CS420 – ARTIFICIAL INTELLIGENCE

Project 2 – Decision Tree

Report

VNUHCM-UNIVERSITY OF SCIENCE

December 14, 2024

Student Information

Full Name	**Student ID**
Vòng Vĩnh Toàn	22125108
Nguyễn Hoàng Phúc	22125077
Huỳnh Hà Phương Linh	22125049
Huỳnh Đăng Khoa	22125038

Contents

Introduction

The Decision Tree algorithm is a supervised learning algorithm that can be used for both classification and regression tasks. The algorithm works by recursively splitting the dataset into subsets based on the most discriminative features. The splitting process continues until the tree reaches a maximum depth or the number of samples in a node falls below a certain threshold. The Decision Tree algorithm is simple to understand and interpret, making it an excellent choice for tasks that require transparency and interpretability.

In this project, we implement the Decision Tree algorithm using Python and the `scikit-learn` library. We evaluate the algorithm's performance on three different datasets from the UCI Machine Learning Repository. The datasets are as follows:

1. The UCI Breast Cancer Wisconsin dataset [1]
2. The UCI Wine Quality dataset [2]
3. The UCI User Knowledge Modeling dataset [3]

Note that **all figures and tables** in this report are in the **vector image format** for better readability and quality as the reader zooms in since the illustrations can be too complex to be displayed.

Dataset 1: The UCI Breast Cancer Wisconsin

In this Section, we will present the classification results of the first dataset that we used in our project, which is the UCI Breast Cancer Wisconsin dataset [1].

The dataset contains features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. These features describe characteristics of the cell nuclei present in the image. A sample of such images can be found at `http://www.cs.wisc.edu/~street/images/`.

Dataset Characteristics

- **Type:** Multivariate
- **Subject Area:** Health and Medicine
- **Associated Task:** Classification
- **Feature Type:** Real-valued
- **Number of Instances:** 569
- **Number of Features:** 30
- **Missing Values:** No

Dataset Insights

- **Source:** Digitized images of fine needle aspirates (FNA) of breast masses.
- **Objective:** Classify tumors as malignant (M) or benign (B).
- **Methodology:** Features were computed using image analysis techniques to describe characteristics of cell nuclei.
- **Additional Details:**
  - Separating planes were determined using the Multisurface Method-Tree (MSM-T).
  - Feature selection involved exhaustive searches in feature and plane spaces.

Feature Description

- **Input Variables:** Ten real-valued features computed for each cell nucleus:
  1. **Radius:** Mean distance from the center to the perimeter.
  2. **Texture:** Standard deviation of gray-scale values.
  3. **Perimeter:** Measurement of the contour length.
  4. **Area:** Size of the cell nucleus.
  5. **Smoothness:** Local variation in radius lengths.
  6. **Compactness:** Computed as $(\text{perimeter}^2/\text{area}) - 1.0$.
  7. **Concavity:** Severity of concave portions of the contour.
  8. **Concave Points:** Number of concave portions of the contour.
  9. **Symmetry:** Symmetry of the cell nucleus.
  10. **Fractal Dimension:** Approximation of the contour's fractal nature.
- **Output Variable:**
  - Diagnosis: Binary classification where M = malignant and B = benign.

1.1 Data preparation

1.1.1 Import the dataset

The UC Irvine Machine Learning Repository allows for direct dataset import in Python using the `ucimlrepo` package. The package is included in the `requirements.txt` file. The following code demonstrates how to import the Breast Cancer Wisconsin (id = 17) dataset using the `fetch_ucirepo` function:

```python
from ucimlrepo import fetch_ucirepo

breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=17)

X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets
```

1.1.2 Splitting the dataset

The dataset is split into training and testing sets using the `train_test_split` function from the `sklearn` library. The dataset is splitted into 4 sets of different test sizes: 60%, 40%, 20%, and 10% of the dataset in stratified fashion, using the random state of 22125 to ensure reproducibility.

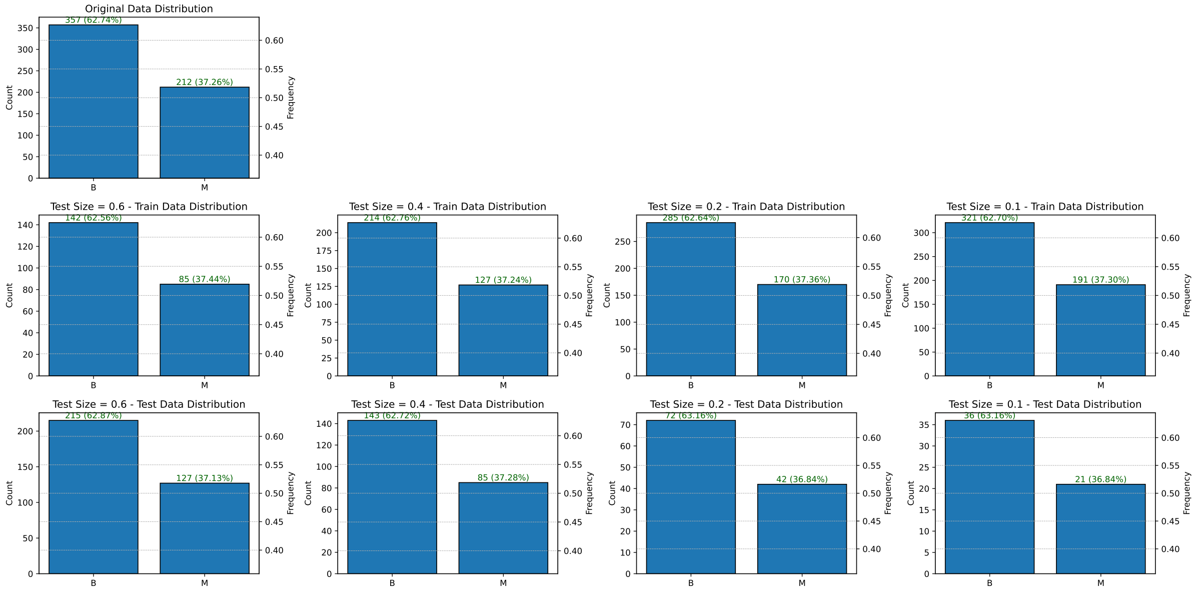
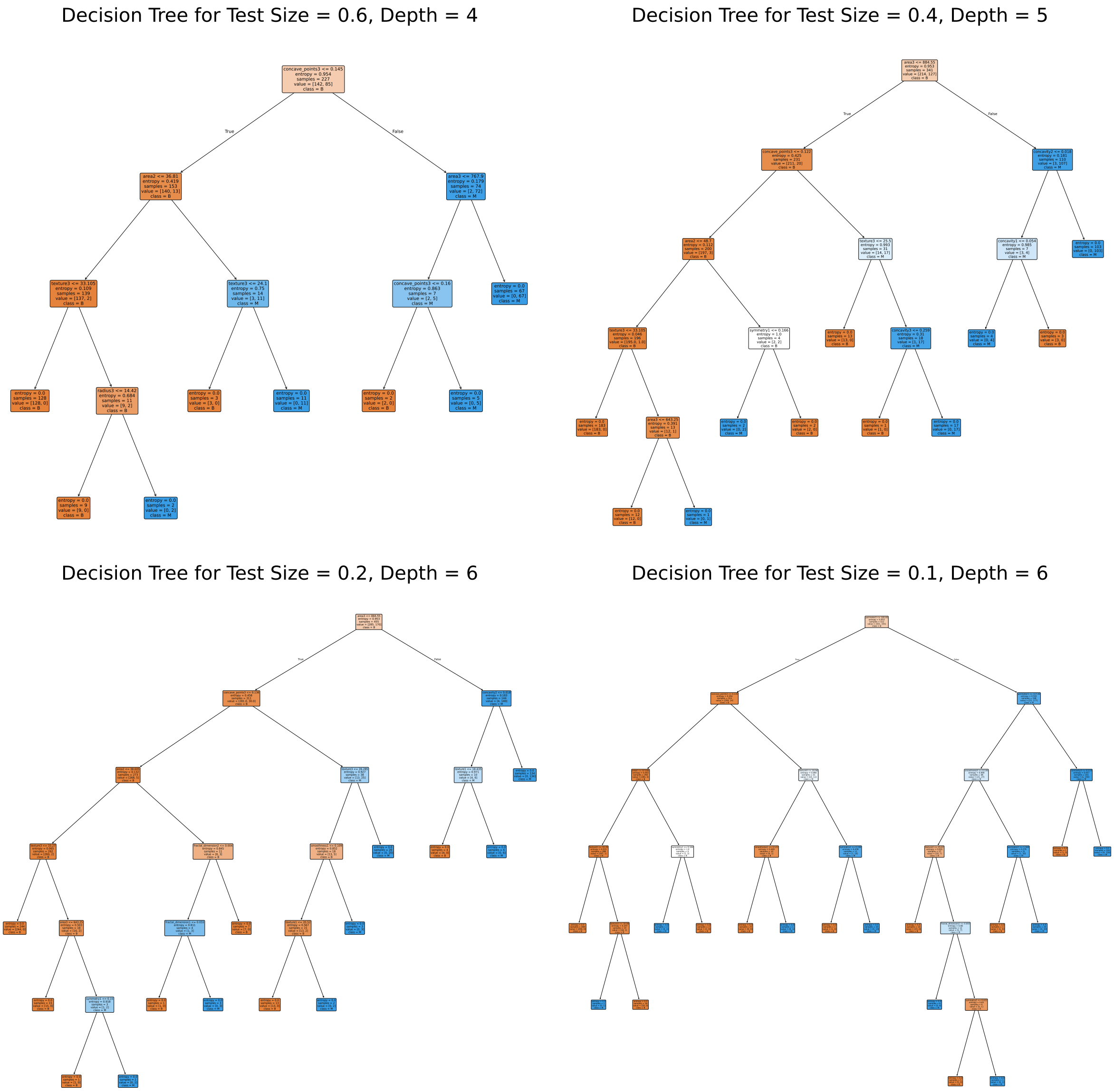


Figure 1: Breast Cancer Wisconsin dataset split

As shown in figure 1, the distribution of the target class is preserved in all splits.

1.2 Decision Tree classifier implementation

Decision Tree classifiers are implemented using the `DecisionTreeClassifier` class from the `sklearn` library. To ensure reproducibility, the random state is set to 22125. We use the Entropy (information gain) criterion to split the nodes and set the maximum depth of the tree to None to allow the tree to grow until all leaves are pure.



Decision Tree for Test Size = 0.6, Depth = 4

Decision Tree for Test Size = 0.4, Depth = 5

Decision Tree for Test Size = 0.2, Depth = 6

Decision Tree for Test Size = 0.1, Depth = 6

concave_points3 <= 0.145
entropy = 0.954
samples = 227
value = [142, 85]
class = B
True
False
area2 <= 36.81
entropy = 0.419
samples = 153
value = [140, 13]
class = B
area3 <= 767.9
entropy = 0.179
samples = 74
value = [2, 72]
class = M
texture3 <= 33.105
entropy = 0.109
samples = 139
value = [137, 2]
class = B
texture3 <= 24.1
entropy = 0.75
samples = 14
value = [3, 11]
class = M
concave_points3 <= 0.16
entropy = 0.863
samples = 7
value = [2, 5]
class = M
entropy = 0.0
samples = 67
value = [0, 67]
class = M
entropy = 0.0
samples = 128
value = [128, 0]
class = B
radius3 <= 14.42
entropy = 0.684
samples = 11
value = [9, 2]
class = B
entropy = 0.0
samples = 3
value = [3, 0]
class = B
entropy = 0.0
samples = 11
value = [0, 11]
class = M
entropy = 0.0
samples = 2
value = [2, 0]
class = B
entropy = 0.0
samples = 5
value = [0, 5]
class = M
entropy = 0.0
samples = 9
value = [9, 0]
class = B
entropy = 0.0
samples = 2
value = [0, 2]
class = M
area3 <= 884.55
entropy = 0.953
samples = 341
value = [214, 127]
class = B
True
False
concave_points3 <= 0.122
entropy = 0.425
samples = 231
value = [211, 20]
class = B
concavity2 <= 0.018
entropy = 0.181
samples = 110
value = [3, 107]
class = M
area2 <= 48.7
entropy = 0.112
samples = 200
value = [197, 3]
class = B
texture3 <= 25.5
entropy = 0.993
samples = 31
value = [14, 17]
class = M
concavity1 <= 0.054
entropy = 0.985
samples = 7
value = [3, 4]
class = M
entropy = 0.0
samples = 103
value = [0, 103]
class = M
texture3 <= 33.105
entropy = 0.046
samples = 196
value = [195.0, 1.0]
class = B
symmetry1 <= 0.166
entropy = 1.0
samples = 4
value = [2, 2]
class = B
entropy = 0.0
samples = 13
value = [13, 0]
class = B
concavity3 <= 0.259
entropy = 0.31
samples = 18
value = [1, 17]
class = M
entropy = 0.0
samples = 4
value = [0, 4]
class = M
entropy = 0.0
samples = 3
value = [3, 0]
class = B
entropy = 0.0
samples = 183
value = [183, 0]
class = B
area3 <= 643.25
entropy = 0.391
samples = 13
value = [12, 1]
class = B
entropy = 0.0
samples = 2
value = [0, 2]
class = M
entropy = 0.0
samples = 2
value = [2, 0]
class = B
entropy = 0.0
samples = 1
value = [1, 0]
class = B
entropy = 0.0
samples = 17
value = [0, 17]
class = M
entropy = 0.0
samples = 12
value = [12, 0]
class = B
entropy = 0.0
samples = 1
value = [0, 1]
class = M
area3 <= 884.55
entropy = 0.953
samples = 455
value = [285, 170]
class = B

Figure 2: Breast Cancer Wisconsin dataset Decision Trees with different test sizes

The Decision Trees are visualized in figure 2. The trees grow deeper as the test size decreases, indicating that the model is more complex and overfits the training data when the test size is small.

The first node of the Decision Tree splits the dataset based on different features:

- Decision tree of test size 60% chooses the feature `concave_points3` to split the dataset.
- Decision tree of test size 40% and 20% chooses the feature `area3` to split the dataset.
- Decision tree of test size 10% chooses the feature `perimeter3` to split the dataset.

Despite the differences in the splitting features, the first node's entropy is approximately 0.953 in all Decision Trees, indicating that the first split is not very informative. However, the Decision Tree classifier effectively splits the dataset into pure leaves with high information gain.

1.3 Performance evaluation

1.3.1 Classification report

The classification report provides a comprehensive evaluation of the Decision Tree classifier's performance using the `classification_report` function from the `sklearn.metrics` module. The report includes the following metrics:

- **Precision:** The ratio of true positive samples to the sum of true positive and false positive samples.
- **Recall:** The ratio of true positive samples to the sum of true positive and false negative samples.
- **F1-score:** The harmonic mean of precision and recall.
- **Support:** The number of samples in each class.

Classification Report for Test Size = 0.6

	Precision	Recall	F1-score	Support
B	0.94	0.94	0.94	215
M	0.90	0.89	0.89	127
Accuracy			0.92	342
Macro avg	0.92	0.91	0.92	342
Weighted avg	0.92	0.92	0.92	342

Classification Report for Test Size = 0.4

	Precision	Recall	F1-score	Support
B	0.94	0.92	0.93	143
M	0.87	0.89	0.88	85
Accuracy			0.91	228
Macro avg	0.91	0.91	0.90	228
Weighted avg	0.91	0.91	0.91	228

Classification Report for Test Size = 0.2

	Precision	Recall	F1-score	Support
B	0.95	0.88	0.91	72
M	0.81	0.93	0.87	42
Accuracy			0.89	114
Macro avg	0.89	0.90	0.88	114
Weighted avg	0.90	0.89	0.90	114

Classification Report for Test Size = 0.1

	Precision	Recall	F1-score	Support
B	0.97	0.92	0.94	36
M	0.87	0.95	0.91	21
Accuracy			0.93	57
Macro avg	0.93	0.93	0.92	57
Weighted avg	0.93	0.93	0.93	57

The classification reports show that the Decision Tree classifier performs well on the Breast Cancer Wisconsin dataset, achieving an accuracy of 92% for the test size of 60%, 91% for the test size of 40%, 89% for the test size of 20%, and 93% for the test size of 10%. The classifier has high precision, recall, and F1-score for both classes, indicating that it effectively distinguishes between malignant and benign samples.

1.3.2 Confusion matrix

The confusion matrix provides a visual representation of the Decision Tree classifier's performance using the `confusion_matrix` function from the `sklearn.metrics` module. The matrix shows the number of true positive, false positive, true negative, and false negative samples for each class.

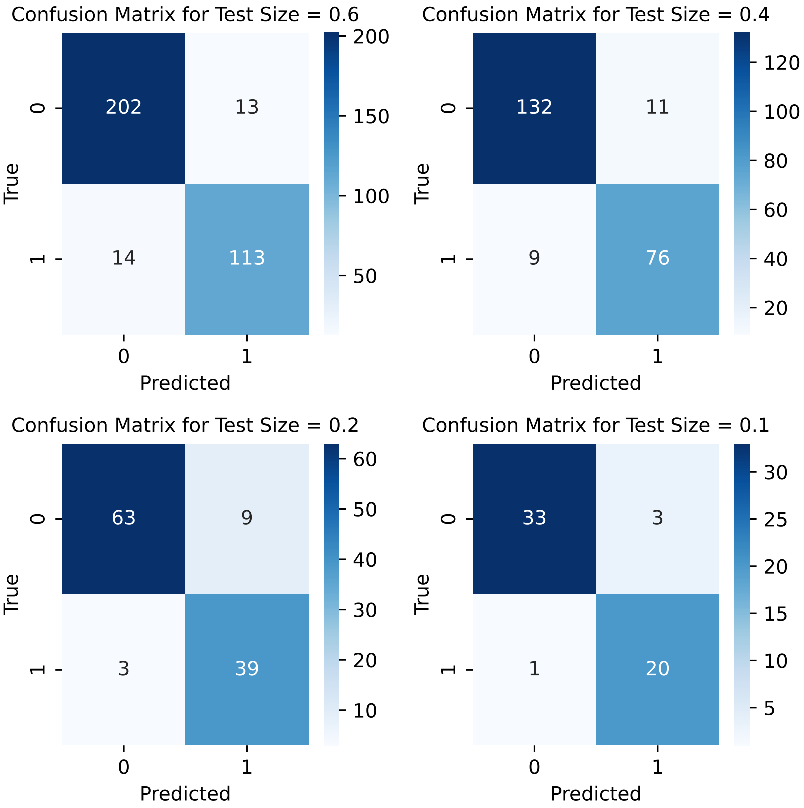


Figure 3: Breast Cancer Wisconsin dataset Confusion Matrices with different test sizes

The confusion matrices in figure 3 show that the Decision Tree classifier correctly classifies most samples in the Breast Cancer Wisconsin dataset. The number of false positive and false negative samples is low, indicating that the classifier has high precision and recall for both classes.

1.4 Depth vs Accuracy evaluation

The Decision Tree classifier's performance is evaluated by varying the maximum depth of the tree. The accuracy of the classifier is computed for different maximum depths using the `accuracy_score` function from the `sklearn.metrics` module for the Decision Tree classifier with a test size of 20%.

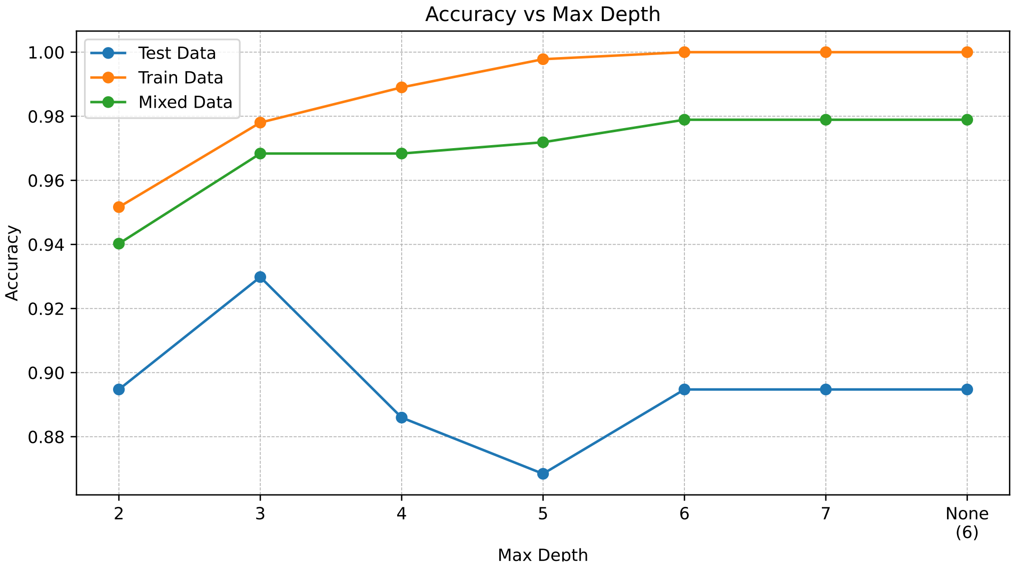


Figure 4: Breast Cancer Wisconsin dataset Depth vs Accuracy

The plot in figure 4 shows that the Decision Tree classifier achieves the highest accuracy on the test data when the maximum depth of the tree is 3. The accuracy decreases as the maximum depth increases, indicating that the model overfits the training data when the tree is too deep.

Dataset 2: The Wine Quality

In this Section, we will present the classification results of the first dataset that we used in our project, which is the UCI Wine Quality dataset [2].

The Wine Quality dataset contains data related to red and white variants of the Portuguese ”Vinho Verde” wine. The dataset includes physicochemical input variables (e.g., acidity, sugar content, alcohol) and an output variable representing the wine quality. Detailed information about the wine can be found at `http://www.vinhoverde.pt/en/` or in the reference by Cortez et al. (2009).

Dataset Characteristics

- **Type:** Multivariate
- **Subject Area:** Business
- **Associated Tasks:** Classification, Regression
- **Feature Type:** Real
- **Number of Instances:** 4898
- **Number of Features:** 11
- **Missing Values:** No

Dataset Insights

- **Data source:** The dataset is derived from physicochemical and sensory tests. No information is provided about grape types, wine brand, or pricing due to privacy and logistical constraints.
- **Task type:** The dataset can be used for both classification and regression tasks. In this project, we focus on the classification task by grouping wine quality into broader categories.
- **Class imbalance:** The dataset is imbalanced, with more samples representing average-quality wines than excellent or poor-quality ones.
- **Feature selection:** Some input variables may not be relevant, making this dataset suitable for testing feature selection methods.

Feature Description

- **Input Variables:**
    1. Fixed acidity
    2. Volatile acidity
    3. Citric acid
    4. Residual sugar
    5. Chlorides
    6. Free sulfur dioxide
    7. Total sulfur dioxide
    8. Density
    9. pH
    10. Sulphates
    11. Alcohol
- **Output Variable:**
    - Quality: A score between 0 and 10, categorized into three broader groups for analysis:
        1. Low quality: Classes 0-4
        2. Standard quality: Classes 5-6
        3. High quality: Classes 7-10

2.1 Data preparation

2.1.1 Import the dataset

The UC Irvine Machine Learning Repository allows for direct dataset import in Python using the `ucimlrepo` package. The package is included in the `requirements.txt` file. The following code demonstrates how to import the Wine Quality (id = 186) dataset using the `fetch_ucirepo` function:

```python
from ucimlrepo import fetch_ucirepo

breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=186)

X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets
```

2.1.2 Splitting the dataset

The dataset is split into training and testing sets using the `train_test_split` function from the `sklearn` library. The dataset is splitted into 4 sets of different test sizes: 60%, 40%, 20%, and 10% of the dataset in stratified fashion, using the random state of 22125 to ensure reproducibility.

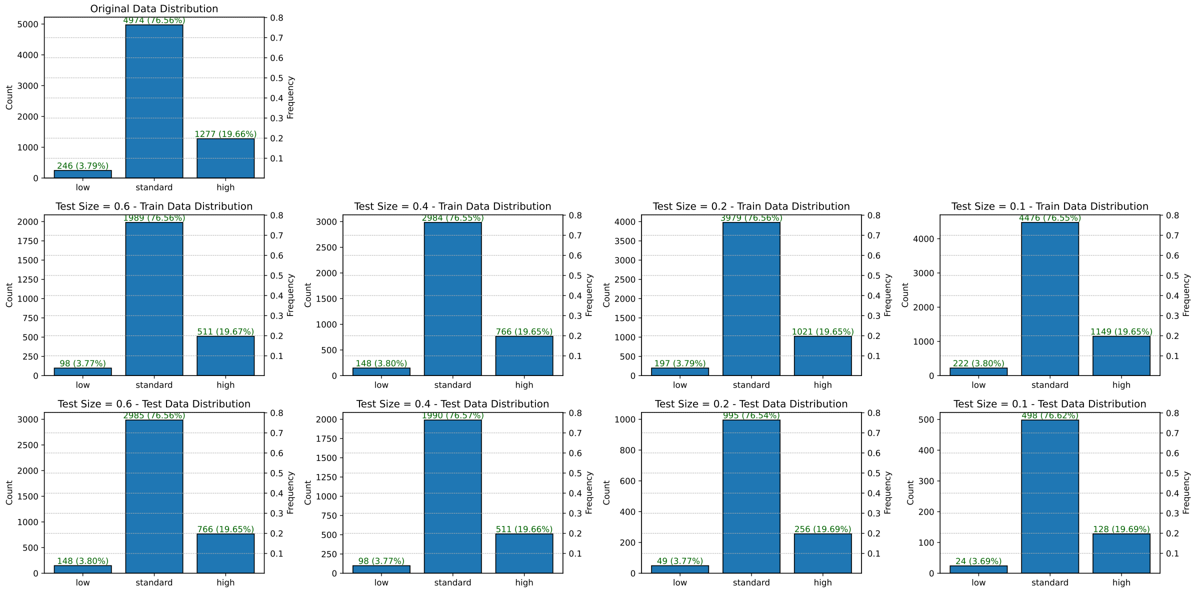


Figure 5: Wine Quality dataset split

As shown in figure 5, the distribution of the target class is preserved in all splits.

2.2 Decision Tree classifier implementation

Decision Tree classifiers are implemented using the `DecisionTreeClassifier` class from the `sklearn` library. To ensure reproducibility, the random state is set to 22125. We use the Entropy (information gain) criterion to split the nodes and set the maximum depth of the tree to None to allow the tree to grow until all leaves are pure.

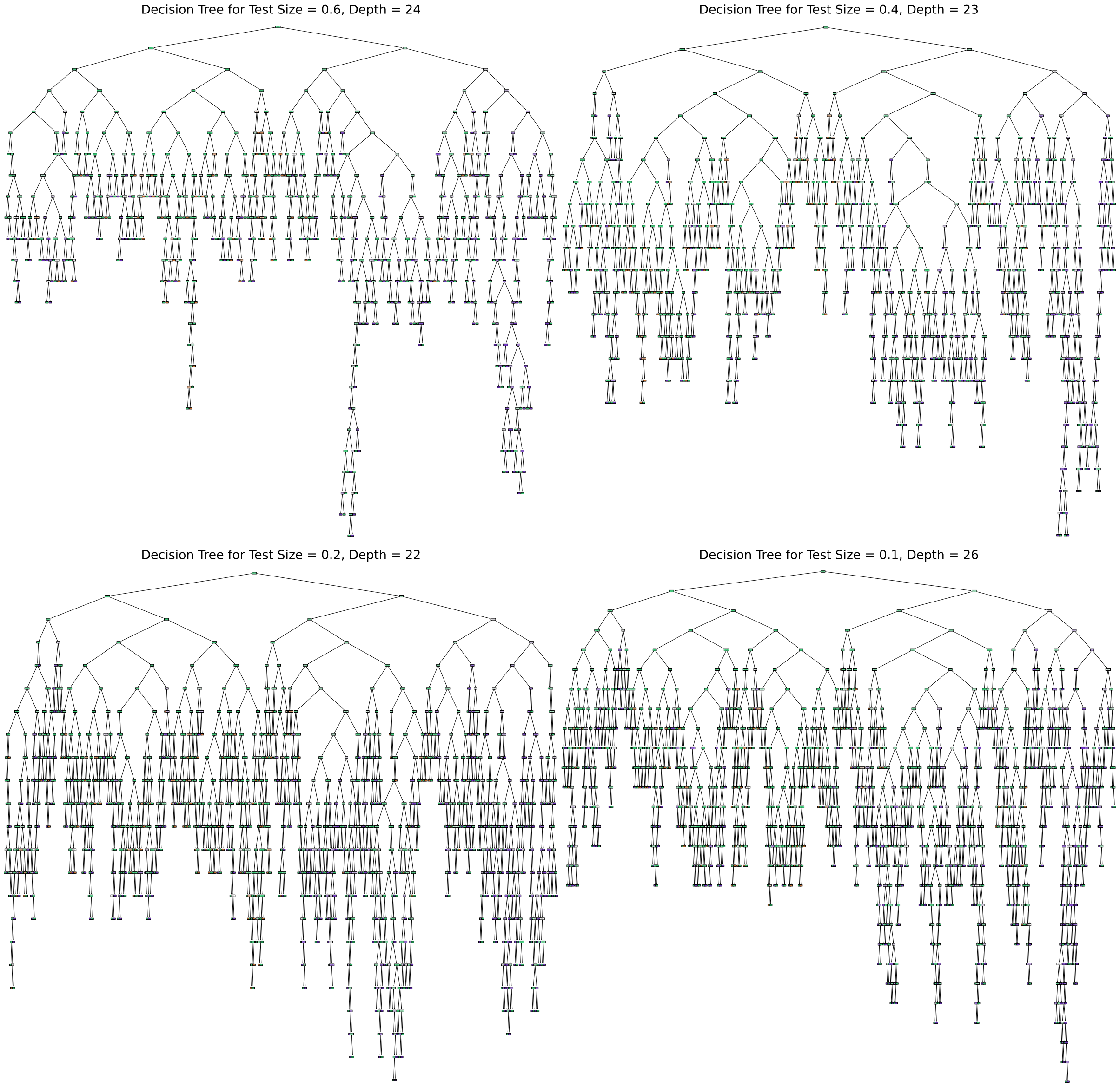

Decision Tree for Test Size = 0.6, Depth = 24
Decision Tree for Test Size = 0.4, Depth = 23
Decision Tree for Test Size = 0.2, Depth = 22
Decision Tree for Test Size = 0.1, Depth = 26

Figure 6: Wine Quality dataset Decision Trees with different test sizes

The Decision Trees are visualized in figure 6. The depths of the trees for the test sizes 60%, 40%, 20%, and 10% are 24, 23, 22, and 26 respectively. This indicates that the model complexity does not consistently increase or decrease with the test size.

The first node of the Decision Tree agrees on the feature `alcohol` to split the dataset for all test sizes. This feature is likely the most discriminative for classifying the wine quality in the dataset.

The first node's entropy is approximately 0.935 in all Decision Trees, indicating that the first split is not very informative. However, the Decision Tree classifier effectively splits the dataset into pure leaves with high information gain.

2.3 Performance evaluation

2.3.1 Classification report

The classification report provides a comprehensive evaluation of the Decision Tree classifier's performance using the `classification_report` function from the `sklearn.metrics` module. The report includes the following metrics:

- **Precision:** The ratio of true positive samples to the sum of true positive and false positive samples.
- **Recall:** The ratio of true positive samples to the sum of true positive and false negative samples.
- **F1-score:** The harmonic mean of precision and recall.
- **Support:** The number of samples in each class.

Classification Report for Test Size = 0.6

	Precision	Recall	F1-score	Support
Low	0.18	0.16	0.17	148
Standard	0.85	0.85	0.85	2985
High	0.53	0.55	0.54	766
Accuracy			0.76	3899
Macro avg	0.52	0.52	0.52	3899
Weighted avg	0.76	0.76	0.76	3899

Classification Report for Test Size = 0.4

	Precision	Recall	F1-score	Support
Low	0.15	0.16	0.16	98
Standard	0.85	0.86	0.85	1990
High	0.57	0.55	0.56	511
Accuracy			0.77	2599
Macro avg	0.52	0.52	0.52	2599
Weighted avg	0.77	0.77	0.77	2599

Classification Report for Test Size = 0.2

	Precision	Recall	F1-score	Support
Low	0.19	0.24	0.22	49
Standard	0.85	0.86	0.86	995
High	0.58	0.54	0.56	256
Accuracy			0.77	1300
Macro avg	0.54	0.55	0.54	1300
Weighted avg	0.77	0.77	0.77	1300

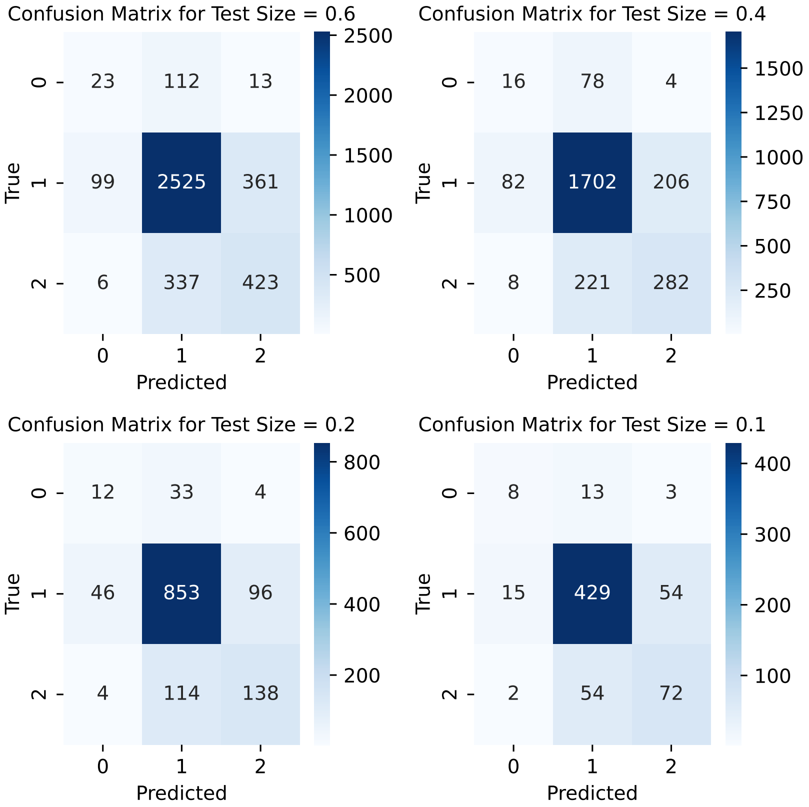
Classification Report for Test Size = 0.1

	Precision	Recall	F1-score	Support
Low	0.32	0.33	0.33	24
Standard	0.86	0.86	0.86	498
High	0.56	0.56	0.56	128
Accuracy			0.78	650
Macro avg	0.58	0.59	0.58	650
Weighted avg	0.78	0.78	0.78	650

The classification reports show that the Decision Tree classifier achieves high precision, recall, and F1-score for the standard quality class. The classifier performs poorly on the low-quality class due to the class imbalance in the dataset. The high-quality class has moderate performance, with precision, recall, and F1-score around 0.5-0.6.

2.3.2 Confusion matrix

The confusion matrix provides a visual representation of the Decision Tree classifier's performance using the `confusion_matrix` function from the `sklearn.metrics` module. The matrix shows the number of true positive, false positive, true negative, and false negative samples for each class.



Confusion Matrix for Test Size = 0.6

Confusion Matrix for Test Size = 0.4

Confusion Matrix for Test Size = 0.2

Confusion Matrix for Test Size = 0.1

Figure 7: Wine Quality dataset Confusion Matrices with different test sizes

The confusion matrices in figure 7 show that the Decision Tree classifier performs well on the standard quality class but struggles with the low and high-quality classes. The classifier misclassifies many low-quality samples as standard quality and high-quality samples as standard quality. This behavior is expected due to the class imbalance in the dataset.

2.4 Depth vs Accuracy evaluation

The Decision Tree classifier's performance is evaluated by varying the maximum depth of the tree. The accuracy of the classifier is computed for different maximum depths using the `accuracy_score` function from the `sklearn.metrics` module for the Decision Tree classifier with a test size of 20%.

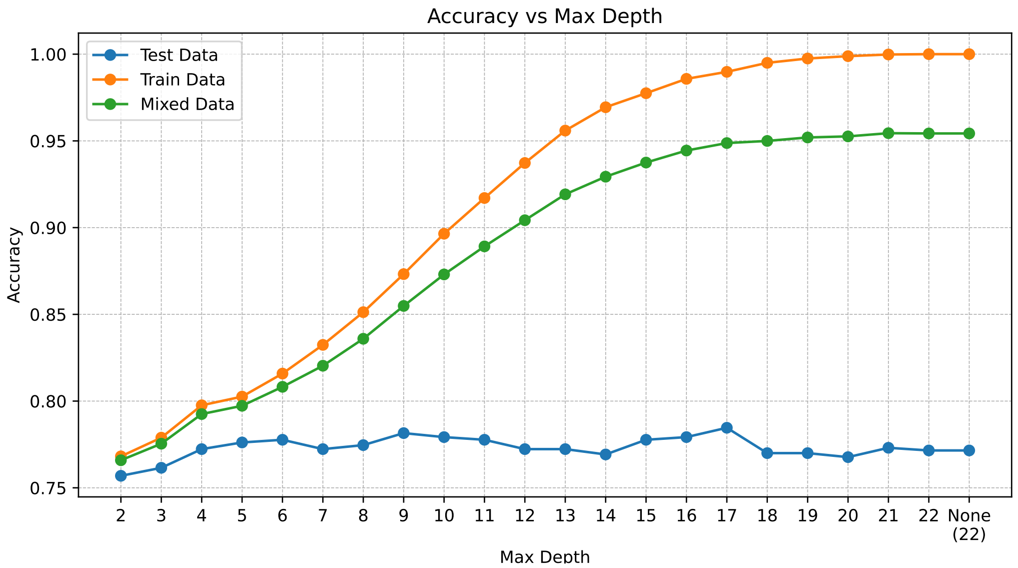


Figure 8: Wine Quality dataset Depth vs Accuracy

The plot in figure 8 shows that the Decision Tree classifier achieves the highest accuracy on the test data when the maximum depth is around 17. The accuracy slightly fluctuates around 0.77 for all depths, indicating that the model's performance is relatively stable.

Dataset 3: The UCI User Knowledge Modeling

In this Section, we will present the classification results of the dataset that we used in our project, which is the UCI User Knowledge Modeling dataset [3].

The User Knowledge Modeling dataset provides real-world data about students' knowledge status on the subject of Electrical DC Machines. The dataset was obtained as part of a Ph.D. thesis and has been analyzed to classify the knowledge levels of students based on various input features. The knowledge classes were determined in the thesis using a hybrid machine learning technique combining k-Nearest Neighbors (k-NN) with meta-heuristic exploration methods.

Dataset Characteristics

- **Type:** Multivariate
- **Subject Area:** Computer Science
- **Associated Tasks:** Classification, Clustering
- **Feature Type:** Integer
- **Number of Instances:** 403
- **Number of Features:** 5
- **Missing Values:** No

Dataset Insights

- **Objective:** To classify students' knowledge levels about Electrical DC Machines into predefined classes based on input features related to study habits and exam performance.
- **Classification Methodology:** The authors utilized an intuitive knowledge classifier, which combines k-Nearest Neighbor (k-NN) with meta-heuristic methods for classifying knowledge levels.
- **Class Distribution:** The dataset contains imbalanced classes:
  - Very Low: 50 instances
  - Low: 129 instances
  - Middle: 122 instances
  - High: 102 instances

Feature Description

- **Input Variables:**
  1. **STG:** The degree of study time for goal object materials.
  2. **SCG:** The degree of repetition of goal object materials.
  3. **STR:** The degree of study time for related objects with the goal object.
  4. **LPR:** The exam performance for related objects with the goal object.
  5. **PEG:** The exam performance for the goal objects.
- **Output Variable:**
  - **UNS:** The knowledge level of the user, classified into four categories:
    1. Very Low
    2. Low
    3. Middle
    4. High

3.1 Data preparation

3.1.1 Import the dataset

The UC Irvine Machine Learning Repository allows for direct dataset import in Python using the `ucimlrepo` package. The package is included in the `requirements.txt` file. The following code demonstrates how to import the Wine Quality (id = 257) dataset using the `fetch_ucirepo` function:

```
from ucimlrepo import fetch_ucirepo

breast_cancer_wisconsin_diagnostic = fetch_ucirepo(id=257)

X = breast_cancer_wisconsin_diagnostic.data.features
y = breast_cancer_wisconsin_diagnostic.data.targets
```

3.1.2 Splitting the dataset

The dataset is split into training and testing sets using the `train_test_split` function from the `sklearn` library. The dataset is splitted into 4 sets of different test sizes: 60%, 40%, 20%, and 10% of the dataset in stratified fashion, using the random state of 22125 to ensure reproducibility.

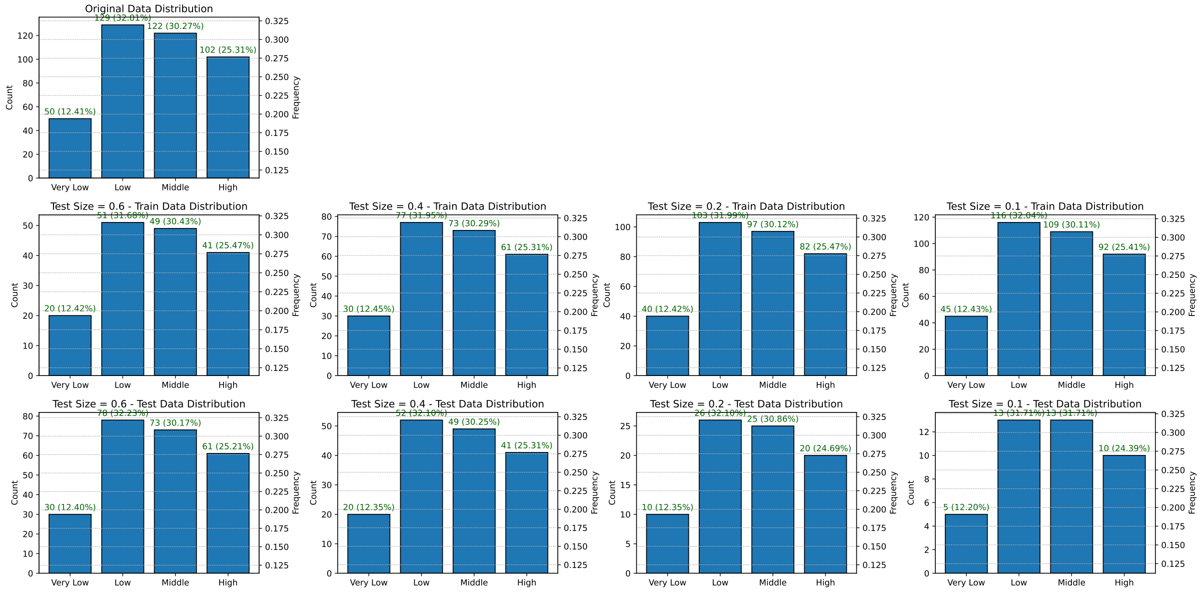


Figure 9: User Knowledge Modeling dataset splits

As shown in figure 9, the distribution of the target class is preserved in all splits.

3.2 Decision Tree classifier implementation

Decision Tree classifiers are implemented using the `DecisionTreeClassifier` class from the `sklearn` library. To ensure reproducibility, the random state is set to 22125. We use the Entropy (information gain) criterion to split the nodes and set the maximum depth of the tree to None to allow the tree to grow until all leaves are pure.

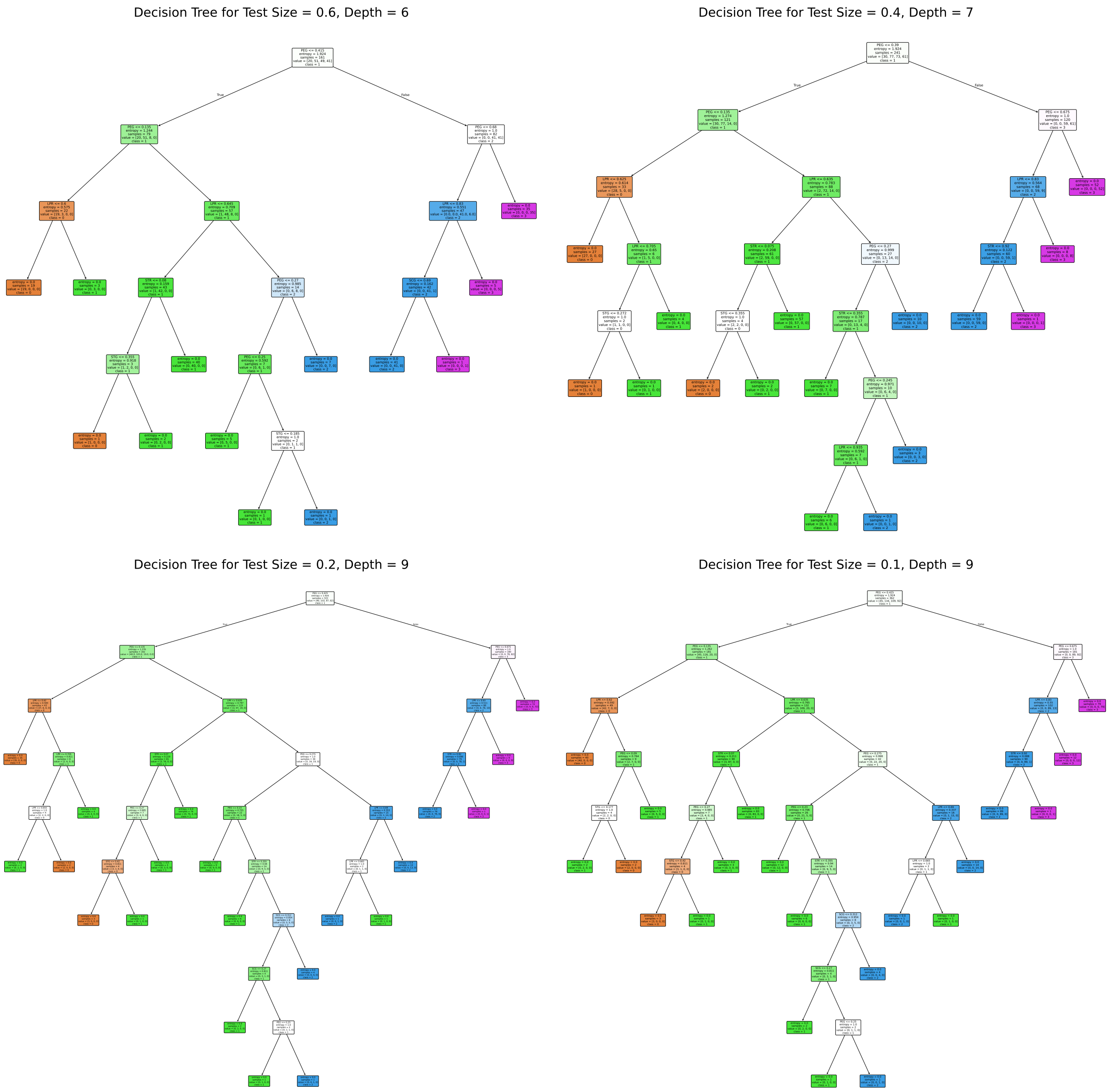

Decision Tree for Test Size = 0.6, Depth = 6
Decision Tree for Test Size = 0.4, Depth = 7
Decision Tree for Test Size = 0.2, Depth = 9
Decision Tree for Test Size = 0.1, Depth = 9

Figure 10: User Knowledge Modeling dataset Decision Trees with different test sizes

The Decision Trees are visualized in figure 10. The trees grow deeper as the test size decreases, indicating that the model is more complex and overfits the training data when the test size is small.

The first node of the Decision Tree agrees on the feature `PEG` to split the dataset for all test sizes. This feature is likely the most discriminative for classifying the knowledge levels of students.

The first node's entropy is approximately 1.924 in all Decision Trees, indicating that the first split is not very informative. However, the Decision Tree classifier effectively splits the dataset into pure leaves with high information gain.

3.3 Performance evaluation

3.3.1 Classification report

The classification report provides a comprehensive evaluation of the Decision Tree classifier's performance using the `classification_report` function from the `sklearn.metrics` module. The report includes the following metrics:

- **Precision:** The ratio of true positive samples to the sum of true positive and false positive samples.
- **Recall:** The ratio of true positive samples to the sum of true positive and false negative samples.
- **F1-score:** The harmonic mean of precision and recall.
- **Support:** The number of samples in each class.

Classification Report for Test Size = 0.6

	Precision	Recall	F1-score	Support
Very Low	0.89	0.80	0.84	30
Low	0.87	0.88	0.88	78
Middle	0.88	0.84	0.86	73
High	0.88	0.97	0.92	61
Accuracy			0.88	242
Macro avg	0.88	0.87	0.88	242
Weighted avg	0.88	0.88	0.88	242

Classification Report for Test Size = 0.4

	Precision	Recall	F1-score	Support
Very Low	0.84	0.80	0.82	20
Low	0.88	0.88	0.88	52
Middle	0.94	0.92	0.93	49
High	0.95	1.00	0.98	41
Accuracy			0.91	162
Macro avg	0.90	0.90	0.90	162
Weighted avg	0.91	0.91	0.91	162

Classification Report for Test Size = 0.2

	Precision	Recall	F1-score	Support
Very Low	1.00	0.70	0.82	10
Low	0.81	1.00	0.90	26
Middle	1.00	0.80	0.89	25
High	0.91	1.00	0.95	20
Accuracy			0.90	81
Macro avg	0.89	0.88	0.93	81
Weighted avg	0.90	0.90	0.92	81

Classification Report for Test Size = 0.1

	Precision	Recall	F1-score	Support
Very Low	1.00	0.40	0.57	5
Low	0.68	1.00	0.81	13
Middle	1.00	0.62	0.76	13
High	0.83	1.00	0.91	10
Accuracy			0.80	41
Macro avg	0.76	0.75	0.88	41
Weighted avg	0.79	0.80	0.86	41

The classification reports show that the Decision Tree classifier achieves high precision, recall, and F1-score for all classes in the test data. The model has an accuracy of approximately 0.88, 0.91, 0.90, and 0.80 for test sizes of 60%, 40%, 20%, and 10%, respectively. The model performs well in classifying the knowledge levels of students based on the input features.

3.3.2 Confusion matrix

The confusion matrix provides a visual representation of the Decision Tree classifier's performance using the `confusion_matrix` function from the `sklearn.metrics` module. The matrix shows the number of true positive, false positive, true negative, and false negative samples for each class.

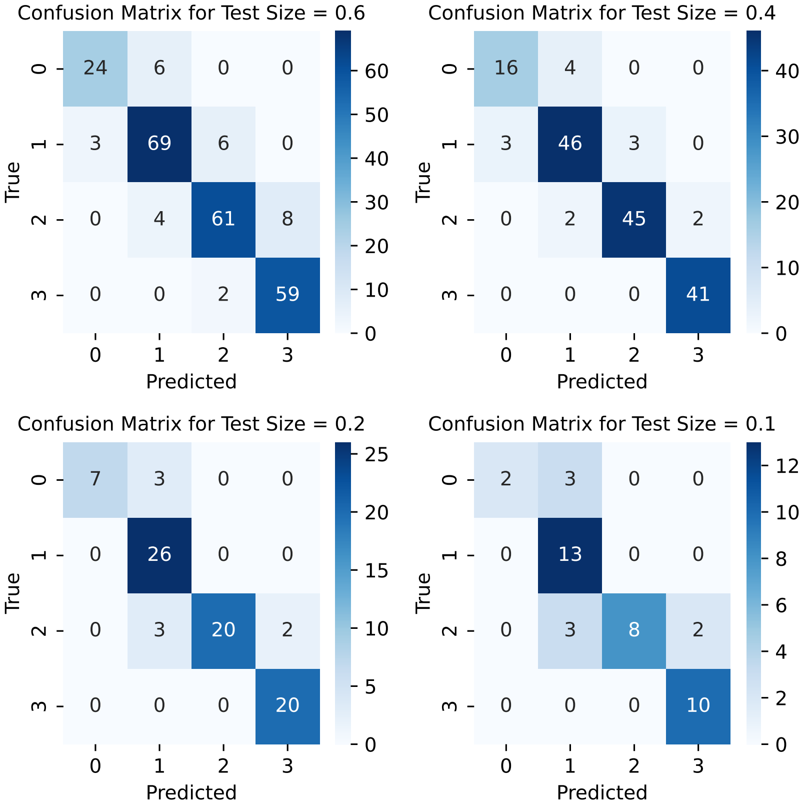


Figure 11: User Knowledge Modeling dataset Confusion Matrices with different test sizes

The confusion matrices in figure 11 show that the Decision Tree classifier correctly classifies most samples in the test data. The classifier has high true positive rates for all classes, indicating that the model effectively identifies the knowledge levels of students.

3.4 Depth vs Accuracy evaluation

The Decision Tree classifier's performance is evaluated by varying the maximum depth of the tree. The accuracy of the classifier is computed for different maximum depths using the `accuracy_score` function from the `sklearn.metrics` module for the Decision Tree classifier with a test size of 20%.

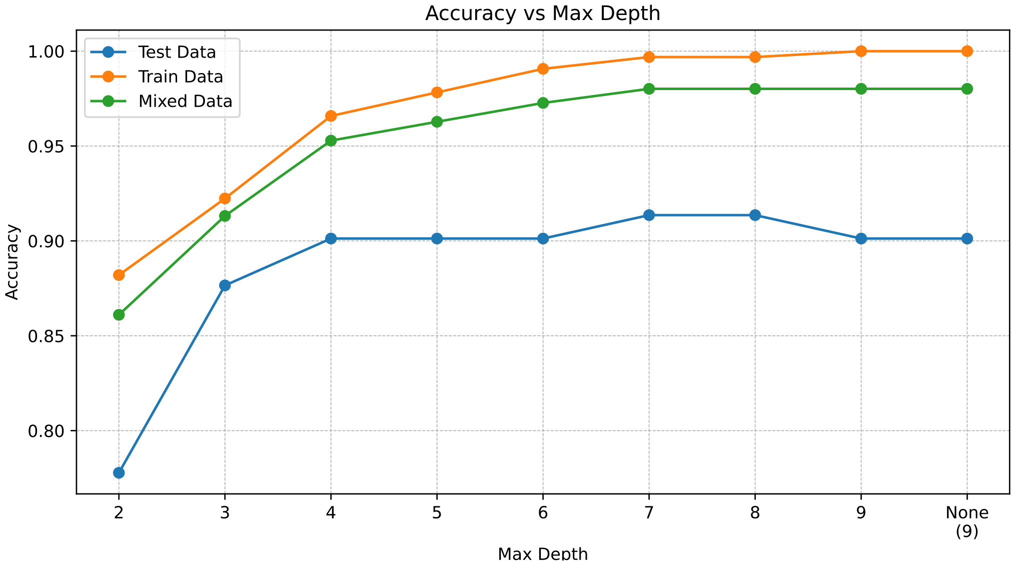


Figure 12: User Knowledge Modeling dataset Depth vs Accuracy

The plot in figure 12 shows that the Decision Tree classifier achieves the highest accuracy on the test data when the maximum depth is around 7. The accuracy decreases slightly as the maximum depth increases beyond 7, indicating that the model starts to overfit the training data but still performs well on the test data.

Conclusion

Self-evaluation

Contributions

What the hell is this?

References

[1] W. Wolberg, O. Mangasarian, N. Street, and W. Street, “Breast Cancer Wisconsin (Diagnostic).” UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.

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

[3] H. Kahraman, I. Colak, and S. Sagiroglu, “User Knowledge Modeling.” UCI Machine Learning Repository, 2009. DOI: https://doi.org/10.24432/C5231X.