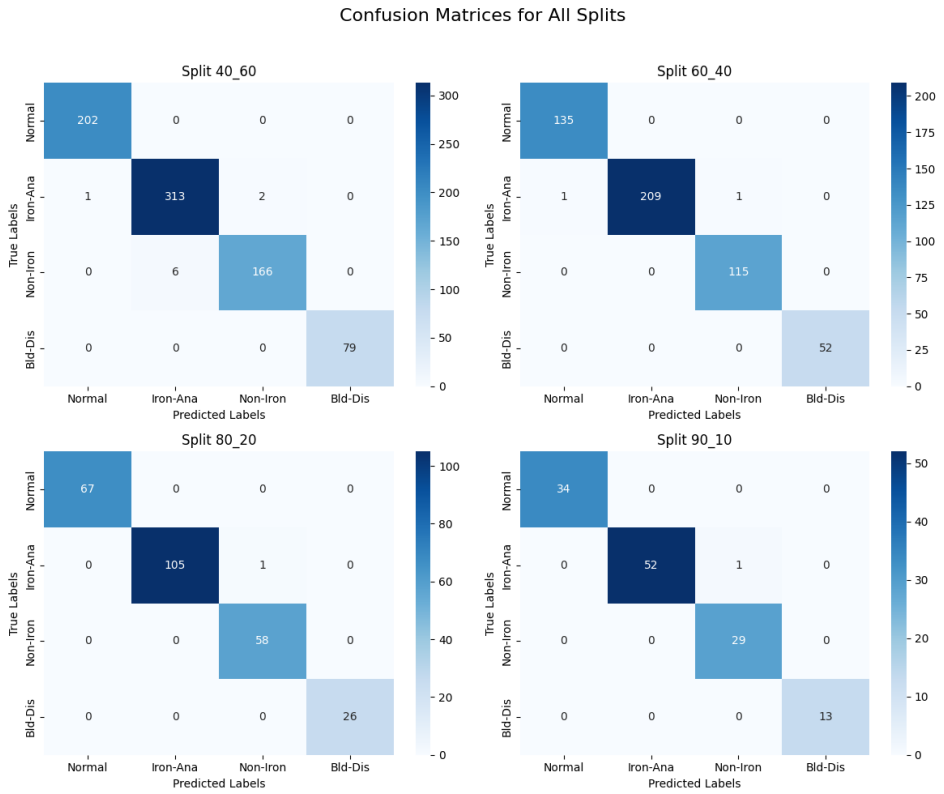

Figure 13: Confusion Matrices for Different Splits

Key Observations

- **Accuracy:** The Decision Tree classifier consistently achieves high accuracy across all train/test splits, delivering near-perfect performance even with smaller training datasets (e.g., the 40/60 split).
- **Precision and Recall:**
  - Precision and recall scores are consistently high for all categories.
  - In the 90/10 split, minor drops in precision and recall are noted for the **Non-Iron** and **Iron-Ana** categories, though both remain above 97%.
- **F1-Score, Macro and Weighted Average:** F1-scores are similar to precision and recall, staying close to 1.00 across all splits. Both macro and weighted averages for F1-scores remain exceptionally high across all splits.
- **Confusion Matrices:** Some confusion appears between "Iron-Related Anemia" and "Non-Iron-Related Anemia." However, these errors have a minimal effect on overall classification

performance.

3.3.4 Insight

The Decision Tree classifier achieves exceptional performance on the anemia dataset. The reason could be the simplicity of the classification problem within this dataset, where clear boundaries between diagnostic categories enable the Decision Tree to achieve high performance with minimal model complexity (maximum depth = 4). Further improvements in the model may not be necessary given these results.

3.4 Impact of Tree Depth

3.4.1 Experiment Setup

For this experiment, the anemia classification dataset was split into an 80/20 ratio for training and testing. The decision tree classifier was evaluated with a range of `max_depth` values, including `None` (no restriction on depth) and integers from 2 to 7. The following steps were performed:

- **Model Training:** Decision trees were trained using the entropy criterion for consistency with the previous experiment.
- **Evaluation:** Test set accuracy was calculated for each depth setting.
- **Reporting:** Results were summarized in Table 6.

3.4.2 Statistical Results

Table 6: Accuracy Scores for Different Max Depths

Max Depth	None	2	3	4	5	6	7
Accuracy	0.9961	0.9066	0.9883	0.9961	0.9961	0.9961	0.9961

3.4.3 Insights

Unlike many datasets where restricting decision tree depth balances bias and variance, the anemia dataset achieves near-perfect accuracy even when the tree depth is unrestricted.

- At `max_depth = 2`, the model accuracy drops significantly to 90.66%, likely due to underfitting caused by insufficient tree depth to capture the complexity of the data.

- For depths of 3 and above, accuracy increases rapidly, stabilizing at 99.61% from `max_depth = 4` onward, equaling the performance of an unrestricted tree.

3.4.4 Optimization

Depth optimization is unnecessary for the anemia dataset, as unrestricted trees perform equivalently to those with `max_depth` ≥ 4. This is in contrast to more complex or noisy datasets, where limiting tree depth is critical to avoid overfitting. These results highlight the simplicity and separability of the anemia classification task.

4 Dataset Influence on Decision Tree Performance

4.1 Characteristics of Datasets

- **Binary Class Dataset (Breast Cancer Wisconsin):**
  - **Classes:** 2 (Malignant, Benign)
  - **Features:** 30 numeric features derived from imaging data
  - **Sample Size:** 569 samples
- **Multi-class Dataset (Wine Quality):**
  - **Classes:** Grouped into 3 categories (Low, Standard, High quality)
  - **Features:** 11 physicochemical properties
  - **Sample Size:** 4,898 samples
- **Anemia Classification Dataset:**
  - **Classes:** Grouped into 4 categories (Normal, Iron-Related Anemia, Non-Iron-Related Anemia, Blood Disorders)
  - **Features:** 14 CBC parameters
  - **Sample Size:** 1,281 samples

4.2 Impact of Dataset Characteristics on Performance

Number of Classes:

- **Binary Class Dataset:** The Breast Cancer dataset achieved consistently high accuracy ($\geq 90\%$) across all `max_depth` values. This performance is because there are only two clear and separate classes.
- **Multi-class Dataset:** The Wine Quality dataset had lower accuracy than the binary dataset because it included three broader classes. These extra classes caused more overlap and made the analysis more complex. For example, if two samples have very similar parameters, they could be rated as 6 and 7, which would categorize them as standard and high quality, respectively, even though the samples may not differ significantly.
- **Multi-class Dataset with Grouping (Anemia):** Despite having four classes, the Anemia dataset achieved the highest accuracy. This suggests that the dataset's features and class separability, in contrast to the previous dataset, is clearer.

Number of Features:

- **Breast Cancer Dataset:** With 30 features, this dataset has a lot of information to make accurate splits at shallow depths.
- **Wine Quality Dataset:** The 11 features in this dataset captured some variability but were less discriminative compared to the Breast Cancer dataset, resulted in slightly lower accuracies and more chance to overfit in deeper trees.
- **Anemia Dataset:**The 14 features, based on clear CBC parameters, worked well to tell different groups apart. This allowed the decision tree to keep high accuracy scores no matter how deep the tree went.

Sample Size:

- **Breast Cancer Dataset:** The relatively small sample size (569 samples) reduces the risk of overfitting, as the decision tree had fewer data points to memorize. However, with limited data, the results of this model may not apply well to larger test datasets.

- **Wine Quality Dataset:** With 4,898 samples, the larger dataset size also increased the computational complexity. This makes it more important to optimize the depth of the tree.
- **Anemia Dataset:** The moderate sample size (1,281 samples) provides enough training data while keeping complexity manageable. The clear separation between classes helps reduce problems that can come with smaller groups.

4.3 Summary of Results

- **Accuracy Trends:**
  - Binary datasets, such as Breast Cancer, achieve high accuracy due to the simplicity of the classification task and well-separated classes.
  - Multi-class datasets, such as Wine Quality, are more prone to classification errors, particularly when class overlap is significant or feature discriminative power is limited.
- **Optimal Depth:** Restricting tree depth improves performance in complex multi-class datasets like Wine Quality by reducing overfitting. However, unrestricted depth is effective for simpler datasets like Breast Cancer and for well-separated datasets like Anemia.
- **Feature Importance:** Datasets with a larger number of meaningful features, such as Breast Cancer (30 features) and Anemia (14 features), tend to perform better than those with fewer or less discriminative features, such as Wine Quality (11 features).

4.4 Conclusion

The performance of decision trees is determined by a combination of dataset characteristics, including the number of classes, features, and sample size. Binary classification tasks with fewer and well-separated classes consistently result in higher accuracy. In contrast, multi-class tasks demand greater feature discriminability and effective class grouping to attain comparable performance. These findings emphasize the importance of thoroughly understanding dataset properties to design and optimize decision tree models effectively.

5 Appendices

Include any additional information, such as: The exact commands used for exporting visualizations. Details about the computational environment (e.g., Python version, libraries used).

5.1 Dataset 1

Listing 1 Preprocessing steps

```
# Remove ID column (not needed for classification)
data = data.drop(columns=["ID"])

# Encode Diagnosis column: M -> 1 (malignant), B -> 0 (benign)
data["Diagnosis"] = data["Diagnosis"].map({"M": 1, "B": 0})

# Separate features (X) and labels (y)
# Features are all columns except 'Diagnosis'
features = data.drop(columns=["Diagnosis"]).values

# Labels are the 'Diagnosis' column
labels = data["Diagnosis"].values
```

Listing 2 Shuffle and Split the Dataset

```
from sklearn.model_selection import train_test_split

# Splits and stratification
proportions = [40, 60, 80, 90]
splits = {}
for prop in proportions:
    X_train, X_test, y_train, y_test = train_test_split(
        features, labels, train_size=prop/100, stratify=labels, random_state=42
    )
    splits[f"{int(prop)}_{int(100-prop)}"] = (X_train, X_test, y_train, y_test)
```

Listing 3 Building the decision tree classifiers and predicting the result

```
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Train and evaluate decision trees for all splits
for key, (X_train, X_test, y_train, y_test) in splits.items():
    clf = DecisionTreeClassifier(criterion="entropy", random_state=42)
    clf.fit(X_train, y_train)

    # Make predictions on the test set
    y_pred = clf.predict(X_test)
```

Listing 4 Visualize the decision tree

```
from sklearn.tree import DecisionTreeClassifier, export_graphviz
import graphviz

# Train and visualize decision trees
for key, (X_train, X_test, y_train, y_test) in splits.items():
    clf = DecisionTreeClassifier(criterion="entropy", random_state=42)
    clf.fit(X_train, y_train)

    # Export the tree
    dot_data = export_graphviz(
        clf,
        out_file=None,
        feature_names=data.drop(columns=["Diagnosis"]).columns,
        class_names=["Benign", "Malignant"],
        filled=True,
        rounded=True,
        special_characters=True
    )
    graph = graphviz.Source(dot_data)
    graph.render(f"plots/dataset1/decision_tree_{key}", format="png")
```

Listing 5 Building the decision tree classifiers for each max depth value

```
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn.metrics import accuracy_score
import graphviz

max_depth_values = [None, 2, 3, 4, 5, 6, 7]
accuracy_scores = []

# Prepare data (use 80/20 split)
X_train, X_test, y_train, y_test = splits["80_20"]

for max_depth in max_depth_values:
    # Train the decision tree classifier with the specified max_depth
    clf = DecisionTreeClassifier(criterion="entropy", max_depth=max_depth, random_state=42)
    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)
    accuracy_scores.append(accuracy)

    # Export the tree
    dot_data = export_graphviz(
        clf,
        out_file=None,  # Output to string (DOT format)
        feature_names=data.drop(columns=["Diagnosis"]).columns,  # Feature names
        class_names=["Benign", "Malignant"],  # Target classes
        filled=True,  # Colorize nodes
        rounded=True,  # Rounded corners
        special_characters=True
    )

    graph = graphviz.Source(dot_data)
    graph.render(f"plots/dataset1/decision_tree_max_depth_{max_depth}", format="png")
    print(f"Decision Tree Graph for Max Depth {max_depth}:")
    display(graph)
```

5.2 Dataset 2

Listing 6 Preprocessing steps

```
# Group quality into 3 categories
def categorize_quality(score):
    if score <= 4:
        return 0 #Low
    elif 5 <= score <= 6:
        return 1 #Standard
    else:
        return 2 #High

data["quality"] = data["quality"].apply(categorize_quality)

# Split data into features and labels
features = data.drop(columns=["quality"])  # Features
labels = data["quality"]  # Labels
```

Listing 7 Visualize the decision tree

```
for key, (X_train, X_test, y_train, y_test) in splits.items():
    clf = DecisionTreeClassifier(criterion="entropy", random_state=42)
    clf.fit(X_train, y_train)

    # Export
    dot_data = export_graphviz(
        clf,
        out_file=None,
        feature_names=data.drop(columns=["quality"]).columns,
        class_names=["Low quality", "Standard quality", "High quality"],
        filled=True,
        rounded=True,
        special_characters=True
    )

    graph = graphviz.Source(dot_data)
    graph.render(f"plots/dataset2/decision_tree_{key}", format="png")
    print(f"Decision Tree Graph for Split {key}:")
```

5.3 Dataset 3

Listing 8 Preprocessing steps

```
# Group and encode the 'Diagnosis' column
def group_diagnosis(diagnosis):
    if diagnosis == "Healthy":
        return 0  # Normal
    elif diagnosis in ["Iron deficiency anemia", "Normocytic hypochromic anemia", "Other mi
        return 1  # Iron-Related Anemia
    elif diagnosis in ["Normocytic normochromic anemia", "Macrocytic anemia"]:
        return 2  # Non-Iron-Related Anemia
    else:
        return 3  # Blood Disorders

data['Diagnosis'] = data['Diagnosis'].apply(group_diagnosis)

# Separate features and labels
features = data.drop(columns=["Diagnosis"]).values  # CBC parameters
labels = data["Diagnosis"].values  # Encoded Diagnosis labels
```

References

[1]

[2] Ehab Aboelnaga. Anemia types classification, May 2024.

[3] Cerdeira A. Almeida F. Matos T. Cortez, Paulo and J. Reis. Wine Quality. UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C56S3T.

[4] Nazia Nafis. The story behind “random.seed(42)” in machine learning, Jan 2023.

[5] Mangasarian Olvi Street Nick Wolberg, William and W. Street. Breast Cancer Wisconsin (Diagnostic). UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.