Practical Lab Report: Implementation of a Decision Tree Classifier

1. Introduction and Objective

The objective of this practical laboratory exercise was to implement a core Decision Tree (DT) classifier from first principles using Python and NumPy. The implementation focused on the core recursive algorithm, utilizing **Gini Impurity** as the criterion for evaluating the quality of potential data splits. The final goal was to build a complete tree and assess its performance on a synthetic, linearly separable dataset.

2. Methodology

2.1 Data Generation and Structure

A synthetic dataset comprising 200 two-dimensional feature points ($\mathbf{X}$) was generated using random values between 0 and 1. The corresponding binary labels ($\mathbf{y}$) were assigned based on a simple, noiseless, linearly separable rule:

$$y = \begin{cases} 1.0 & \text{if } X_{[:,0]} < X_{[:,1]} \\ 0.0 & \text{otherwise} \end{cases}$$

The entire dataset was processed into a single array of rows, where each row has the structure: `[Feature 0, Feature 1, Label]`. This structure facilitates easy iteration and partitioning by the classifier's helper functions.

2.2 Algorithm Implementation

The classifier was built around the standard recursive tree induction pseudocode, implemented through five core components:

1. `Question` **Class:** Defines a potential split based on a feature column and a continuous threshold value (e.g., "Is Feature $i \geq v$?").
2. **Gini Impurity (** `gini` **):** Measures the heterogeneity of labels within a subset of data. A Gini value of 0 indicates perfect purity.

$$Gini(S) = 1 - \sum_{i=1}^{C} p_i^2$$

3. **Information Gain (** `information_gain` **):** Quantifies the reduction in Gini Impurity achieved by a proposed split. The objective of the algorithm is to maximize this value.

$$\text{Gain} = \text{Gini(Parent)} - [\text{Weighted Average of Gini(Children)}]$$

4. **Optimal Split (`optimal_split`):** Iterates through every unique value across every feature to find the `Question` that yields the maximum Information Gain.
5. **Recursive Tree Building (`decisionTree`):**
    - Calls `optimal_split` .
    - **Stopping Condition:** Halts recursion and returns a **Leaf Node** (class distribution dictionary) if the `Information Gain` is 0.
    - Otherwise, it splits the data and recurses on the resulting two subsets, returning a **Decision Node** (containing the optimal question and its two branches).

3. Results

3.1 Training Accuracy

Upon training the custom Decision Tree classifier on the synthetic dataset, the model achieved the following performance metrics on the training set:

Metric	Value
Total Examples Tested	200
Accuracy on Training Data	**100.00%**

3.2 Decision Boundary Visualization

A plot generated using a comparable `scikit-learn` classifier confirmed that the trained model perfectly partitioned the feature space. The boundary is characteristic of a Decision Tree, composed entirely of axis-parallel (vertical and horizontal) lines that approximate the underlying diagonal separation.

4. Conclusion and Discussion

The lab successfully implemented a functional Decision Tree classifier, capable of recursively partitioning data based on the Gini Impurity criterion.

The resulting **100% accuracy on the training data is expected** due to the specific, noiseless nature of the generated synthetic dataset. Since the data is perfectly separable by a simple function ($X_{[:,0]} = X_{[:,1]}$), the unconstrained Decision Tree is able to find the exact sequence of axis-parallel splits necessary to perfectly classify every point, effectively learning the generating rule without error.

In this context, the perfect training accuracy is a sign of **successful implementation**, not problematic overfitting. However, it serves as an important reminder that for real-world datasets containing noise, achieving 100% accuracy on the training set is a strong indicator of **overfitting**, necessitating model regularization techniques such as setting a `max_depth` or minimum leaf size to improve generalization to unseen data.