Date of publication July 13, 2023
Digital Object Identifier

Implementation and Analysis of Decision Tree for Glass Classification Dataset

RASHMI KHADKA[1] (THA076BCT035), SHIWANI SHAH[2] (THA076BCT042)
[1]Department of Electronics and Computer Engineering, Thapathali Campus (e-mail: rashmikhadka6255@gmail.com)
[2]Department of Electronics and Computer Engineering, Thapathali Campus (e-mail: shahshiwani70@gmail.com)

ABSTRACT Decision trees are widely used machine learning models for classification tasks. In this report, we present the implementation of a decision tree model for the classification of glass types based on various attributes. The dataset consists of information such as refractive index, sodium content, magnesium content, aluminum content, silicon content, potassium content, calcium content, barium content, and iron content. The objective of this research is to develop an accurate decision tree model that can predict the type of glass based on these attributes. The performance of the model is evaluated using training and testing data, and the interpretability of the decision tree provides insights into the significance of different attributes in determining the glass type. This study has implications in industries such as glass manufacturing and product categorization, where precise classification of glass types is essential for quality control and inventory management. The results contribute to the understanding and application of machine learning techniques in the glass industry.

INDEX TERMS Classification, Confusion matrix, Decision tree, Entropy, Gini, Information Gain

I. INTRODUCTION

DECISION trees are a popular and powerful machine learning algorithm used for both classification and regression tasks. The nodes in the tree represent features or attributes, while the edges or branches represent the decisions or rules based on those features. At the leaf nodes, the decision tree provides predictions or class labels for the given input data.Decision trees offer interpretability and can handle both categorical and numerical features. They are widely used in various domains, such as finance, healthcare, and marketing, for their ability to provide intuitive decision-making rules and handle complex decision-making scenarios. The objective of this lab is to build a decision tree model that can accurately predict the type of glass based on the given attributes. Decision tree model is particularly suitable for this task as it can handle both continuous and categorical target variables. By constructing a decision tree from the training data and evaluating its performance on the testing data, we can assess the effectiveness of this model in predicting the glass type.

Understanding and accurately predicting the type of glass has various practical implications. For instance, in glass manufacturing, being able to classify glass types can aid in quality control and process optimization. In product categorization, it can help in ensuring appropriate classification of glass products for inventory management or market segmentation. By leveraging decision tree regression, we aim to contribute to the understanding and application of machine learning techniques in the glass industry.The big advantage of decision trees is the easy readability since one can always make a visual representation of the build decision trees.

II. METHODOLOGY

The Decision Tree Classification algorithm produces a tree structure, which gives fairly easy interpretation to the marketing people and easy identification of significant variables for the churn management. The model consists of a set of rules that can be used to predict the target variable. The Decision Tree approach provides a straightforward description of the data's underlying distribution, making it easier to understand and interpret the relationships between variables.

A. DECISION TREE FOR CLASSIFICATION

Decision trees are powerful supervised learning models widely used for classification tasks. They offer versatility, interpretability, and the ability to handle both numerical and categorical data. The decision tree structure consists of decision nodes representing features or attributes, and leaf nodes representing the final class labels or predictions. While decision trees can also be used for regression tasks, we will focus on their application in classification.

The construction of a decision tree involves recursively partitioning the data based on feature values to maximize the homogeneity or purity of subsets at each node. The splitting criterion, such as Gini impurity or entropy, is used to measure the impurity or disorder within subsets. The feature and threshold that minimize the impurity measure are selected as optimal split points during the tree-building process. This allows the tree to make decisions based on the most informative features.

During the decision-making process, the input data is traversed down the tree by answering questions or performing tests based on the feature values. Each test guides the traversal to a subsequent branch until reaching a leaf node. The class label of the leaf node is then assigned as the final prediction for the input data.

The goal of decision tree classification is to create a tree that accurately represents the underlying patterns and relationships in the data, enabling effective classification of new, unseen instances. By finding optimal splits that maximize information gain or decrease in impurity, decision trees can create more homogeneous subsets, improving the overall predictive accuracy of the model.

The graphical representation of a decision tree showcases its branching structure, starting from the root node and expanding into subtrees. Each subtree represents a specific path and set of conditions that lead to a particular class label. The interpretable nature of decision trees allows users to understand and interpret the decision rules, making them valuable for various classification tasks.

Overall, decision tree classification is a valuable technique for solving classification problems, offering a combination of accuracy, interpretability, and ease of implementation. The choice of appropriate splitting criteria and careful tree pruning can help optimize the model's performance and prevent overfitting.

B. STEPS OF DECISION TREE CLASSIFICATION

The following steps outline the process of building a decision tree for classification:

- **Step 1: Data Collection**
  Collect a dataset that contains both input features (predictors) and corresponding class labels (target variable). Ensure that the dataset represents a set of observations or examples with known class labels.
- **Step 2: Test-Train Data Splitting**
  Split the dataset into a training set and a testing set. The training set is used to build the decision tree classifier, while the testing set is used to evaluate its performance.
- **Step 3: Building the Decision Tree**
  Construct the decision tree by recursively partitioning the training set based on feature values. The goal is to create a tree that accurately predicts class labels while maintaining simplicity and interpretability.
- **Step 4: Choosing the Best Split**
  At each decision node, select the feature and threshold that best splits the data into the most homogeneous subsets. Different metrics such as Gini impurity or entropy can be used to measure the impurity or disorder within the subsets. The chosen metric aims to minimize the impurity and maximize the information gain.
- **Step 5: Pruning the Tree**
  After the decision tree is fully grown, consider pruning techniques to reduce overfitting. Pruning involves removing branches or nodes that do not significantly improve the model's performance on unseen data. This helps prevent the tree from becoming too complex and improves its generalization ability.
- **Step 6: Making Predictions**
  To make predictions, pass new input data through the decision tree by traversing the branches based on feature values. At each decision node, follow the appropriate branch until reaching a leaf node. The class label of the leaf node represents the predicted class for the input data.
- **Step 7: Evaluating the Model**
  Evaluate the performance of the decision tree classifier using the testing set. Calculate evaluation metrics such as accuracy, precision, recall, and F1 score to assess the model's effectiveness in predicting class labels.
- **Step 8: Using the Model**
  Once the decision tree classifier is trained and evaluated, it can be used to classify new, unseen data. By inputting the feature values into the trained model, the decision tree will determine the corresponding class label for the input data.

This procedure also stops when one of the following conditions is fulfilled:

- **Maximum tree depth is reached**
- **Minimum number of cases in node for being a parent is reached, so it can not be split any further.**
- **Minimum number of cases in node for being a child node is reached**

These conditions help control the complexity and prevent overfitting of the decision tree.

C. FORMULA USED

These formulas are used in decision tree algorithms to evaluate the quality of splits and select the best attribute for partitioning the data based on information gain or Gini index. Entropy and Gini impurity are two commonly used measures to determine the quality of a split in decision tree algorithms. They are used as splitting criteria to evaluate the homogeneity or impurity of subsets at each node. These formulas are used as measures to evaluate the quality of splits in decision tree algorithms. The feature and threshold that minimize these measures are chosen as optimal splitting points to create more homogeneous subsets.

- **Probability Vector**

$$Py(S) = \left(\frac{|y = c_1 S|}{|S|}, \ldots, \frac{y = c_{\text{dom}(y)} S|}{|S|} \right) \quad (1)$$

- **Entropy:** The entropy measures the impurity or disorder of a set of samples. In the context of decision trees, entropy is used to quantify the heterogeneity of the target variable (class labels) within a subset of data. The formula for entropy is as follows:

$$\text{Entropy}(y, S) = - \sum_{cj \in \text{dom}(y)} \frac{y = cjS}{|S|} \cdot \log_2 \left(\frac{y = cjS}{|S|} \right) \quad (2)$$

Or in simple form

$$Entropy = - \sum_{i=1}^{n} p_i \log_2(p_i) \quad (3)$$

where p_i represents the proportion of samples in a given class (i.e., class probability). The entropy value ranges from 0 to 1. A value of 0 indicates perfect homogeneity (all samples belong to the same class), while a value of 1 indicates maximum heterogeneity (samples are evenly distributed across all classes).

- **Information Gain:**

$$\text{IG}(ai, S) = \text{Entropy}(y, S) - \sum_{vi,j \in \text{dom}(ai)} \frac{|ai = vi, jS|}{|S|} \cdot \text{Entropy}(y, ai = vi, jS) \quad (4)$$

- **Gini Impurity:** The Gini impurity measures the probability of incorrectly classifying a randomly chosen element in a set. It quantifies the impurity or diversity of the target variable (class labels) within a subset of data. The formula for Gini impurity is as follows:

$$\text{Gini}(y, S) = 1 - \sum_{cj \in \text{dom}(y)} \left(\frac{y = cjS}{|S|} \right)^2 \quad (5)$$

Or, in simple way

$$GiniImpurity = 1 - \sum_{i=1}^{n} (p_i)^2 \quad (6)$$

where p_i represents the proportion of samples in a given class (i.e., class probability)
Consequently, the evaluation criterion for selecting the attribute ai is defined as:

$$\text{Ga}(a_i, S) = \text{Gini}(y, S) - \sum_{v_{i,j} \in \text{dom}(a_i)} \frac{\bar{\sigma}_{a_i = v_{i,j}} S}{|S|} \cdot \text{Gini}(y, \sigma_{a_i = v_{i,j}} S)$$

The Gini impurity value ranges from 0 to 1. A value of 0 indicates perfect homogeneity (all samples belong to the same class), while a value of 1 indicates maximum heterogeneity (samples are evenly distributed across all classes). The metrics below are commonly used to evaluate the performance of classification models, including decision tree classification. They provide insights into the model's accuracy, precision, recall, and overall predictive capability.

Confusion Matrix:

	Predicted Class	
Actual Class	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Accuracy:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \quad (8)$$

Precision:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \quad (9)$$

Recall:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \quad (10)$$

F1 Score:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

D. IMPLEMENTATION ON GLASS CLASSIFICATION DATASET

The Glass Identification Data Set from UCI consists of 214 samples with 10 attributes, including an identification number (id). The dataset focuses on various properties of glass, such as refractive index (RI), sodium content (Na), magnesium content (Mg), aluminum content (Al), silicon content (Si), potassium content (K), calcium content (Ca), barium content (Ba), and iron content (Fe) and the target attribute representing the type of glass.
The target variable of interest is the type of glass, which is a discrete attribute with 7 different values representing different glass types. The types include building windows (float processed and non-float processed), vehicle windows (float processed and non-float processed), containers, tableware, and headlamps.The id numbers are excluded from the provided CSV file, and the remaining attributes represent the weight percent of various elements present in the glass samples.
In the implementation of the decision tree on the glass classification dataset, a dataset obtained from Kaggle was used. The dataset already contained the tabular

values of records, attributes, and class for the glass samples. The categorical data in the dataset was transformed into numerical data using one-hot encoding.
Next, a decision tree model was instantiated, using entropy as the splitting criterion. The dataset was split into a training set and a test set, with 67
The decision tree model was then trained using the training dataset, and inference was performed on this dataset to obtain predicted class values. The predicted class values and ground truth class values of the test data points were compared and displayed.
The confusion matrix, which provides a summary of the model's performance, was computed based on the predicted and true class values. To visualize the confusion matrix, it was displayed as a heatmap. Additionally, a classification report was generated, providing more detailed evaluation metrics such as precision, recall, and F1-score for each class.Finally, the decision tree itself was plotted using the plot tree function, allowing for a visual representation of the decision-making process and the importance of different attributes in determining the glass type.

E. SYSTEM ARCHITECTURE

The system architecture for the glass classification dataset is shown in the FIGURE 1

III. INSTRUMENTATION

In the implementation of the decision tree on the Glass Classification dataset, we utilized several libraries for data manipulation, model training, and evaluation.

– **sklearn split model**: This library provides the split function, which is used to split the dataset into training and testing sets. It helps in evaluating the performance of the model on unseen data.
– **sklearn.tree**: This library contains the DecisionTreeClassifier class, which is used to create a decision tree model. It provides various parameters and methods to control the construction and behavior of the decision tree.
– **matplotlib.pyplot**: This library provides a collection of functions for creating plots and visualizations. In this case, it is used to plot the decision tree using the plot tree function and to create bar graphs using the plt.bar function.
– **sklearn.metrices**: From sklearn.metrices , we imported functions for confusion matrix and classification report. confusion matrix fuction used to compute the confusion matrix , which is a table that summarizes the performance of a classification model. It compares the predicted labels with the true labels and counts the number of true positives, true negatives, false positives, and false negatives. The resulting matrix provides insights into the model's accuracy, precision, recall, and other performance metrics. classification report function is used to generate a text report that includes various metrics for evaluating the performance of a classification model. It computes metrics such as precision, recall, F1-score, and support for each class in the dataset. The report provides a comprehensive summary of the model's performance, making it easier to assess its effectiveness in classifying the data.
– **seaborn**: This library is built on top of Matplotlib and provides additional functionality for creating visually appealing statistical graphics. In this case, it is used to create a heatmap for the confusion matrix using the sns.heatmap function.
– **numpy**: This library provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on them. It is used for various numerical operations and computations.
– **pandas**: This library provides data structures and data analysis tools. It is used for manipulating and analyzing the dataset, such as reading data from files, creating data frames, and performing operations on the data.

IV. RESULT

We implemented a decision tree algorithm on the glass classification dataset to classify different types of glass based on their chemical composition. The decision tree algorithm is a popular machine learning technique that creates a tree-like model of decisions and their possible consequences. It is particularly effective for classification tasks.
To perform the implementation, we used a Python programming language and the scikit-learn library, which provides efficient tools for machine learning tasks. The glass classification dataset consists of various features such as refractive index, sodium content, magnesium content, and more, which are used to predict the glass type. The class distribution of Glass classification dataset in shown in FIGURE 2
After that decision tree for the entropy split criterion was visualized which is shown in FIGURE 3
similarly for understanding model complexity, detecting overfitting, enhancing interpretability, and evaluating model performance of the decison tree of glass classification dataset, decision tree at diffrent depth level were visualized which can be seen in FIGURE 4, FIGURE 5 and FIGURE 6
Confusion matrix for various decision trees were also visualized which can be seen in FIGURE 7, FIGURE 8, FIGURE 9 and FIGURE 10
For understanding the model classification report was also visualized for diffrent depth of decision tree which can be seen in FIGURE 11, FIGURE 12, FIGURE 13 and FIGURE 14
Also the visualization of decision tree was done using

the gini index at diffrent depth labels which is shown in FIGURE 15, FIGURE 16, FIGURE 17 and FIGURE 18 Confusion matrix using gini index can be visualized at FIGURE 19, FIGURE 20, FIGURE 21 and FIGURE 22 Similarly classifiaction report using gini index can be seen at FIGURE 23, FIGURE 24, FIGURE 25 and FIGURE 26

V. DISCUSSION AND ANALYSIS

A. ANALYSIS OF GLASS CLASSIFICATION DATASET

The Glass classification dataset served as a suitable dataset for classification tasks, particularly for predicting the type of glass based on the provided attributes. The analysis involved building decision tree classification models to fulfill this objective. By examining decision trees at different depths and evaluating their performance using metrics such as accuracy, precision, recall, and F1 score, valuable observations were made.

First, decision trees were constructed using both entropy and Gini index as splitting criteria. The performance of the decision tree models was assessed by visualizing the trees themselves, observing the decision rules and attribute importance. This allowed for a better understanding of how different attributes contributed to the classification process.

Additionally, the confusion matrix provided insights into the performance of the decision tree models at different depths. By visualizing the confusion matrix as a heatmap, it was possible to observe the distribution of correct and incorrect predictions for each class. This facilitated the identification of any patterns or areas of improvement in the model's performance.

Furthermore, the classification report provided a comprehensive summary of the model's performance, including metrics such as precision, recall, and F1 score for each class. This report offered a detailed assessment of the model's ability to correctly classify instances from the testing set.

Overall, the analysis of decision trees at different depths, visualization of the trees, examination of the confusion matrix, and generation of the classification report provided valuable insights into the performance and interpretability of the decision tree classification models. By leveraging both entropy and Gini index as splitting criteria, this analysis contributed to a compre
evaluation of the Glass classification dataset a
lighted the potential applications of decision
glass manufacturing quality control and product
rization tasks

VI. CONCLUSION

Decision trees is a powerful tool for classificati
in data mining. Decision trees excel in dividing data into predefined classes and making accurate predictions for new data based on these classes. They achieve this by determining optimal splits at each decision node using impurity measures such as Gini impurity or entropy. By maximizing information gain and creating more homogeneous subsets, decision trees improve the overall predictive accuracy of the model.One of the notable advantages of decision trees is their interpretability. The tree structure provides a clear and intuitive representation of the decision-making process, allowing users to easily understand and evaluate the model's predictions. This interpretability is crucial in domains where explainability and transparency are desired, such as healthcare or finance.

Moreover, decision trees offer a practical and efficient solution to classification problems. They are computationally efficient, making them suitable for handling large datasets. Decision trees can handle both categorical and numerical features, making them versatile for various types of data.In conclusion, decision trees are a valuable tool for classification tasks in data mining. They provide accurate predictions, interpretability, and efficiency, making them a popular choice among practitioners and researchers. By understanding the principles and techniques behind decision trees, one can gain insights into the underlying patterns and relationships in the data, enabling informed decision-making in various domains.

VII. REFERENCES

[1] L. Rokach and O. Maimon, *"Data Mining with Decision Trees: Theroy and Applications"* . River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2008.

[2] Hagenlocher, P. *"Decision Tree Learning"*. Fakultät für Informatik, Technische Universität München.

[3] *Understanding the decision tree structure*, [Online]. Available: https://scikit-learn.org/stable/auto_examples/tree/plot_unveil_tree_structure.html.

[4] *Glass Classification*,[Online]. Available: https://www.kaggle.com/datasets/uciml/glass.

RASHMI KHADKA is a driven individual currently pursuing a Bachelor's degree in Computer Engineering at Thapathali Campus. She possesses a strong passion for machine learning and data mining, consistently seeking to delve into the latest advancements in these domains. While Rashmi may not have amassed notable achievements at this point, her enthusiasm and determination to learn and apply state-of-the-art technologies make her a promising and ambitious figure in the field of computer engineering.

SHIWANI SHAH is a dedicated individual currently studying Bachelor's in Computer Engineering at Thapathali Campus. With a strong passion for research and innovation, Shiwani actively engages in various projects and practical applications to apply his knowledge. Her continuous quest for learning drives her to stay updated with the latest advancements and trends in the field. Equipped with expertise in machine learning and data science, Shiwani's relentless determination, inquisitive mindset, and commitment to technology position her to make significant contributions to the ever-evolving landscape of the industry.

APPENDIX

A. FIGURES

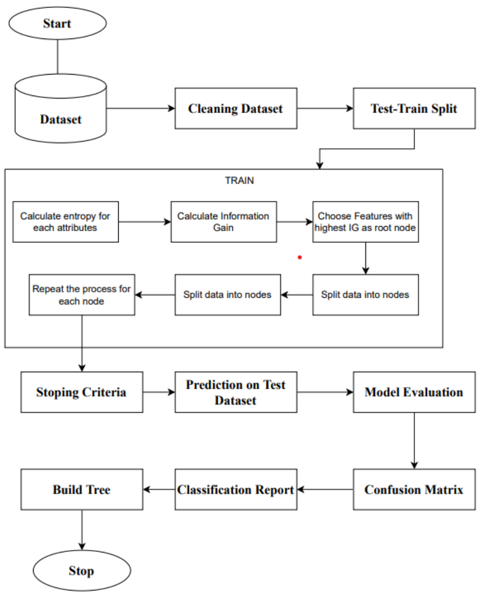


FIGURE 1: Block diagram of system Architecture

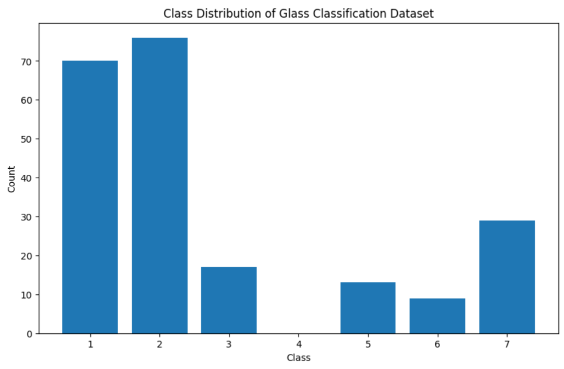


FIGURE 2: plot of proportion of variance of principal component

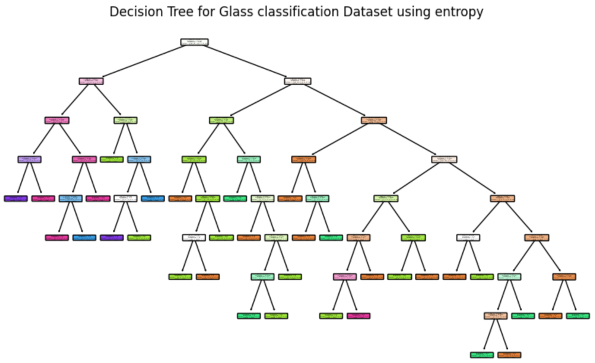

Decision Tree for Glass classification Dataset using entropy

FIGURE 3: Decision tree using entropy

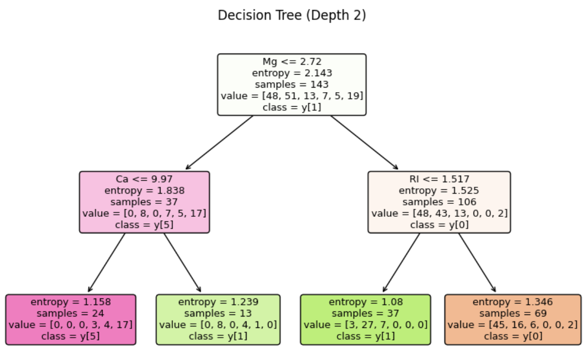


FIGURE 4: decision tree for depth 2

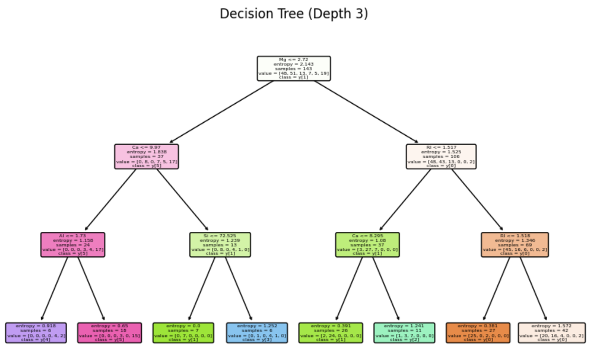


FIGURE 5: decision tree for depth 3

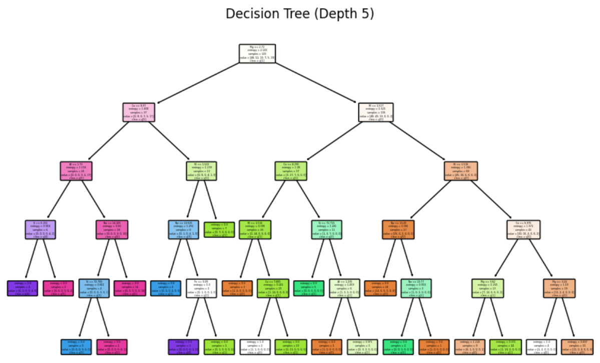


FIGURE 6: decision tree for depth 5

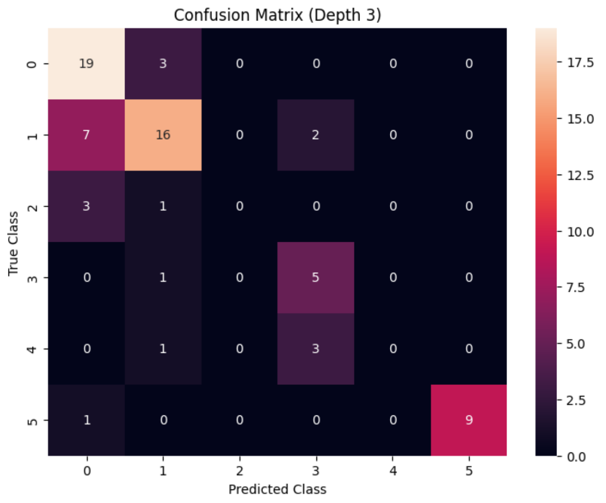


FIGURE 9: confusion matrix for depth 3

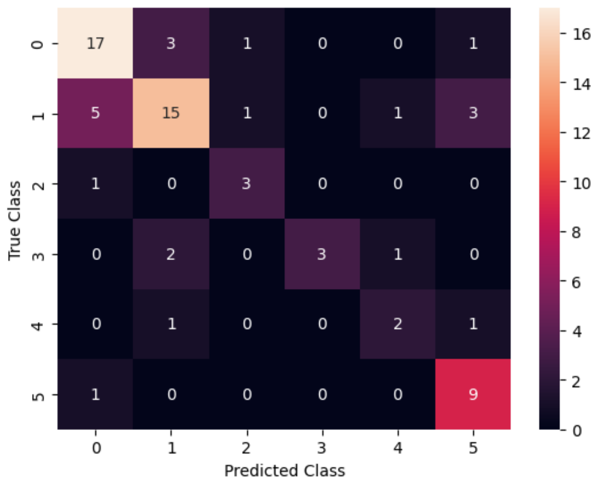


FIGURE 7: confusion matrix using entropy

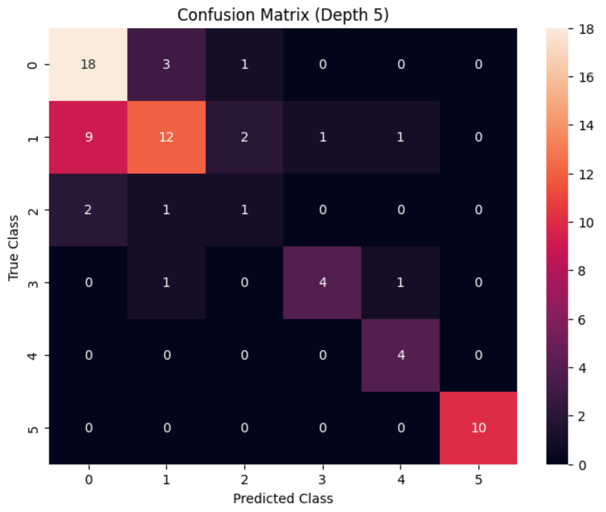


FIGURE 10: confusion matrix for depth 5

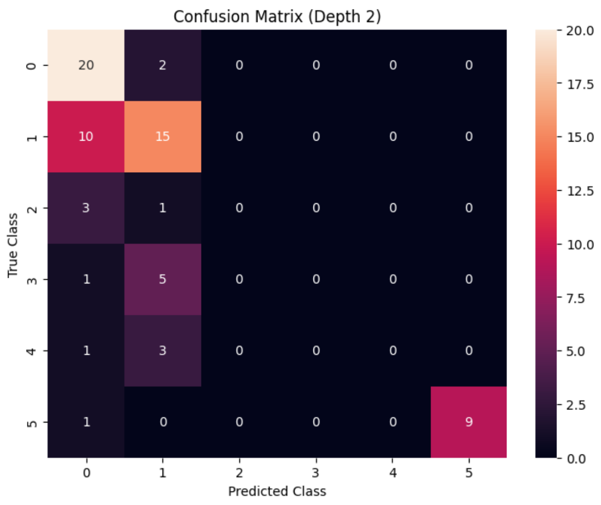
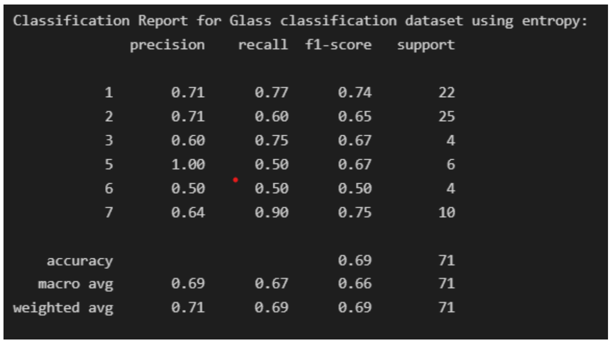


FIGURE 8: confusion matrix for depth 2

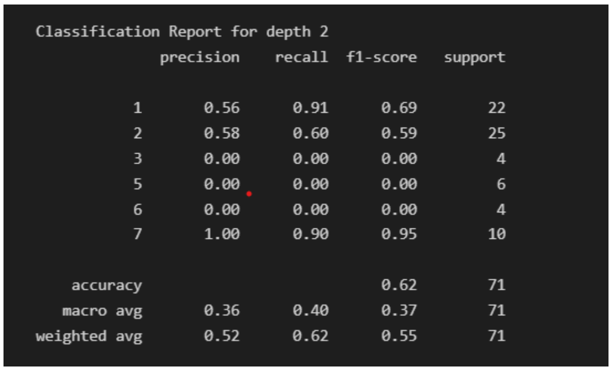


```
Classification Report for Glass classification dataset using entropy:
              precision    recall  f1-score   support

           1       0.71      0.77      0.74        22
           2       0.71      0.60      0.65        25
           3       0.60      0.75      0.67         4
           5       1.00      0.50      0.67         6
           6       0.50      0.50      0.50         4
           7       0.64      0.90      0.75        10

    accuracy                           0.69        71
   macro avg       0.69      0.67      0.66        71
weighted avg       0.71      0.69      0.69        71
```

FIGURE 11: classification report using entropy



```
Classification Report for depth 2
                precision    recall  f1-score   support

           1       0.56      0.91      0.69        22
           2       0.58      0.60      0.59        25
           3       0.00      0.00      0.00         4
           5       0.00      0.00      0.00         6
           6       0.00      0.00      0.00         4
           7       1.00      0.90      0.95        10

    accuracy                           0.62        71
   macro avg       0.36      0.40      0.37        71
weighted avg       0.52      0.62      0.55        71
```

FIGURE 12: classification report for depth 2

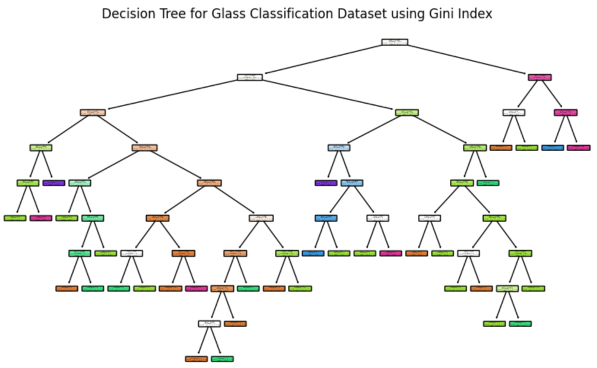
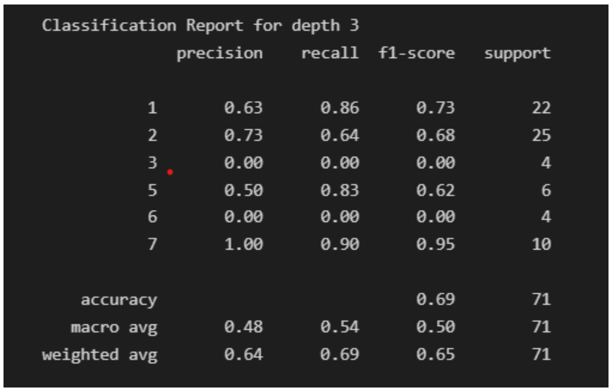

Decision Tree for Glass Classification Dataset using Gini Index

FIGURE 15: Decision tree using Gini Index



```
Classification Report for depth 3
                precision    recall  f1-score   support

           1       0.63      0.86      0.73        22
           2       0.73      0.64      0.68        25
           3       0.00      0.00      0.00         4
           5       0.50      0.83      0.62         6
           6       0.00      0.00      0.00         4
           7       1.00      0.90      0.95        10

    accuracy                           0.69        71
   macro avg       0.48      0.54      0.50        71
weighted avg       0.64      0.69      0.65        71
```

FIGURE 13: classification report for depth 3

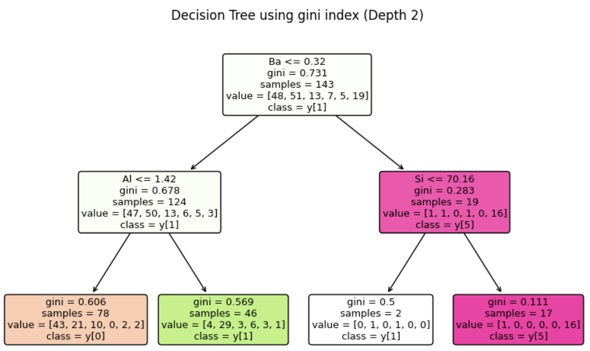
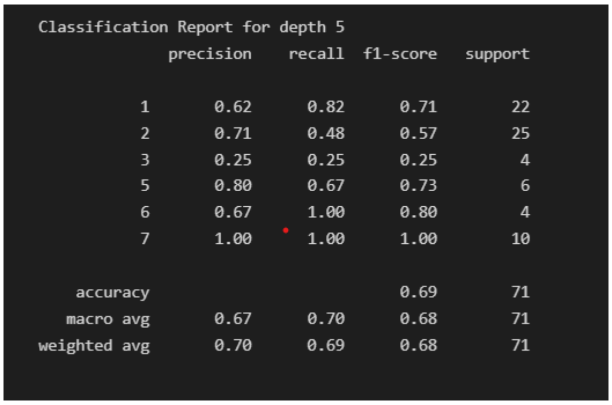


FIGURE 16: Decision tree at depth 2 using gini



```
Classification Report for depth 5
                precision    recall  f1-score   support

           1       0.62      0.82      0.71        22
           2       0.71      0.48      0.57        25
           3       0.25      0.25      0.25         4
           5       0.80      0.67      0.73         6
           6       0.67      1.00      0.80         4
           7       1.00      1.00      1.00        10

    accuracy                           0.69        71
   macro avg       0.67      0.70      0.68        71
weighted avg       0.70      0.69      0.68        71
```

FIGURE 14: classification report for depth 5

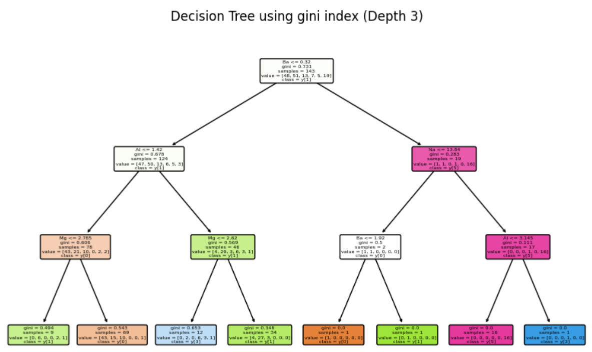


FIGURE 17: Decision tree at depth 3 using gini

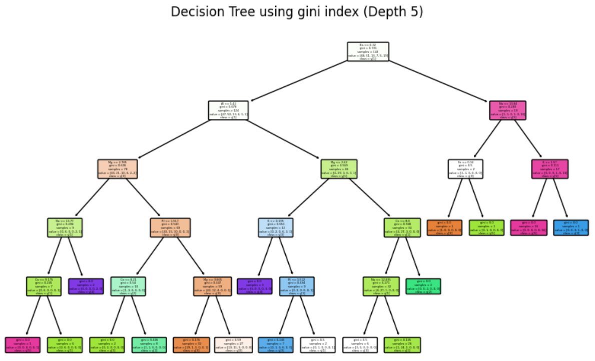


FIGURE 18: Decision tree at depth 5 using gini

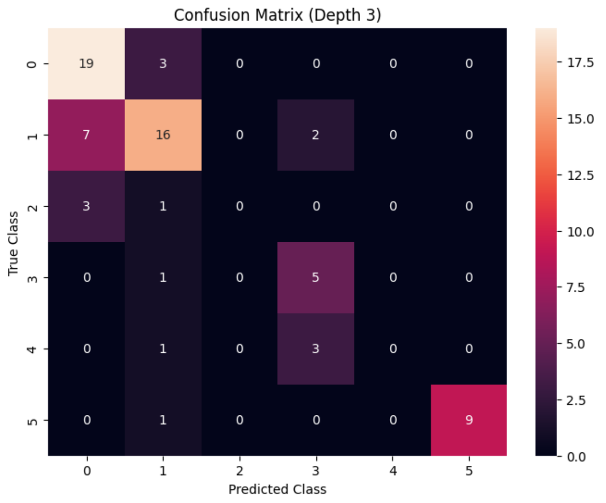


FIGURE 21: confusion matrix at depth 3 using gini

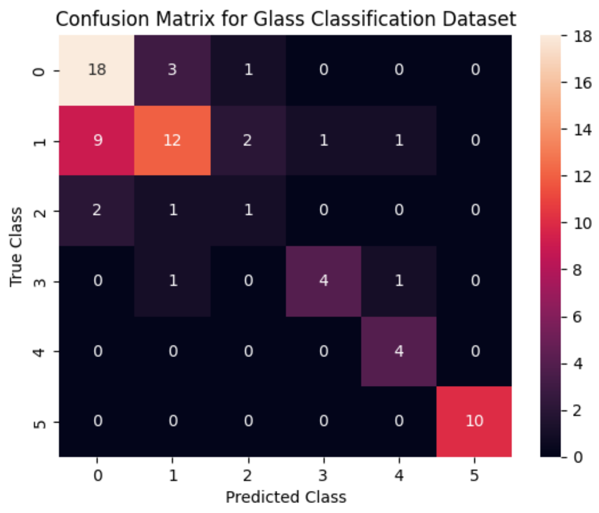


FIGURE 19: confusion matrix using gini

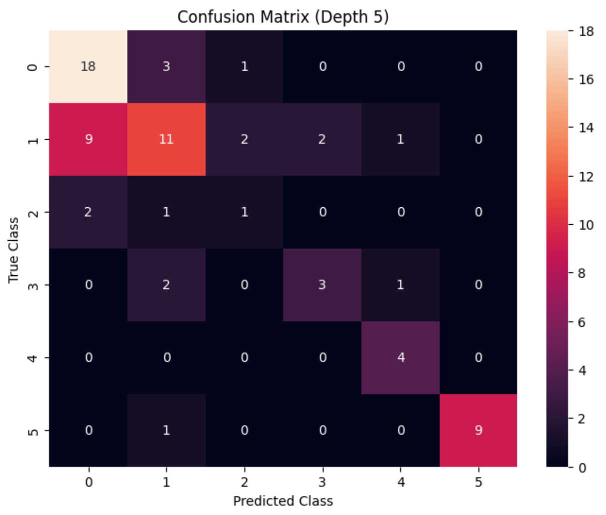


FIGURE 22: confusion matrix at depth 5 using gini

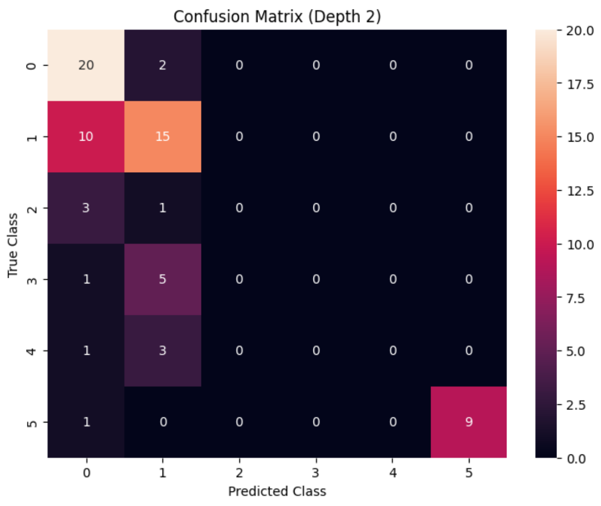
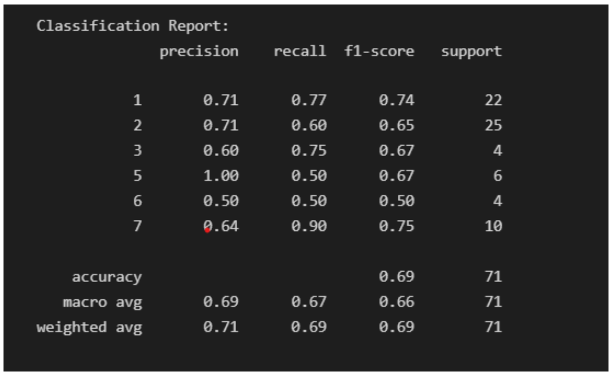


FIGURE 20: confusion matrix at depth 2 using gini

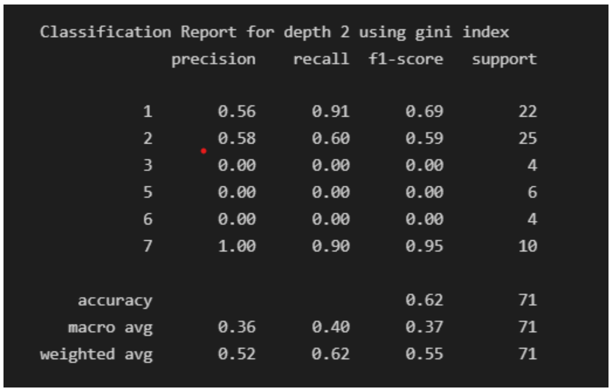


```
Classification Report:
              precision    recall  f1-score   support

           1       0.71      0.77      0.74        22
           2       0.71      0.60      0.65        25
           3       0.60      0.75      0.67         4
           5       1.00      0.50      0.67         6
           6       0.50      0.50      0.50         4
           7       0.64      0.90      0.75        10

    accuracy                           0.69        71
   macro avg       0.69      0.67      0.66        71
weighted avg       0.71      0.69      0.69        71
```

FIGURE 23: classification report using gini index

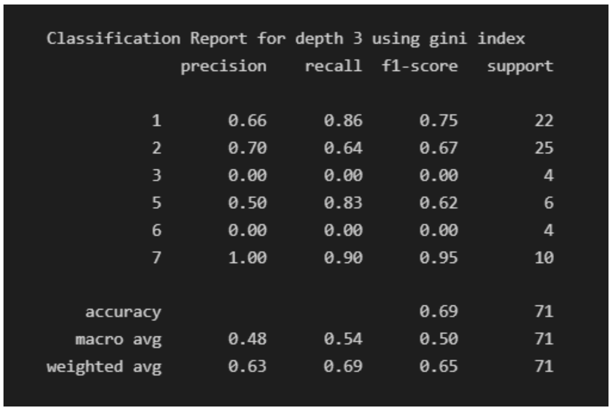


```
Classification Report for depth 2 using gini index
              precision    recall  f1-score   support

           1       0.56      0.91      0.69        22
           2       0.58      0.60      0.59        25
           3       0.00      0.00      0.00         4
           5       0.00      0.00      0.00         6
           6       0.00      0.00      0.00         4
           7       1.00      0.90      0.95        10

    accuracy                           0.62        71
   macro avg       0.36      0.40      0.37        71
weighted avg       0.52      0.62      0.55        71
```

FIGURE 24: classification report at depth 2 using gini index

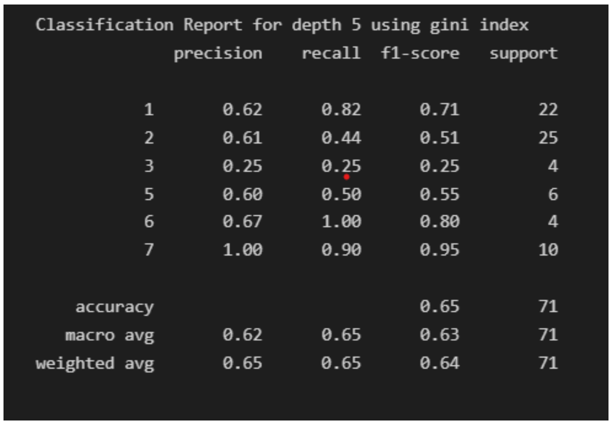


```
Classification Report for depth 3 using gini index
              precision    recall  f1-score   support

           1       0.66      0.86      0.75        22
           2       0.70      0.64      0.67        25
           3       0.00      0.00      0.00         4
           5       0.50      0.83      0.62         6
           6       0.00      0.00      0.00         4
           7       1.00      0.90      0.95        10

    accuracy                           0.69        71
   macro avg       0.48      0.54      0.50        71
weighted avg       0.63      0.69      0.65        71
```

FIGURE 25: classification report at depth 3 using gini index



```
Classification Report for depth 5 using gini index
              precision    recall  f1-score   support

           1       0.62      0.82      0.71        22
           2       0.61      0.44      0.51        25
           3       0.25      0.25      0.25         4
           5       0.60      0.50      0.55         6
           6       0.67      1.00      0.80         4
           7       1.00      0.90      0.95        10

    accuracy                           0.65        71
   macro avg       0.62      0.65      0.63        71
weighted avg       0.65      0.65      0.64        71
```

FIGURE 26: classification report at depth 5 using gini index

B. CODE

```
#Import all the necessary libraries
from sklearn.model_selection import
    train_test_split
from sklearn import tree
import matplotlib.pyplot as plt
from sklearn.tree import
    DecisionTreeClassifier
from sklearn.tree import plot_tree
from sklearn.metrics import confusion_matrix
    , classification_report
import seaborn as sns
import numpy as np
import pandas as pd

#Load the dataset
df= pd.read_csv('glass.csv')
df

#Information of dataset
df.info()


df.head(214)

#get the dunny dataset
df2 = pd.get_dummies(df)

df2.head(10)


# Decision tree using entropy

clf = tree.DecisionTreeClassifier(criterion=
    'entropy')


X = df2.iloc[:,:-1]
X.head()

y = df2.iloc[:, -1:]
y.head()


#.split the dataset such that 33% goes to
    test data and 67% goes to train data
X_train,X_test,y_train,y_test =
    train_test_split(X,y,test_size=0.33,
    random_state=42)



X_train.head()



X_test.head()


#concatenate the feature columns and class
    column to visualize the complete
    training data
training_data =pd.concat([X_train,y_train],
    axis=1)


#observe training data
training_data.head()



# fit the decision tree model
clf = clf.fit(X_train,y_train)
```

```
#make predictions on test data
predicted = clf.predict(X_test)



y_test.head()



# Get the class distribution from the
    dataset
class_counts = df['Type'].value_counts()

# Create a bar plot for class distribution
    of glass classification dataset
plt.figure(figsize=(10, 6))
plt.bar(class_counts.index, class_counts.
    values)
plt.xlabel('Class')
plt.ylabel('Count')
plt.title('Class Distribution of Glass
    Classification Dataset')
plt.show()



# Define the feature names
feature_names = X.columns.tolist()

# Plot the decision tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=feature_names,
    class_names=True, filled=True, rounded=
    True)
plt.title("Decision Tree for Glass
    classification Dataset using entropy")
plt.show()

# know depth of decision tree
clf = DecisionTreeClassifier().fit(X_train,
    y_train)
tree_depth = clf.tree_.max_depth
print(f"The depth of the decision tree is: {
    tree_depth}")


#gives tree with all the textual information
     of the tree
tree.plot_tree(clf)


#Confusion matrix in text form

confusion_mat = confusion_matrix(y_test,
    predicted)
print("Confusion Matrix:")
print(confusion_mat)


#Heatmap of confusion matrix
sns.heatmap(conf_mat,annot=True)
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.show()


#classification report of the decision tree
classification_report=classification_report(
    y_test, predicted)
print("Classification Report for Glass
    classification dataset using entropy:")
print(classification_report)


#Gives all the decision tree at the
    specified range of depth (optional)
# Initialize lists to store results
depths = []
confusion_matrices = []
classification_reports = []

# Define the range of depths to explore
min_depth = 1
max_depth = 10

# Iterate over different depths
for depth in range(min_depth, max_depth + 1)
    :
    # Create and fit the decision tree model
         with entropy criterion
    dt_model = DecisionTreeClassifier(
        max_depth=depth, criterion='entropy'
        )
    dt_model.fit(X_train, y_train)

    # Append the depth to the list
    depths.append(depth)

    # Plot the decision tree
    plt.figure(figsize=(10, 6))
    plot_tree(dt_model, feature_names=
        feature_names, filled=True, rounded=
        True, class_names=True)
    plt.title(f"Decision Tree (Depth {depth
        })")
    plt.show()

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

    # Compute confusion matrix and
        classification report
    cm = confusion_matrix(y_test, y_pred)
    cr = classification_report(y_test,
        y_pred)

    # Append the confusion matrix and
        classification report to the
        respective lists
    confusion_matrices.append(cm)
    classification_reports.append(cr)

# Print the confusion matrix and
    classification report for each depth
for depth, cm, cr in zip(depths,
    confusion_matrices,
    classification_reports):
    print(f"Depth {depth}:")
    print("Confusion Matrix:")
    print(cm)
    print("Classification Report:")
    print(cr)
    print("--------------------")



#Decision tree, confusion matrixa and
    classification report at depth 2
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import
    DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix
```

```
    , classification_report

# Define the feature names
feature_names = X.columns.tolist()

# Initialize the list to store depths
depths = []

# Create and fit the decision tree model
    with depth 2 and entropy criterion
clf = DecisionTreeClassifier(max_depth=2,
    criterion='entropy')
clf.fit(X_train, y_train)

# Append the depth to the list
depths.append(2)

# Plot the decision tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=feature_names,
    filled=True, rounded=True, class_names=
    True)
plt.title("Decision Tree (Depth 2)")
plt.show()

# Create and fit the decision tree model
    with a specific depth
depth = 2
dt_model = DecisionTreeClassifier(max_depth=
    depth)
dt_model.fit(X_train, y_train)

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

# Compute the confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True)
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title(f'Confusion Matrix (Depth {depth})
    ')
plt.show()

print("Classification Report for depth 2")
classification_report =
    classification_report(y_test, y_pred)
print(classification_report)



#Decision tree, confusion matrixa and
    classification report at depth 3
# Define the feature names
feature_names = X.columns.tolist()

# Initialize the list to store depths
depths = []

# Create and fit the decision tree model
    with depth 2 and entropy criterion
clf = DecisionTreeClassifier(max_depth=3,
    criterion='entropy')
clf.fit(X_train, y_train)

# Append the depth to the list
depths.append(3)

# Plot the decision tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=feature_names,
    filled=True, rounded=True, class_names=
    True)
plt.title("Decision Tree (Depth 3)")
plt.show()

# Create and fit the decision tree model
    with a specific depth
depth = 3
dt_model = DecisionTreeClassifier(max_depth=
    depth)
dt_model.fit(X_train, y_train)

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

# Compute the confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True)
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title(f'Confusion Matrix (Depth {depth})
    ')
plt.show()

print("Classification Report for depth 3")
classification_report =
    classification_report(y_test, y_pred)
print(classification_report)



#Decision tree, confusion matrixa and
    classification report at depth 5
# Define the feature names
feature_names = X.columns.tolist()

# Initialize the list to store depths
depths = []

# Create and fit the decision tree model
    with depth 2 and entropy criterion
clf = DecisionTreeClassifier(max_depth=5,
    criterion='entropy')
clf.fit(X_train, y_train)

# Append the depth to the list
depths.append(5)

# Plot the decision tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=feature_names,
    filled=True, rounded=True, class_names=
    True)
plt.title("Decision Tree (Depth 5)")
plt.show()

# Create and fit the decision tree model
    with a specific depth
depth = 5
dt_model = DecisionTreeClassifier(max_depth=
    depth)
dt_model.fit(X_train, y_train)

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

# Compute the confusion matrix
```

```
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True)
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title(f'Confusion Matrix (Depth {depth})
    ')
plt.show()

print("Classification Report for depth 5")
classification_report =
    classification_report(y_test, y_pred)
print(classification_report)



# Using Gini index

import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.tree import
    DecisionTreeClassifier, plot_tree
from sklearn.metrics import confusion_matrix
    , classification_report

# Create and fit the decision tree model
    with Gini index criterion
clf = DecisionTreeClassifier(criterion='gini
    ')
clf.fit(X_train, y_train)

# Plot the decision tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=feature_names,
    filled=True, rounded=True, class_names=
    True)
plt.title("Decision Tree for Glass
    Classification Dataset using Gini Index"
    )
plt.show()

clf = DecisionTreeClassifier().fit(X_train,
    y_train)
tree_depth = clf.tree_.max_depth
print(f"The depth of the decision tree is: {
    tree_depth}")


confusion_mat = confusion_matrix(y_test,
    predicted)
print("Confusion Matrix:")
print(confusion_mat)

import seaborn as sns
from sklearn.model_selection import
    train_test_split
from sklearn.tree import
    DecisionTreeClassifier
from sklearn.metrics import confusion_matrix
    , classification_report
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(conf_mat,annot=True)
plt.title('Confusion Matrix for Glass
    Classification Dataset')
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.show()


print("Classification Report:")
classification_report=classification_report(
    y_test, predicted)
print(classification_report)




#Decision tree, confusion matrixa and
    classification report at depth 2 using
    gini index
# Define the feature names
feature_names = X.columns.tolist()

# Initialize the list to store depths
depths = []

# Create and fit the decision tree model
    with depth 2 and entropy criterion
clf = DecisionTreeClassifier(max_depth=2,
    criterion='gini')
clf.fit(X_train, y_train)

# Append the depth to the list
depths.append(2)

# Plot the decision tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=feature_names,
    filled=True, rounded=True, class_names=
    True)
plt.title("Decision Tree using gini index (
    Depth 2)")
plt.show()

# Create and fit the decision tree model
    with a specific depth
depth = 2
dt_model = DecisionTreeClassifier(max_depth=
    depth)
dt_model.fit(X_train, y_train)

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

# Compute the confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True)
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title(f'Confusion Matrix (Depth {depth})
    ')
plt.show()

print("Classification Report for depth 2
    using gini index ")
classification_report =
    classification_report(y_test, y_pred)
print(classification_report)


#Decision tree, confusion matrixa and
    classification report at depth 3 using
    gini index
# Define the feature names
feature_names = X.columns.tolist()

# Initialize the list to store depths
depths = []
```

```
# Create and fit the decision tree model
    with depth 2 and Gini index criterion
clf = DecisionTreeClassifier(max_depth=3,
    criterion='gini')
clf.fit(X_train, y_train)

# Append the depth to the list
depths.append(3)

# Plot the decision tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=feature_names,
    filled=True, rounded=True, class_names=
    True)
plt.title("Decision Tree using gini index (
    Depth 3)")
plt.show()

# Create and fit the decision tree model
    with a specific depth
depth = 3
dt_model = DecisionTreeClassifier(max_depth=
    depth)
dt_model.fit(X_train, y_train)

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

# Compute the confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True)
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title(f'Confusion Matrix (Depth {depth})
    ')
plt.show()

# Print the classification report
print("Classification Report for depth 3
    using gini index")
cr = classification_report(y_test, y_pred)
print(cr)



#Decision tree, confusion matrixa and
    classification report at depth 5 using
    gini index
# Define the feature names
feature_names = X.columns.tolist()

# Initialize the list to store depths
depths = []

# Create and fit the decision tree model
    with depth 2 and Gini index criterion
clf = DecisionTreeClassifier(max_depth=5,
    criterion='gini')
clf.fit(X_train, y_train)

# Append the depth to the list
depths.append(5)

# Plot the decision tree
plt.figure(figsize=(10, 6))
plot_tree(clf, feature_names=feature_names,
    filled=True, rounded=True, class_names=
    True)
plt.title("Decision Tree using gini index (
    Depth 5)")
plt.show()

# Create and fit the decision tree model
    with a specific depth
depth = 5
dt_model = DecisionTreeClassifier(max_depth=
    depth)
dt_model.fit(X_train, y_train)

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

# Compute the confusion matrix
conf_mat = confusion_matrix(y_test, y_pred)
print(conf_mat)

# Plot the heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(conf_mat, annot=True)
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.title(f'Confusion Matrix (Depth {depth})
    ')
plt.show()

# Print the classification report
print("Classification Report for depth 5
    using gini index")
cr = classification_report(y_test, y_pred)
print(cr)
```

•••