Thesis

Implementation and Validation of a Decision Tree-Based Approach for Interpretable Supervised Clustering

Author: Michele Guderzo

Padua, Academic Year 2024/25

Abstract

Statistical learning is a fundamental field that unifies statistical principles and machine learning approaches to enable data-driven analysis and prediction. Among its key techniques, clustering represents a pivotal task, allowing the identification of homogeneous groups within a dataset. While traditional clustering methods often prioritize accuracy and efficiency, interpretability remains a crucial aspect, particularly in domains where understanding the reasoning behind classifications is essential. This thesis explores the integration of decision trees into a supervised clustering framework, aiming to enhance transparency as well as effectiveness in cluster formation.

The proposed method is based on a novel algorithm called *Best Node Selection*, which leverages decision trees to iteratively partition the dataset. At each iteration, a decision tree of fixed depth is trained, and the best node, along with its decision path, is selected using the F-score metric as the primary criterion. This node is then removed from the dataset, and the process continues on the remaining data, ensuring an interpretable step-by-step clustering approach.

The algorithm has been implemented in Python and evaluated on the Breast Cancer Wisconsin (Diagnostic) dataset. The results demonstrate its ability to identify well-defined clusters while maintaining a clear selection process. However, a tendency to favor small, highly pure nodes raises concerns regarding generalizability and robustness. This trade-off between interpretability and cluster representativeness suggests avenues for further refinement.

Future developments could focus on improving the node selection strategy to balance purity and sample size, integrating external knowledge, and exploring applications to different datasets to assess the method's adaptability and effectiveness across various domains.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Statistical learning is a key area of study within data analysis, statistics, and machine learning that focuses on developing methods and algorithms to extract valuable insights from data. Broadly speaking, it ”refers to a vast set of tools for understanding data” (James et al., 2013). The field plays a significant role in solving complex problems by identifying relationships among variables, making predictions, or deriving general patterns in large datasets. Its relevance is growing, especially in today’s world, where data availability is increasing and computational power continues to expand.

Although the concept of statistical learning is relatively recent, its theoretical foundations extend back in time. The field’s earliest developments emerged at the beginning of the 19th century, when Legendre and Gauss published their work on the *method of least squares*, establishing the foundation for *linear regression*. To address the need for predicting qualitative outcomes, rather than the quantitative values provided by linear regression, researchers developed *logistic regression*. In the early 1970s, Nelder and Wedderburn unified these approaches under the term *generalized linear models*, encompassing a broad range of statistical learning methods. While researchers developed various techniques during this period, they remained constrained to linear methods due to limited computational capabilities, which prevented the exploration of *non-linear* relationships. This limitation persisted until the late 20th century, when advances in computational technology finally enabled the investigation of non-linear methods. This progress led to the development of *classification and regression trees*, followed by *generalized additive models*—introduced by Hastie and Tibshirani in 1986 as a non-linear extension of generalized linear models. Since then, ongoing advancements in computer science and statistical software have steadily increased the significance of statistical learning, further enhanced by the remarkable emergence of *machine learning*.

Machine learning represents a field closely related to statistical learning. It is recognized as a branch of *artificial intelligence* that employs statistical methods to develop algorithms for making accurate data-driven predictions. Through the analysis of large amount of data, machine learning enables systems to learn and improve autonomously without requiring explicit programming.

"Since learning involves an interaction between the learner and the environment, one can divide learning tasks according to the nature of that interaction", as reported by Shalev-Shwartz and Ben-David (2014). Based on this interaction principle, the field of machine learning distinguishes between two fundamental approaches: *supervised learning* and *unsupervised learning*. In supervised learning, the goal is to predict an output variable (also called the *target*) based on one or more input variables (also called *features*). The model is trained using labeled data, meaning that for each observation in the dataset, both the inputs and the corresponding outputs are known. Examples of supervised learning include regression problems, like predicting house prices, and classification problems, such as classifying images into predefined categories. Conversely, unsupervised learning deals with unlabeled data, aiming to uncover hidden patterns or structures without predefined outcomes. Common tasks include clustering and dimensionality reduction, which help group similar observations or simplify datasets, respectively.

Among these tasks, *clustering* is a widely used technique in unsupervised learning. The concept behind clustering is effectively expressed by Hastie et al. (2008): "all relate to grouping or segmenting a collection of objects into subsets or *clusters*, such that those within each cluster are more closely related to one another than objects assigned to different clusters". Clustering serves diverse purposes, such as customer segmentation, pattern recognition, and anomaly detection, providing valuable insights for decision-making in business, science, and technology.

This study explores a less conventional yet promising approach to clustering, known as *supervised clustering*. Unlike traditional clustering techniques, which operate in unsupervised learning environments with unlabeled data and specific error functions, supervised clustering works with pre-classified examples to identify clusters that exhibit a high probability density for a given class. This methodology is characterized by two key features: it prioritizes a parsimonious number of clusters and assigns elements to groups based on their proximity using a defined distance function. Through this approach, the resulting clusters achieve both internal coherence and alignment with meaningful external criteria, enhancing interpretability and practical relevance.

This thesis further investigates an interpretable approach to supervised clustering founded on *decision trees*, a widely adopted supervised learning methodology renowned for its simplicity and comprehensibility. Decision trees model the decision-making process through a hierarchical structure of binary splits based on feature values, creating an intuitive tree-like representation. By adapting this established framework to the clustering domain, we can create clusters that are not only data-driven but also readily interpretable. This integration yields a particularly valuable analytical tool for domains where interpretability is as crucial as performance.

1.2 Literature Review

In recent years, the focus on *clustering interpretability*—both in the supervised and unsupervised domains—has often been overshadowed by the pursuit of efficiency and accuracy, which are frequently prioritized. However, in fields such as finance, technology, and medicine, the need for interpretability in clustering results has become increasingly critical, making it a key factor in the process of creating homogeneous clusters from data.

The work of Hu et al. (2024) specifically examines methods that prioritize cluster interpretability. The authors categorize interpretable clustering techniques into pre-clustering, in-clustering, and post-clus-tering approaches, emphasizing the role of decision trees, rules, prototypes, and convex polyhedral models in balancing interpretability and clustering performance. They highlight how supervised constraints influence the clustering process, shaping the trade-off between explainability and accuracy.

Eick et al. (2004) extend this idea by proposing a fitness-based supervised clustering framework, where clusters are optimized based on class purity and compactness. Their work demonstrates the effectiveness of *evolutionary computing algorithms* (SCEC) in improving cluster formation, particularly when ”greedy” methods fail to find globally optimal solutions. These approaches lay the foundation for modifying clustering methodologies to incorporate supervision, leading to improvements in both classification accuracy and understanding of datasets.

A different adaptation of clustering techniques is explored by Al-Harbi and Rayward-Smith (2006), who modify *k-means*, a traditionally unsupervised algorithm, for supervised clustering. Their approach adjusts the distance metric, optimizing it through *Simulated Annealing* to ensure better class separation. Their results indicate that incorporating supervised information into distance-based clustering algorithms can significantly improve clustering performance, reinforcing the importance of metric learning in supervised clustering.

Moving beyond distance-based adaptations, Finley and Joachims (2005) introduce *support vector machines* (SVMs) for supervised clustering, formulating the problem as a structured prediction task. Their method groups similar data points into clusters while simultaneously optimizing classification boundaries, proving particularly effective when class distributions are non-linearly separable. This highlights the potential of margin-based clustering approaches, which go beyond simple proximity measures.

Dettling and Bühlmann (2002) apply supervised clustering to gene expression data, proposing an algorithm that directly incorporates information on cancer types into the clustering process. Unlike traditional unsupervised techniques, which group genes based solely on similarity, their method identifies functionally relevant gene clusters for medical diagnostics. They demonstrate that the proposed supervised clustering approach yields more stable and predictive clusters, often outperforming state-of-the-art classification techniques based on single genes. Their empirical validation on eight microarray datasets confirms that supervised clustering can be a powerful tool in functional genomics, improving interpretability alongside classification accuracy.

Expanding on the concept of supervision, Gao et al. (2024) introduce a *semi-supervised clustering* framework designed for scenarios where labeled data is limited. Their approach integrates probabilistic priors and domain knowledge to improve the detection of hazardous aviation winds, demonstrating that semi-supervised clustering can outperform fully supervised approaches in low-label environments. This study emphasizes how unlabeled data can be leveraged to enhance generalization, bridging the gap between fully supervised and unsupervised methods.

Ravenda et al. (2024) further extend clustering techniques to *self-supervised learning*, applying them to topic modeling. Their seed-driven Bayesian model (SSBM) integrates word embeddings and probabilistic clustering, improving upon traditional methods such as *Latent Dirichlet Allocation* (LDA). By combining BERT-based embeddings with clustering techniques, SSBM achieves better topic coherence and interpretability, demonstrating that clustering can benefit from linguistic representations and self-supervised learning principles.

Finally, a domain-specific adaptation of supervised clustering is presented by de Boer et al. (2024), who develop *SurvivalLVQ*, a method designed for survival analysis. Unlike generic clustering methods, which often fail to incorporate event-time information, SurvivalLVQ extends *Generalized Matrix Learning Vector Quantization* (GMLVQ) by assigning each prototype a survival curve instead of a fixed class label. This enables both interpretable clustering and individualized survival predictions, ensuring that clusters align with meaningful survival patterns. The analysis of 76 survival datasets validates the effectiveness of supervised clustering techniques in survival analysis domains, combining easily interpretable outcomes with robust predictive performance.

1.3 Aim and Objectives

The aim of this thesis is to develop an interpretable supervised clustering method based on decision trees, striving for high classification accuracy while enhancing interpretability. To achieve this, the study proposes to develop a supervised clustering algorithm that leverages decision trees to generate structured and explainable clusters. The algorithm's performance is then evaluated on a labeled dataset, considering both accuracy and interpretability. Furthermore, the study assesses its practical applicability in real-world domains and concludes by exploring limitations and potential improvements to guide future research.

1.4 Thesis Structure

The remainder of this thesis is organized as follows.

Chapter 2 (*Methodology*) describes the proposed approach in detail, outlining the algorithms and the performance metrics employed to achieve the thesis objectives.

Chapter 3 (*Data Analysis and Algorithm Evaluation*) provides a description of the dataset used, along with an exploratory data analysis (EDA) to understand the data structure. It also provides a detailed representation of the obtained results based on specific values for hyperparameters.

Chapter 4 (*Discussion*) interprets the results in relation to the study purposes, highlighting key findings, potential limitations, and implications.

Lastly, Chapter 5 (*Conclusions and Future Work*) summarizes the main contributions of this work and discusses possible directions for future research.

Chapter 2

Methodology

2.1 Overview

This chapter focuses extensively on the algorithm's design and implementation, as they constitute the foundation of this thesis. In particular, its operation and the evaluation metrics used are described in detail. The algorithm is based on the study of Kokash and Makhnist (2024), *Using Decision Trees for Interpretable Supervised Clustering*, from which its implementation was structured by adopting the methodologies and indicators proposed by the authors.

From a broader perspective, the implemented algorithm iteratively selects the most relevant groups (nodes) generated by a decision tree-based procedure and identifies the splitting features that characterize them. The objective is to determine the variables that defines the best nodes, facilitating a more intuitive and immediate interpretation of the data.

2.2 Algorithm Explanation

2.2.1 Decision Tree

To begin with, after importing the necessary libraries and loading the dataset, a first decision tree is trained and displayed. As briefly introduced in Chapter 1, a *decision tree* is a type of machine learning algorithm used for both classification and regression tasks. It is a tree-like model that represents decisions and their possible consequences, including outcomes, resource costs, and utility. Decision trees are widely used due to their simplicity, interpretability, and effectiveness in handling diverse data types. The structure of a decision tree includes:

1. *Root Node*: This is the starting point of the tree and represents the entire dataset. A decision or test is made at this node based on a specific feature.

2. *Internal Nodes*: These represent tests or decisions based on features of the data. Each node splits into branches depending on the outcome of the test.

3. *Branches*: These connect nodes and represent the outcomes of the tests at each node.

4. *Leaf Nodes*: These are the endpoints of the tree and represent the final output or decision (e.g., a class label in classification tasks or a value in regression tasks).

A decision tree-based algorithm recursively splits the dataset based on the feature that provides the best separation or minimizes the error. The splitting criterion is typically based on metrics like *Gini Impurity*, *Information Gain*, or *Mean Squared Error* (for regression).

2.2.2 F$_\beta$-score Metric

In this study, none of these metrics are used. Instead, the algorithm relies on a specific score: F_β. The *F_β-score* is a generalization of the *F_1-score* that introduces a parameter β to weigh recall more heavily than precision (or vice versa). This metric—which ranges between 0 and 1, where 0 indicates that either precision or recall (or both) is 0, and 1 indicates perfect precision and recall (precision = 1 and recall = 1)—is particularly useful when the trade-off between precision and recall is not equally important in a given problem. The F$_\beta$-score emphasizes recall when $\beta > 1$, and precision when $\beta < 1$. When $\beta = 1$, the F$_\beta$-score reduces to the standard F$_1$-score, which equally balances precision and recall.

The formula of the F$_\beta$-score is:

$$F_\beta = (1+\beta^2) \cdot \frac{\text{Precision} \cdot \text{Recall}}{(\beta^2 \cdot \text{Precision}) + \text{Recall}} \tag{2.1}$$

Where:

- *Precision* is the fraction of correctly predicted positive instances among all predicted positive instances:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2.2}$$

- *Recall* is the fraction of correctly predicted positive instances among all actual positive instances:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{2.3}$$

Here, *True Positives* (TP) represent cases correctly classified as positive, *False Positives* (FP) represent cases incorrectly classified as positive, and *False Negatives* (FN) represent cases that are positive but were incorrectly classified as negative.

2.2.3 Node Selection Algorithm

Once the tree has been trained, the core of the algorithm, known as *Best Node Selection*, is executed. The following pseudo-codes illustrate its operation in detail.

Algorithm 1 Best Node Selection

```
**Input:** Dataset $D$, Iterations $N$, Tree Depth $depth$, Parameter $\beta$
**Output:** Best nodes and their decision paths
$X\_copy, y\_copy \leftarrow$ copy of $X, y$          ▷ Data and target of dataset $D$ respectively
$N \leftarrow 3$                                      ▷ Number of iterations
$depth \leftarrow 2$                                  ▷ Tree depth for each iteration
$\beta \leftarrow 1$                                  ▷ Value of $\beta$ parameter

for $k \leftarrow 1$ to $N$ do
    Initialize $T$ as a Decision Tree model
    Train $T$ on $(X\_copy, y\_copy)$ with $max\_depth = depth$
    VISUALIZE_TREE(T)
    Extract tree structure:
        $n\_nodes \leftarrow$ number of nodes in $T$
        $decision\_path \leftarrow$ path of samples in $T$
    Compute best node:
        $best\_node \leftarrow$ CALCULATE_BEST_NODE($y\_copy, T, n\_nodes, decision\_path, \beta$)
    Extract node path:
        $best\_node\_id \leftarrow$ identifier of best node
        $node\_path \leftarrow$ GET_NODE_PATH($T, best\_node\_id, feature\_names$)
    Remove samples associated with $best\_node$:
        $X\_copy, y\_copy \leftarrow$
            REMOVE_NODE_SAMPLES($X\_copy, y\_copy, decision\_path, best\_node\_id$)
    Print best node metrics and decision path
end for
Return best nodes and their decision paths
```

Algorithm 2 Visualize Decision Tree

```
function VISUALIZE_TREE($T$)
    Export decision tree structure:
      $dot\_data \leftarrow$ Export_Graphviz($T$) with parameters:
        feature_names $\leftarrow$ $data.feature\_names$
        class_names $\leftarrow$ $data.target\_names$
        filled, rounded, special_characters, node_ids $\leftarrow$ True
        proportion $\leftarrow$ False
    Generate graphical representation:
      $graph \leftarrow$ Graphviz_Source($dot\_data$)
      Display($graph$)
    Return graphical representation
end function
```

Algorithm 3 Calculate Best Node

```
function CALCULATE_BEST_NODE($y, T, n\_nodes, decision\_path, \beta$)
    $node\_metrics \leftarrow \emptyset$
    for each $node\_id$ in $T$ do
        $samples \leftarrow$ indices of samples passing through $node\_id$
        $num\_samples \leftarrow |samples|$
        $y\_true \leftarrow$ true labels of $samples$
        $y\_pred \leftarrow$ majority class in $node\_id$
        Compute confusion matrix values:
          $TP \leftarrow$ count($y\_true = 1 \wedge y\_pred = 1$)
          $FP \leftarrow$ count($y\_true = 0 \wedge y\_pred = 1$)
          $TN \leftarrow$ count($y\_true = 0 \wedge y\_pred = 0$)
          $FN \leftarrow$ count($y\_true = 1 \wedge y\_pred = 0$)
        Compute evaluation metrics:
          $Precision \leftarrow TP/(TP + FP)$
          $Recall \leftarrow TP/(TP + FN)$
          $F\_score \leftarrow (1 + \beta^2) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$
        Store $(node\_id, num\_samples, impurity, metrics)$ in $node\_metrics$
    end for
    Sort $node\_metrics$ by $F\_score$ (descending), $num\_samples$ (descending)
    Return best node
end function
```

Algorithm 4 Get Node Path

```
function GET_NODE_PATH($T, node\_id, feature\_names$)
    $node\_path \leftarrow \emptyset$
    while $node\_id \neq root$ do
        $parent\_node \leftarrow$ parent of $node\_id$ in $T$
        if $node\_id$ is left child of $parent\_node$ then
            $direction \leftarrow "<="$
        else
            $direction \leftarrow ">"$
        end if
        $feature \leftarrow$ feature used for split at $parent\_node$
        $threshold \leftarrow$ split threshold at $parent\_node$
        Store "feature direction threshold" in $node\_path$
        Move to $parent\_node$
    end while
    Return reversed($node\_path$)
end function
```

Algorithm 5 Remove Node Samples

```
function REMOVE_NODE_SAMPLES($X, y, decision\_path, node\_id$)
    $samples \leftarrow$ indices of samples passing through $node\_id$
    $X\_new \leftarrow$ dataset obtained by removing $samples$ from $X$
    $y\_new \leftarrow$ dataset obtained by removing $samples$ from $y$
    Return $(X\_new, y\_new)$
end function
```

Algorithm 1 provides the implementation of Best Node Selection, which calls several dedicated functions, each explained in Algorithms 2, 3, 4, and 5.

In essence, this procedure selects the best node in the decision tree based on the F_β metric: the higher the value (closer to 1), the better the node. If multiple nodes achieve the same score, the one with the highest number of samples is preferred. Best Node Selection will be repeated several times to obtain, step-by-step, the best nodes created by the decision trees that will be trained from time to time, at each iteration. Specifically, at each iteration, the decision tree is trained and visualized, the best node is selected, its split features (i.e., the decision path) are extracted, and its samples are removed from the dataset. This ensures that the next decision tree is trained on a dataset that excludes samples already associated with previously selected nodes.

2.3 Software and Tools Used

The implementation of the algorithm and the associated analysis were carried out using Python, a versatile and widely used programming language for data science, machine learning, and statistical analysis. Specifically, the code was written and executed in *Visual Studio Code* with the *Jupyter Notebook* extension, which allows for an interactive coding environment within the IDE. This setup provides a seamless experience for running and editing Jupyter notebooks while benefiting from the features and tools available in *Visual Studio Code*.

The libraries utilized in the implementation are:

- *Graphviz*: Utilized for visualizing decision trees and generating graphical outputs that support the analysis. *Graphviz* is a powerful library for rendering graphs and tree-like structures, making it ideal for displaying the hierarchical nature of decision trees.
- *Matplotlib*: A widely used library for creating static, animated, and interactive visualizations in Python. It is employed to generate plots that help in understanding data distributions and model performance.
- *NumPy*: Used for numerical operations and handling multi-di-mensional arrays efficiently. It serves as the backbone for numerical computations in Python.
- *Pandas*: Employed for data manipulation and analysis, particularly for handling structured data in the form of dataframes.
- *Scikit-learn*: A machine learning library that provides tools for training decision trees, calculating metrics (e.g., F_β-score), and performing related operations.
- *SciPy.stats*: A module from the SciPy library that provides statistical functions and tests, useful for analyzing and interpreting data distributions.
- *Seaborn*: A statistical data visualization library built on top of Matplotlib, designed to create visually appealing and informative graphs, often used for exploring relationships in datasets.

The Jupyter notebooks were executed within an *Anaconda* environment, which simplifies package management and ensures compatibility among various Python libraries. This setup facilitates reproducibility and minimizes potential issues with library dependencies.

Chapter 3

Data Analysis and Algorithm Evaluation

3.1 Dataset Description

The dataset used for this thesis is *Breast Cancer Wisconsin (Diagnostic)*, one of the toy datasets made available by *scikit-learn*. This dataset contains the cell nuclei characteristics of 569 digitized images obtained by Fine Needle Aspiration (FNA) of breast mass tissue. Each of the images are classified (diagnosed) as being *Benign* or *Malignant*.

The dataset's features encapsulate various aspects of cell nuclei characteristics:

- *radius*: mean distance from the center to points on the perimeter of the nucleus;
- *texture*: standard deviation of gray-scale values in the image of the nucleus;
- *perimeter*: length of the boundary of the nucleus;
- *area*: total area of the nucleus;
- *smoothness*: measure of local variation in radius lengths;
- *compactness*: perimeter^2 / area - 1.0, indicating how compact the nucleus is;
- *concavity*: severity of concave portions of the nucleus boundary;
- *concave points*: number of concave portions of the nucleus boundary;
- *symmetry*: symmetry of the nucleus shape;
- *fractal dimension*: "coastline approximation" - 1, indicating how the nucleus boundary changes at different scales.

Each attribute is computed for the *mean*, *standard error*, and *"worst"* or largest (mean of the three worst/largest values) across all nuclei in an image, resulting in a total of 30 variables.

3.2 Exploratory Data Analysis

In this section, an exploratory data analysis (EDA) is conducted to understand the structure, characteristics, and potential patterns within the data. As a first step, the dataset is checked for missing values and duplicate rows, none of which are found. Then, to obtain an initial overview, the basic descriptive statistics of the 30 variables are computed. Since the dataset, for 10 attributes, reports *mean*, *standard error*, and *worst*, the values of the statistics are presented in three tables for clarity and ease of interpretation.

Table 3.1: Statistics of the *mean* variables

	count	mean	std	min	25%	50%	75%	max
mean radius	569	14.127	3.524	6.981	11.700	13.370	15.780	28.110
mean texture	569	19.290	4.301	9.710	16.170	18.840	21.800	39.280
mean perimeter	569	91.969	24.299	43.790	75.170	86.240	104.100	188.500
mean area	569	654.889	351.914	143.500	420.300	551.100	782.700	2501.000
mean smoothness	569	0.096	0.014	0.053	0.086	0.096	0.105	0.163
mean compactness	569	0.104	0.053	0.019	0.065	0.093	0.130	0.345
mean concavity	569	0.089	0.080	0.000	0.030	0.062	0.131	0.427
mean concave points	569	0.049	0.039	0.000	0.020	0.034	0.074	0.201
mean symmetry	569	0.181	0.027	0.106	0.162	0.179	0.196	0.304
mean fractal dimension	569	0.063	0.007	0.050	0.058	0.062	0.066	0.097

Table 3.2: Statistics of the *error* variables

	count	mean	std	min	25%	50%	75%	max
radius error	569	0.405	0.277	0.112	0.232	0.324	0.479	2.873
texture error	569	1.217	0.552	0.360	0.834	1.108	1.474	4.885
perimeter error	569	2.866	2.022	0.757	1.606	2.287	3.357	21.980
area error	569	40.337	45.491	6.802	17.850	24.530	45.190	542.200
smoothness error	569	0.007	0.003	0.002	0.005	0.006	0.008	0.031
compactness error	569	0.025	0.018	0.002	0.013	0.020	0.032	0.135
concavity error	569	0.032	0.030	0.000	0.015	0.026	0.042	0.396
concave points error	569	0.012	0.006	0.000	0.008	0.011	0.015	0.053
symmetry error	569	0.021	0.008	0.008	0.015	0.019	0.023	0.079
fractal dimension error	569	0.004	0.003	0.001	0.002	0.003	0.005	0.030

Table 3.3: Statistics of the *worst* variables

	count	mean	std	min	25%	50%	75%	max
worst radius	569	16.269	4.833	7.930	13.010	14.970	18.790	36.040
worst texture	569	25.677	6.146	12.020	21.080	25.410	29.720	49.540
worst perimeter	569	107.261	33.603	50.410	84.110	97.660	125.400	251.200
worst area	569	880.583	569.357	185.200	515.300	686.500	1084.000	4254.000
worst smoothness	569	0.132	0.023	0.071	0.117	0.131	0.146	0.223
worst compactness	569	0.254	0.157	0.027	0.147	0.212	0.339	1.058
worst concavity	569	0.272	0.209	0.000	0.115	0.227	0.383	1.252
worst concave points	569	0.115	0.066	0.000	0.065	0.100	0.161	0.291
worst symmetry	569	0.290	0.062	0.157	0.250	0.282	0.318	0.664
worst fractal dimension	569	0.084	0.018	0.055	0.071	0.080	0.092	0.207

A consistent pattern emerges across all three tables. Specifically, the *area* attribute shows the highest values, followed by progressively smaller values for the *perimeter*, *texture*, and *radius* attributes. The remaining attributes exhibit generally small and similar values.

Subsequently, box plots are drawn. The box plot is a valuable tool when it comes to understanding, at a glance, the distribution of a variable and the presence of any outliers. As shown in Figure 3.1, a general trend of right-skewed distributions is observed across most variables. Notably, all outliers are located in the right tail, except for three variables—`mean smoothness`, `mean symmetry`, and `worst smoothness` —which also reveal outliers in the left tail. The predominance of right-tailed outliers is quite logical: it is more likely to encounter individuals with exceptionally large tumor sizes than those with exceptionally small sizes, as the latter are much more difficult to diagnose. Additionally, the box size, representing the interquartile range—i.e., the difference between the third and first quartiles of the variable's distribution—is relatively narrow for all the *error* variables, indicating limited variability within the central portion of their distributions. These variables also display extreme outliers that deviate substantially from the central values, suggesting the presence of rare observations with different characteristics from the majority of the dataset.

Further insights are provided by Figure 3.2, which compares the histograms of each variable with the theoretical normal distribution curve. This comparison aids in evaluating how well each variable conforms to the normality assumption and offers a more detailed view of their individual distributions. The plots suggest that most variables follow a normal distribution to some extent. In particular, the variables associated to *texture*, *smoothness*, and *symmetry* show the closest adherence to normality, whereas those related to *perimeter*, *area*, and *concavity* deviate more significantly.

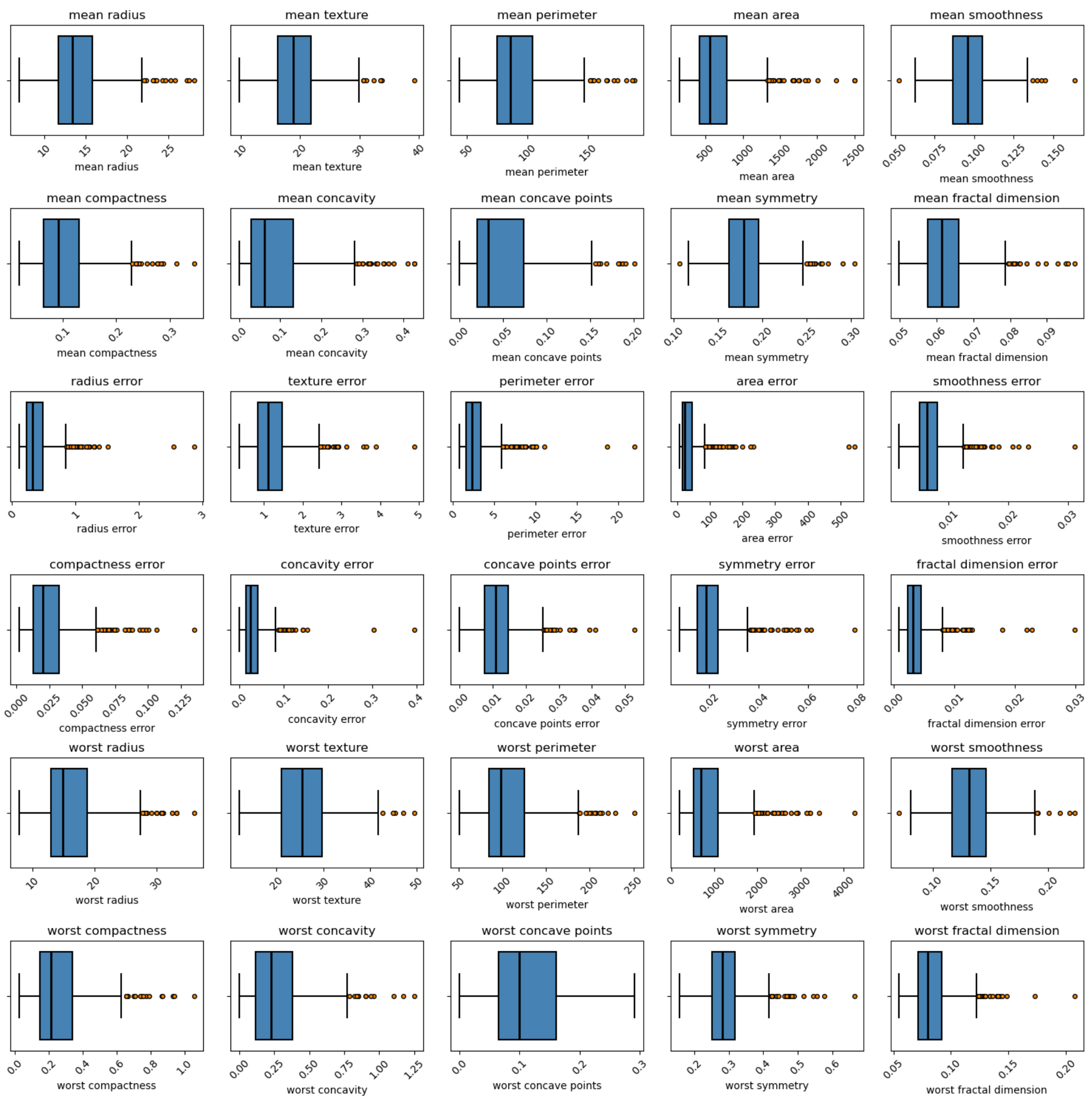

Figure 3.1: Box plots of all variables

Correlation analysis is another effective tool for exploring relationships between variables. A heatmap, shown in Figure 3.3, visualizes the correlations among all variables. Strong correlations—both positive and negative—are represented by brighter colors, while weaker correlations are indicated by softer shades. As seen in Figure 3.3, the strongest correlations occur along the diagonal, where each variable correlates with itself. Generally, positive correlations (depicted in red) are more prevalent than negative correlations (shown in blue), which are relatively weak and light in color. Among the positive correlations, five main groups exhibit nearly maximum correlation, all of which are associated with attributes referring to the size of the tumor mass. These groups primarily involve combinations of the *mean* and *worst* variables, with an additional group representing the *error* variables. On the negative cor-

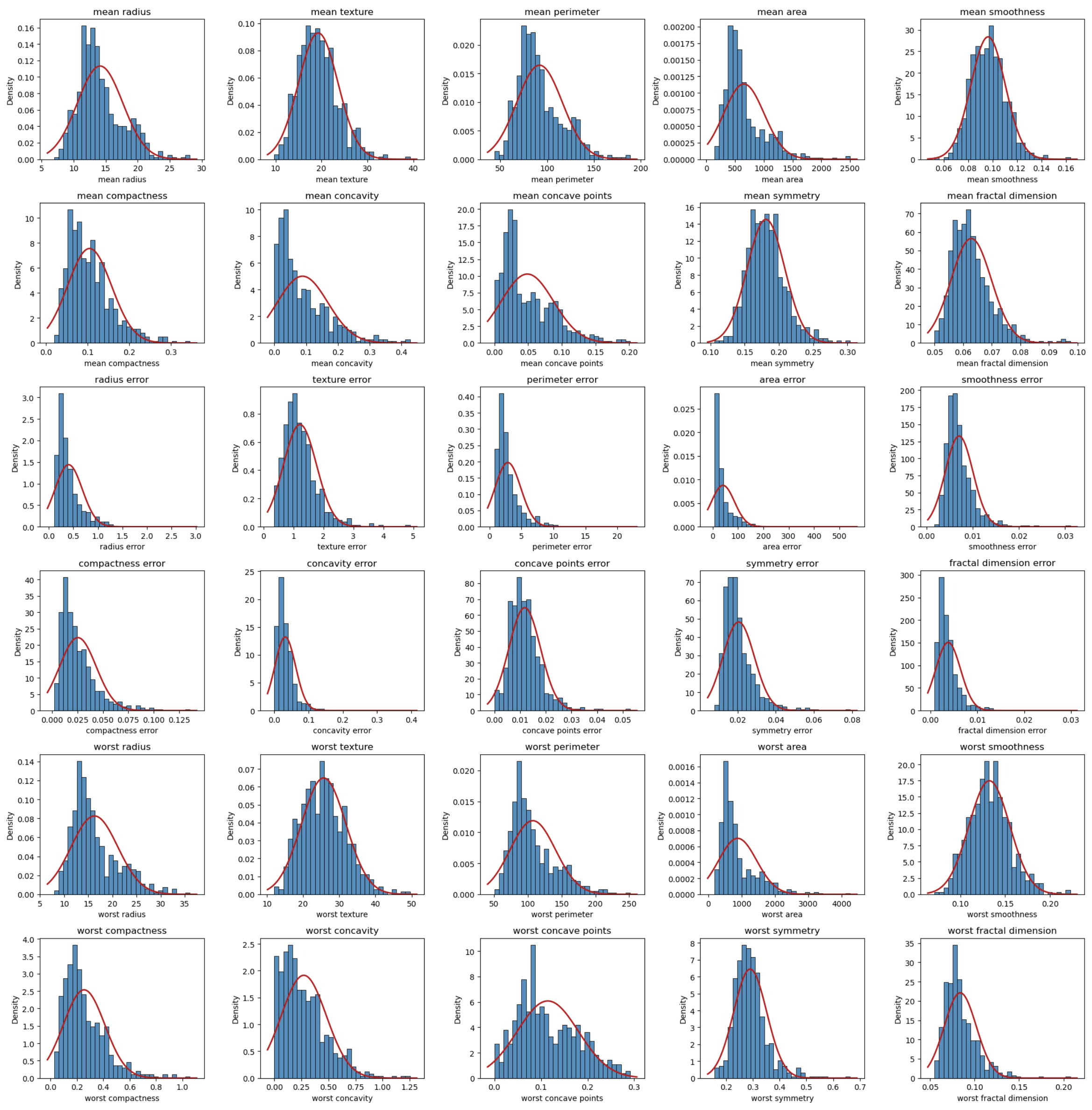

Figure 3.2: Histograms vs theoretical normal curve of all variables

relation side, the strongest negative correlations involve `mean fractal dimension` and `smoothness error`, while the weakest correlations are associated with `symmetry error`, `worst symmetry`, and `worst fractal dimension`.

Given that this study focuses on supervised clustering, it is essential to examine the initial distribution of class labels assigned to each observation. A bar chart, presented in Figure 3.4, illustrates the number of observations classified as Benign and Malignant. The chart clearly shows that the number of Benign observations exceeds that of Malignant observations (by 145 units).

Lastly, it is interesting to examine the pairwise relationships between the first ten variables—the *mean* variables—grouped by class label. For this purpose, a pair plot is employed to highlight any potential patterns or clusters between pairs of vari-

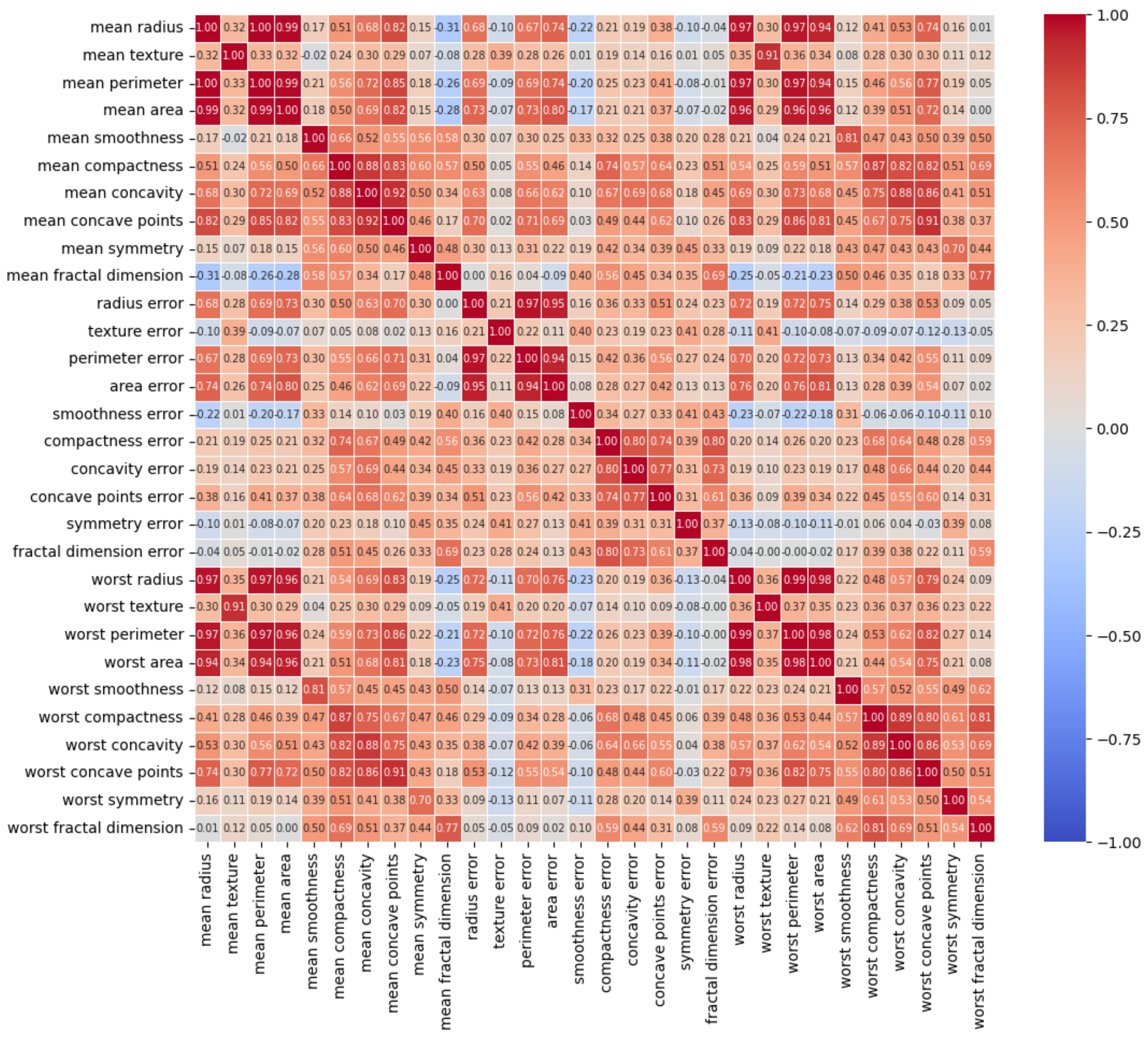

Figure 3.3: Heatmap of correlations

ables. The overall trend observed in Figure 3.5 reveals a clear distinction between the two classes of observations, emphasizing the presence of distinct groups in the data. Particularly, observations belonging to the Benign class showcase lower and less variable values compared to those of the Malignant class. This finding aligns with clinical and statistical expectations: benign tumors tend to have smaller sizes and more uniform structures, whereas malignant tumors are generally larger and manifest greater irregularities. Although most scatter plots display a dispersed pattern, notable relationships can be identified. A strong linear relationship is observed between `mean radius` and `mean perimeter`, while a quadratic relationship is quite evident between `mean radius` and `mean area`, as well as between `mean perimeter` and `mean area`. Furthermore, other plots suggest less pronounced but still discernible linear relationships, such as those between `mean concavity` and `mean compactness`, and between `mean concavity` and `mean concave points`.

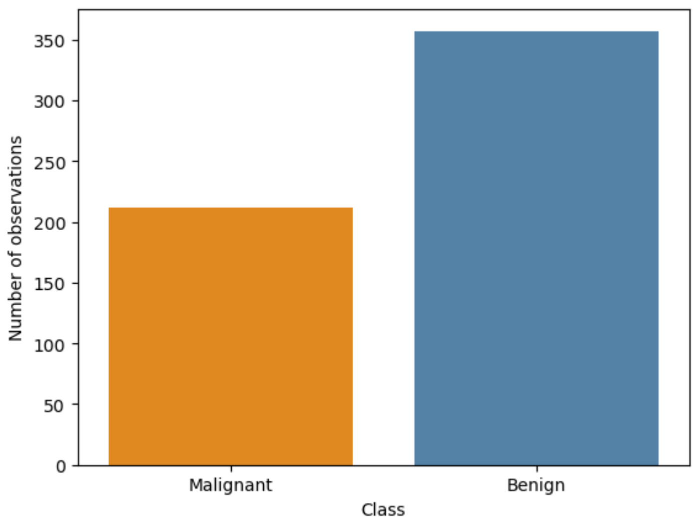

Figure 3.4: Bar chart of Benign vs Malignant observations

3.3 Application and Results

This section presents the direct application of the implemented algorithm to the Breast Cancer Wisconsin (Diagnostic) dataset, which includes a description of the obtained results. In this experiment, decision trees of depth two were selected, performing three iterations of Best Node Selection process. Additionally, the parameter $\beta = 1$ was chosen to equally balance Precision and Recall.

Below are displayed, for each iteration, the generated decision tree along with three tables: one containing the metrics of each node in the tree, another displaying the metrics of the best node for that iteration, and the last outlining the decision path of the best node.

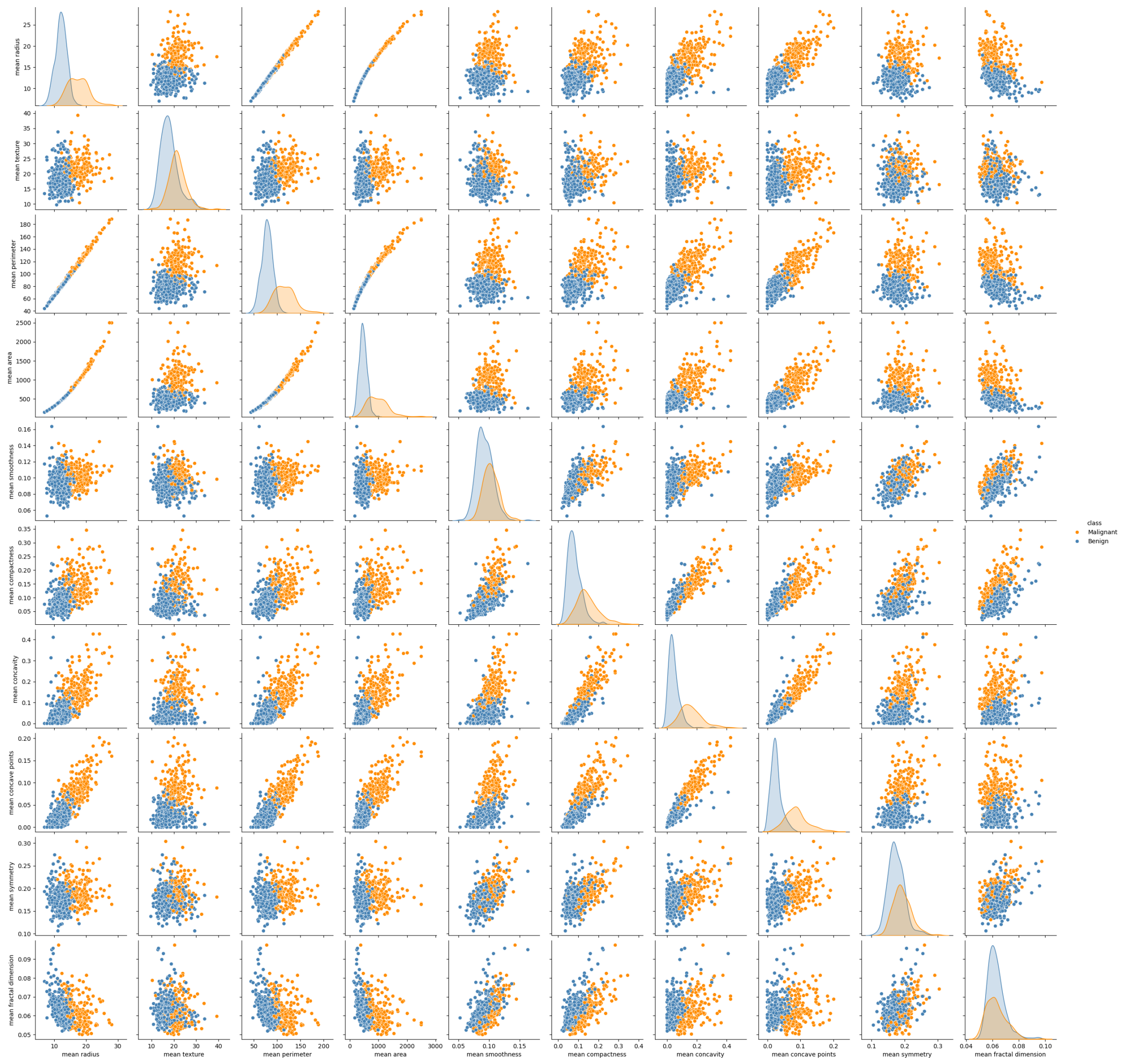

Figure 3.5: Pair plot of the *mean* variables

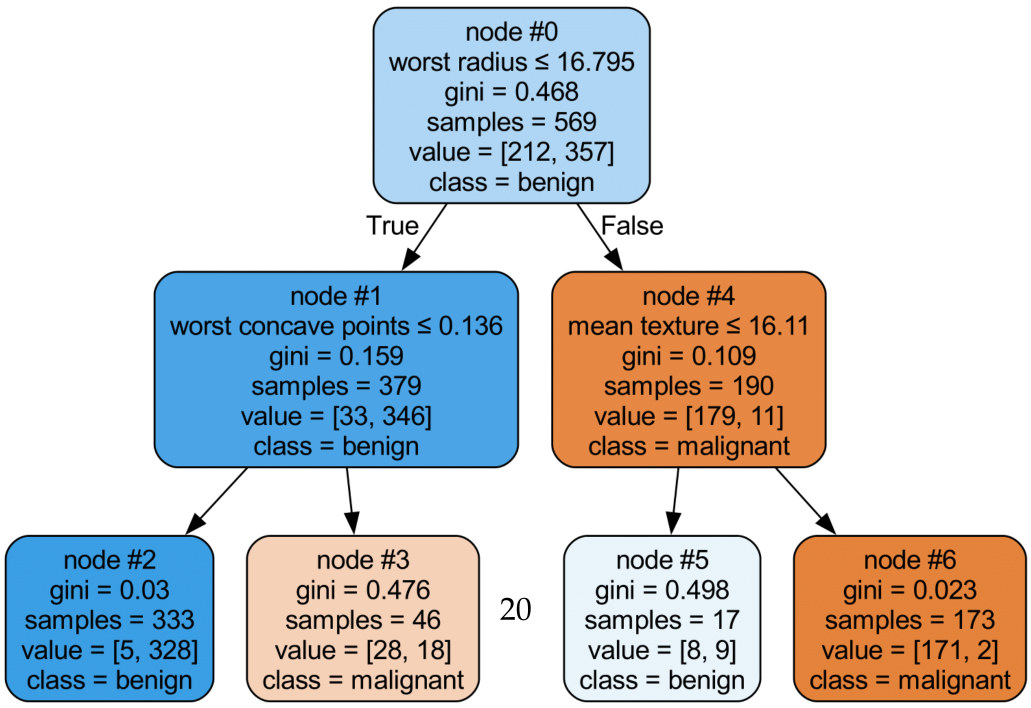

Table 3.4: Metrics for each node of the first iteration

Node ID	Samples	Impurity	TP	FP	TN	FN	Precision	Recall	F-score
0	569	0.468	357	212	0	0	0.627	1	0.771
1	379	0.159	346	33	0	0	0.913	1	0.954
2	333	0.030	328	5	0	0	0.985	1	0.992
3	46	0.476	0	0	28	18	0	0	0
4	190	0.109	0	0	179	11	0	0	0
5	17	0.498	9	8	0	0	0.529	1	0.692
6	173	0.023	0	0	171	2	0	0	0

Table 3.5: Metrics for the best node of the first iteration

Node ID	Samples	Impurity	TP	FP	TN	FN	Precision	Recall	F-score
2	333	0.030	328	5	0	0	0.985	1	0.992

Table 3.6: Decision path of the best node of the first iteration

Variable	Direction	Value
worst radius	$\leq$	16.795
worst concave points	$\leq$	0.136

Figure 3.6 illustrates the first decision tree constructed during the initial iteration of the algorithm. As shown in Table 3.4, the nodes reveal notable differences in terms of sample size: some nodes contain a high number of samples, while others have significantly fewer. Interestingly, nodes with fewer samples tend to have a high impurity index, whereas impurity remains generally low in nodes with a larger sample size. Regarding TP, FP, TN, and FN values, the first three nodes display high TP values, although the root node also presents a high FP count. The TN values are high only for node 4 and node 6, while FN values remain generally low. Notably, only four out of the seven nodes have nonzero Precision and Recall values, resulting in a nonzero F-score. Among them, node 2 stands out as the best-performing node, achieving an F-score of approximately 0.992. This node contains 333 samples and has a Gini impurity index of about 0.03. Table 3.5 provides a detailed view of its characteristics, while its decision path, determined by the variables `worst radius` and `worst concave points` with threshold values of 16.795 and 0.136, is shown in Table 3.6.

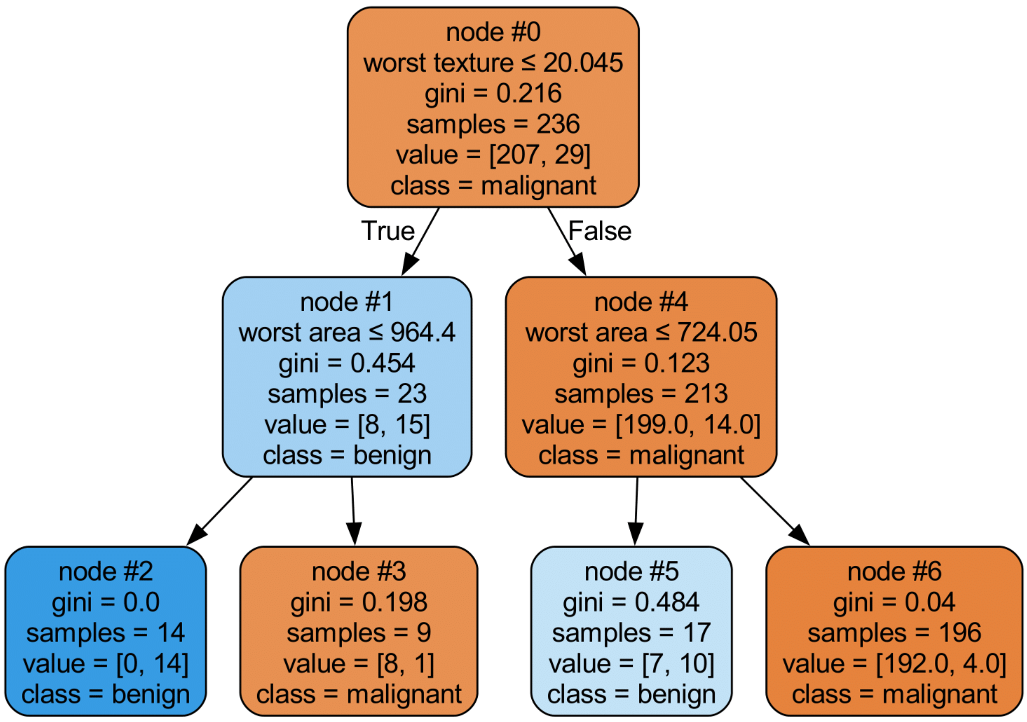

Figure 3.7: Decision tree of the second iteration

Table 3.7: Metrics for each node of the second iteration

Node ID	Samples	Impurity	TP	FP	TN	FN	Precision	Recall	F-score
0	236	0.216	0	0	207	29	0	0	0
1	23	0.454	15	8	0	0	0.652	1	0.789
2	14	0	14	0	0	0	1	1	1
3	9	0.198	0	0	8	1	0	0	0
4	213	0.123	0	0	199	14	0	0	0
5	17	0.484	10	7	0	0	0.588	1	0.741
6	196	0.040	0	0	192	4	0	0	0

Table 3.8: Metrics for the best node of the second iteration

Node ID	Samples	Impurity	TP	FP	TN	FN	Precision	Recall	F-score
2	14	0	14	0	0	0	1	1	1

Table 3.9: Decision path of the best node of the second iteration

Variable	Direction	Value
worst texture	$\leq$	20.045
worst area	$\leq$	964.400

The decision tree constructed during the second iteration is depicted in Figure 3.7. As in the first iteration, Table 3.7 highlights substantial differences in sample size among the nodes. However, since the samples corresponding to the best node from the first iteration (333 samples) have been removed, the nodes in the current tree generally have a lower sample size. The impurity index remains higher for nodes with fewer samples, with the exception of node 2, and lower for the others. In this iteration, TP, FP, and FN values are generally low, while TN exhibits consistently high values. The F-score is nonzero in only three cases, reaching its maximum at node 2, which is identified as the best-performing node. As shown in Table 3.8, this node achieves the highest F-score, Precision, and Recall values (i.e., 1), while also maintaining a zero impurity index. However, this comes at the cost of a limited number of samples. The decision path leading to node 2, detailed in Table 3.9, is determined by the variables `worst texture` and `worst area`, with threshold values of 20.045 and 964.4, respectively.

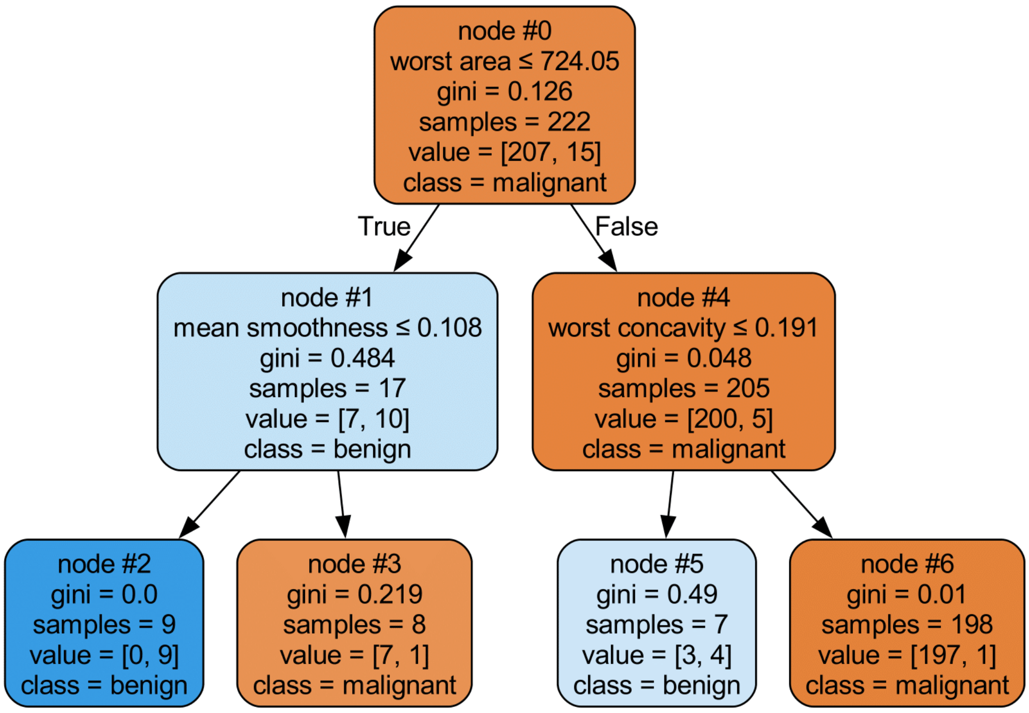

Figure 3.8: Decision tree of the third iteration

Table 3.10: Metrics for each node of the third iteration

Node ID	Samples	Impurity	TP	FP	TN	FN	Precision	Recall	F-score
0	222	0.126	0	0	207	15	0	0	0
1	17	0.484	10	7	0	0	0.588	1	0.741
2	9	0	9	0	0	0	1	1	1
3	8	0.219	0	0	7	1	0	0	0
4	205	0.048	0	0	200	5	0	0	0
5	7	0.490	4	3	0	0	0.571	1	0.727
6	198	0.010	0	0	197	1	0	0	0

Table 3.11: Metrics for the best node of the third iteration

Node ID	Samples	Impurity	TP	FP	TN	FN	Precision	Recall	F-score
2	9	0	9	0	0	0	1	1	1

Table 3.12: Decision path of the best node of the third iteration

Variable	Direction	Value
worst area	$\leq$	724.050
mean smoothness	$\leq$	0.108

Finally, the last decision tree is illustrated in Figure 3.8. As shown in Table 3.10, the conclusions drawn from this iteration are largely analogous to those of the previous one. This similarity arises from the fact that only 14 samples were removed in the second iteration, resulting in a node composition and impurity index values that closely resemble those observed in the previous step. The same pattern is reflected in all the metrics used to compute the F-score. Once again, as reported in Table 3.11, node 2 is identified as the best-performing node. Specifically, it achieves an F-score, Precision, and Recall of 1, with an impurity index of 0 and a total of 9 samples. The distinguishing variables for this node are `worst area` and `mean smoothness`, with threshold values of 724.05 and 0.108, listed in Table 3.12.

Since the split direction is always to the left, the values of these variables are less than or equal to the given thresholds.

Chapter 4

Discussion

4.1 Interpretation of Results

In the previous chapter, node 2 was consistently selected as the best node. In all three iterations, the chosen nodes exhibit a high F-score and low impurity, indicating the effectiveness of the decision tree's splits. This may stem from the strong distinction between Benign and Malignant classes within the dataset: splitting the data in the first levels of the tree could lead to a clear separation between classes, with a strong predominance of one class in that branch.

A key distinction emerges: only the best node in the first iteration contains a large number of samples. This, combined with its high F-score—though not maximal as in the second and third iterations, where it reaches 1—makes it a representative cluster that effectively captures the underlying data structure.

The non-maximal F-score in the first iteration is due to the presence of observations from more than one class, meaning the node is not entirely pure. In contrast, the best nodes in the second and third iterations contain only samples from a single class, making them fully pure and leading to the highest possible F-score. Based solely on this metric, these nodes would be preferable to the best node in the first iteration. However, their limited number of samples makes them poor representatives of the overall dataset, as they retain overly selective information. Therefore, despite having a slightly lower F-score, the best node from the first iteration is the preferred choice for interpretability.

In addition to the previous considerations, it is observed that among the split variables defining the decision path of the best nodes extracted in the three iterations (i.e., those specific to nodes 0 and 1), five belong to the *worst* category, while only one is *mean*. A similar pattern emerges in node 4, where two out of three split variables are *worst* and one is *mean*. This prevalence suggests that *worst* variables may carry more discriminative information, as they represent the maximum observed value of a given attribute in the tumor mass. Such characteristics could enhance the distinction between Benign and Malignant classes. However, the model's strong reliance on these variables might indicate a heightened sensitivity to extreme values, potentially increasing the risk of overfitting.

Notably, none of the nine split variables belongs to the *error* category, suggesting that these measures either provide less discriminative information or were deemed less useful by the tree selection criterion compared to *worst* and *mean*. This behavior could be attributed to the dataset's structure or to the model's strategy for selecting the most informative attributes.

4.2 Performance in Different Scenarios

As explained at the beginning of Section 3.3, decision trees with a depth of two, a total of three iterations of the Best Node Selection algorithm, and a β parameter set to 1 were selected. These parameters were chosen to provide a concise yet effective demonstration of the algorithm's functioning and performance on a predefined dataset.

That said, an important clarification is needed: the algorithm was tested under different conditions by varying the tree depth, the number of iterations, and the β parameter. The results showed that the algorithm's performance remained largely consistent across all tested scenarios, regardless of the modifications applied. However, the most significant difference was observed in relation to tree depth: as the depth increases, the risk of overfitting becomes more pronounced. This occurs because deeper trees are more likely to generate nodes containing only a few samples from a single class. Since these nodes are maximally pure, their F-score reaches its highest value, leading the algorithm to select them as the best nodes.

Moreover, varying the β parameter, including both very small values (close to zero) and very large values, did not lead to any changes in the structure of the tree or the clusters produced, resulting in identical outcomes. This suggests that β does not play a crucial role in this specific context. When the dataset shows a clear separation between classes, the tree's splitting decisions remain unaffected by β, as the best nodes emerge distinctly without ambiguity.

4.3 Comparison with Existing Literature

The results obtained from the Best Node Selection algorithm highlight key distinctions and similarities with existing supervised clustering methods. Unlike traditional approaches that determine all clusters at once, this algorithm iteratively refines selection, prioritizing interpretability through decision tree structures.

Hu et al. (2024) categorize interpretable clustering methods into pre-clustering, in-clustering, and post-clustering approaches. Best Node Selection aligns with in-clustering techniques but differs in its greedy, stepwise selection, which ensures transparency but may lead to locally optimal clusters rather than globally optimal ones. Similarly, Eick et al. (2004) optimize clustering through fitness-based selection, but their global optimization approach contrasts with the locally-driven decisions of Best Node Selection, reinforcing the purity vs representativeness dilemma observed in both studies.

A different optimization strategy is found in metric-learning-based clustering (Al-Harbi & Rayward-Smith, 2006), where distances are adapted to improve class separation. Best Node Selection, however, does not require explicit metric learning, instead leveraging decision tree partitions for cluster formation. This allows for a more interpretable structure but lacks the flexibility of metric-based adaptations.

Finley and Joachims (2005) integrate SVMs into supervised clustering, optimizing both cluster assignments and classification boundaries. While their approach maximizes class separation, it does not provide explicit decision rules, unlike Best Node Selection, which enhances explainability at the cost of potentially lower generalization.

Application-specific methods, such as supervised gene clustering (Dettling & Bühlmann, 2002) and SurvivalLVQ (de Boer et al., 2024), demonstrate how domain constraints can refine clustering effectiveness. Unlike these methods, Best Node Selection remains purely classi-fication-driven, forming clusters solely based on internal performance metrics rather than external knowledge. This leads to high-purity clusters but raises concerns about overfitting and small-sample bias, similar to challenges seen in semi-supervised clustering (Gao et al., 2024).

Lastly, Ravenda et al. (2024) show that self-supervised clustering benefits from integrating external constraints, which Best Node Selection does not incorporate. However, its structure could be extended to include domain knowledge or probabilistic refinements, potentially improving generalizability.

4.4 Limitations and Implications

While the Best Node Selection algorithm demonstrates strong interpretability and effectiveness in forming high-purity clusters, certain limitations must be acknowledged, as they have important implications for both its practical applications and potential enhancements.

One key limitation is the algorithm's sensitivity to tree depth. As discussed, deeper trees tend to overfit, selecting nodes with few observations that maximize purity at the cost of representativeness. This is particularly relevant when working with datasets characterized by noise or class imbalance, where deeper partitions may emphasize spurious patterns rather than meaningful structures. A direct implication is that, in such cases, tuning depth constraints or incorporating post-hoc pruning mechanisms becomes necessary to ensure that cluster definitions remain robust beyond the training data.

Another limitation concerns the role of the β parameter. Empirical results indicate that, within the tested dataset, variations in β do not affect node selection or clustering outcomes. This suggests that class separability is already well-defined by the tree's natural structure, rendering β redundant in this context. However, the parameter becomes particularly relevant in scenarios with strong class imbalance, where one class is significantly underrepresented. In such cases, setting $\beta = 1$ may lead to clusters dominated by the majority class, increasing the risk of failing to capture meaningful patterns in the minority class. This raises the need to investigate whether alternative formulations or dynamic adjustments of β could enhance its utility in handling imbalanced datasets.

The algorithm's stepwise and greedy selection process also presents an inherent limitation. By prioritizing locally optimal nodes at each step, it may lead to partitions that maximize purity within individual nodes but do not necessarily result in an optimal clustering structure from a global perspective. This suggests that while the method ensures transparency and interpretability, it may benefit from modifications that introduce global optimization mechanisms or ensemble-based adjustments to refine the overall clustering outcome.

Furthermore, the variable selection pattern observed in Section 4.1 suggests a strong reliance on specific feature types, particularly those classified under the *worst* category, with no contribution from *error* measures. While this highlights the discriminative power of certain variables, it also raises concerns regarding potential bias. If the dataset exhibits systematic measurement errors or domain-specific distortions, an excessive dependence on particular variable types could limit the generalizability of the model. Addressing this issue may require feature selection techniques that promote a more balanced contribution of different variable categories.

Finally, the algorithm does not incorporate external domain knowledge or constraints into its decision-making process. Unlike some supervised clustering approaches that exploit additional information to refine partitions, Best Node Selection relies solely on decision tree metrics. This ensures an objective selection process, but it may also restrict its adaptability in fields where domain expertise could provide valuable refinements. Exploring ways to integrate contextual knowledge without compromising interpretability could expand the method's applicability.

Chapter 5

Conclusions and Future Work

This thesis introduced the Best Node Selection algorithm as an interpretable approach to supervised clustering, leveraging decision trees to iteratively refine cluster selection. By prioritizing nodes based on performance metrics, the method offers a structured framework for identifying class-distinct regions within a dataset while maintaining transparency in the selection process. Experimental results confirmed that the algorithm effectively identifies nodes with high F-score and low impurity, demonstrating its ability to exploit decision tree partitions for meaningful clustering.

The analysis provided insights into the algorithm's selection behavior, revealing its strong preference for nodes with well-separated class structures and its reliance on specific feature types. This contributes to high cluster purity but also introduces potential risks of overfitting, particularly when tree depth increases. While this trade-off is inherent to many decision tree-based approaches, refining the method to better balance purity and representativeness remains an important avenue for improvement. Also, the observed insensitivity of the results to variations in the β parameter suggests that future research could explore alternative formulations to enhance its role in datasets with a pronounced class imbalance.

Building on these findings, future work could focus on several directions to enhance the algorithm's flexibility and robustness. One promising avenue involves refining the selection mechanism to balance local purity with global clustering objectives, potentially through optimization strategies that mitigate the effects of greedy selection. Adapting the algorithm to incorporate pruning or depth constraints dynamically could also help prevent overfitting in high-dimensional or noisy datasets. Further investigations into alternative formulations of β could clarify its role in different clustering scenarios and inform refinements that improve adaptability.

Another extension could involve integrating external knowledge into the selection process. Although the current approach relies purely on internal decision tree metrics, hybrid models that incorporate domain-driven constraints or probabilistic refinements could enhance applicability in specialized fields. Additionally, exploring ensemble-based adaptations or semi-supervised clustering techniques may provide further improvements in robustness while preserving interpretability.

In conclusion, the Best Node Selection algorithm provides a structured and interpretable framework for supervised clustering, demonstrating strong performance in controlled scenarios. While its iterative and locally optimal selection process offers advantages in explainability, further refinements could broaden its applicability and address challenges related to overfitting and variable reliance. Future work focused on refining selection criteria, integrating domain-aware mechanisms, and extending the method to alternative datasets and clustering paradigms could enhance its utility in both theoretical research and real-world applications.

References

Al-Harbi, S. H., & Rayward-Smith, V. J. (2006). Adapting k-means for supervised clustering. *Springer*.

de Boer, J., Dedja, K., & Vens, C. (2024). SurvivalLVQ: Interpretable supervised clustering and prediction in survival analysis via learning vector quantization. *Pattern Recognition*.

Dettling, M., & Bühlmann, P. (2002). Supervised clustering of genes. *Genome Biology*.

Eick, C. F., Zeidat, N., & Zhao, Z. (2004). Supervised clustering – algorithms and benefits.

Finley, T., & Joachims, T. (2005). Supervised clustering with support vector machines.

Gao, H., Shen, C., Wang, X., Chan, P.-W., Hon, K.-K., & Li, J. (2024). Interpretable semi-supervised clustering enables universal detection and intensity assessment of diverse aviation hazardous winds. *Nature*.

Hastie, T., Tibshirani, R., & Friedman, J. (2008). *The elements of statistical learning*. Springer.

Hu, L., Jiang, M., Dong, J., Liu, X., & He, Z. (2024). Interpretable clustering: A survey. *arXiv*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning*. Springer.

Kokash, N., & Makhnist, L. (2024). Using decision trees for interpretable supervised clustering. *SN Computer Science*.

Ravenda, F., Ali Bahrainian, S., Raballo, A., Mira, A., & Crestani, F. (2024). A self-supervised seed-driven approach to topicmodelling and clustering. *Journal of Intelligent Information Systems*.

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge University Press.

Further Sources

7.1. toy datasets [Scikit-learn]. (n.d.). https://scikit-learn.org/stable/datasets/toy_dataset.html

Algorithms [Overleaf]. (n.d.). https://www.overleaf.com/learn/latex/Algorithms

DecisionTreeClassifier [Scikit-learn]. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Fbeta_score [Scikit-learn]. (n.d.). https://scikit-learn.org/stable/modules/generated/sklearn.metrics.fbeta_score.html

Tortora, S. (2022). *Basics of machine learning in python with scikit learn*.

Understanding the decision tree structure [Scikit-learn]. (n.d.). https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html