MUHAMMAD SAIF (22F-3165)

SITARA SHABBIR (22F-3838)

MACHINE LEARNING

ASSIGNMENT # 2

October 8, 2024

**Submitted to:** Dr. Hashim Yasin

**Deliverables:**

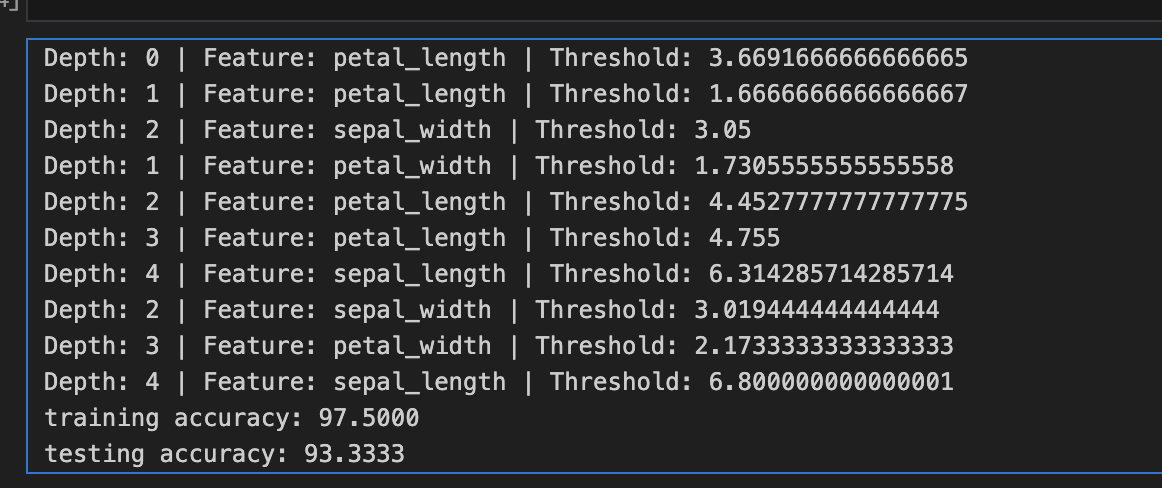
* Three notebook files are attached with this report each containing code and outputs.
* Visualized outputs are also attached as required in the manual.
* Codes in the notebook file are commented for further clarification.
* This report includes code descriptions, outputs, accuracies, evaluations and discussions required in the assignment manual.

**QUESTION 1:**

In the first part, we trained a decision tree model using ID3 (Iterative dichotomiser 3) on the given Iris dataset.   
In this process we,

* Preprocessed the data.
* Calculated the entropy and the information gain.
* Then we found the best split using the mean as the threshold.
* Then we recursively built the tree until it reaches a stopping criterion.
* We also calculated different thresholds and calculated accuracy of both splits (training and testing).
* We used evaluation metrics like precision, recall and f1 score to evaluate the performance of the model.
* After that, we plotted the graph of decision tree using matplotlib.

The results below show thresholds that were created at each depth and the accuracy of both training and testing sets.



This picture shows evaluation results that were calculated using evaluation metrics.

A screenshot of a computer

Description automatically generated

For the second part, we trained the model on the same Iris dataset but this time we used CART (Classification and Regression trees) method.

Here we,

* Used same preprocessing techniques.
* Instead of entropy, we calculated Gini impurity and information gain to evaluate splits.
* For each feature, we evaluated multiple thresholds and found the best split that maximized information gain.
* We recursively built the decision tree until stopping criteria like maximum depth or minimum samples were reached.
* After training, we applied tree pruning to simplify the model and prevent overfitting by removing branches that didn't improve validation accuracy.

Here in the below output recall measures how well a model can identify all the positive cases. It focuses on capturing as many true positives as possible, showing the proportion of actual positives that the model correctly classified. A higher recall means fewer positive instances were missed.

The F1 score combines both precision and recall into one metric, balancing the two. It's useful when you need a balance between catching positive instances (recall) and making sure the positive predictions are accurate (precision). The F1 score is especially helpful when dealing with imbalanced datasets.

A screen shot of a computer

Description automatically generated

**COMPARISON OF THE RESULTS**

**ID3 and CART**

The results show that the ID3 and CART algorithms achieved different training accuracies, with ID3 reaching 97.5% and CART achieving 99.17%. Both algorithms performed equally well on the testing dataset, with an accuracy of 93.33%. Interestingly, the algorithms had different precision, recall, and F1 scores for both training and testing data.

On the training data, ID3 had a precision of 0.9756, recall of 0.9744, and an F1 score of 0.9740, while CART had a precision of 0.9919, recall of 0.9917, and an F1 score of 0.9917. For the testing data, ID3 showed precision at 0.9487, recall at 0.9444, and an F1 score of 0.9419, whereas CART had precision at 0.9436, recall at 0.9333, and an F1 score of 0.9331.

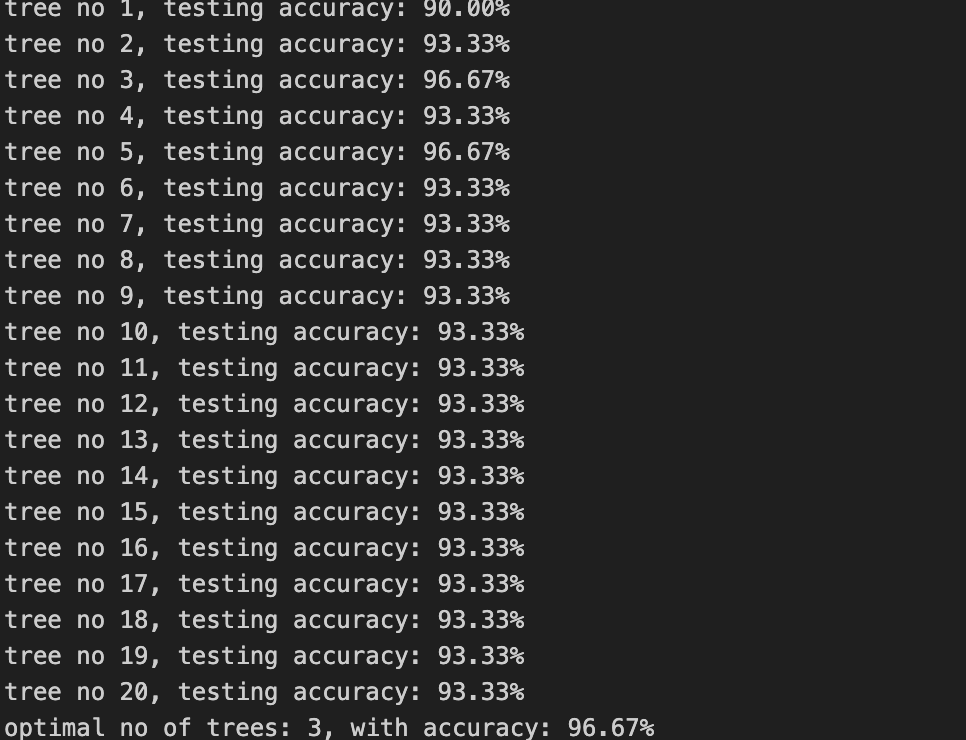
Overall, both algorithms performed well, but CART showed superior performance on the training dataset, while their performance on the testing dataset was comparable, suggesting that for this problem, ID3 and CART are effective classification methods but with notable differences in training efficiency.

**QUESTION 2:**

In this question we trained a model on random forest.

Here we,

* Preprocessed the data by selecting features and mapping them to specific values.
* Evaluated the Gini index to evaluate the quality of the splits.
* built a decision tree recursively until reaching a maximum depth or homogeneity in samples.
* Used majority voting for final predictions and assessed performance by calculating accuracy on the test set.
* Identified the optimal number of trees for the best accuracy.
* Visualized the relationship between tree count and accuracy, and displayed feature importance to understand which features significantly influenced predictions.
* We also trained a model on only two features sepal length, sepal width to visualize a decision boundary graph.



The output displays the testing accuracy of the Random Forest model as the number of trees increases from 1 to 20. Initially, accuracy improves significantly, rising from 90.00% with one tree to 93.33% with two trees. The most notable increase occurs with the third tree, which achieves an accuracy of 96.67%, showing that the model learns better with more trees at first.

After the third tree, accuracy stabilizes at 93.33% for most additional trees, indicating diminishing returns. The optimal number of trees is found to be 3, with a peak accuracy of 96.67%. This suggests that fewer trees can still deliver strong performance, emphasizing the importance of balancing model complexity and efficiency in Random Forests.

In a previous iteration, the optimal number of trees was found to be 9, demonstrating that this value can vary based on different training and testing splits or dataset characteristics. This variability arises from factors such as data complexity and feature relationships, highlighting the importance of experimentation in machine learning. It underscores the need for careful tuning, as the optimal configuration may change with variations in data or model parameters.

The optimal number of trees in a Random Forest model varies depending on the dataset and aims to balance model accuracy with computational efficiency, as too few trees may underfit while too many can lead to diminishing returns in performance.

**QUESTION 3:**

In this question, we trained a model using Naïve Bayes classifier.

Here,

* We loaded and prepared the Iris dataset.
* Performed a manual train-test split, allocating 80% of the data for training and 20% for testing.
* Then we applied Min-Max normalization to scale the feature values between 0 and 1.
* We implemented a Naïve Bayes classifier by calculating the mean, variance, and prior probabilities for each class.
* Finally, we predicted the test set labels and calculated the accuracy of the model, visualizing the data with a pair plot.

**Results obtained after visualizing the graph:**

The pair plot that we obtained from our model shows the relationships between different features (sepal length, sepal width, petal length, and petal width) of three Iris flower species: Iris-setosa, Iris-versicolor, and Iris-virginica. The diagonal plots display the distribution of each feature, while the scatterplots show how features are related across species.

By looking at the scatterplots, we can see that petal length and petal width provide the best separation between the species, especially for Iris-setosa. The clear distinction between species in these features makes them useful for classification.

We used this graph to visualize how our Naive Bayes classifier performs on the dataset. It helps us understand how well the features separate the species and shows which features are more likely to help our model make accurate predictions.