CART - CLASSIFICATION AND REGRESSION TREE FOR CLASSIFICATION

Yixun Kang, David Ning, Liang Zhang

UC Providence

December 12, 2024

GitHub Link: https://github.com/ihiroo/DATA2060_Final_Project.git

DATA

- **Heart Disease Dataset [1]**
  - Contains multiple biomarkers to predict the presence or absence of heart disease
  - **13 + 1 features and 1025 observations**
  - Our target variable is **binary**
  - Includes categorical (binary) and continuous (or ordinal) variables
  - This is an **IID** dataset
- **Iris Dataset [2]**
  - Data loaded from **Scikit-learn**
  - **4 + 1 features and 150 observations**
  - Our target variable is **categorical** (3 classes)
  - This is an **IID** dataset
  - Used to test the implementation from scratch

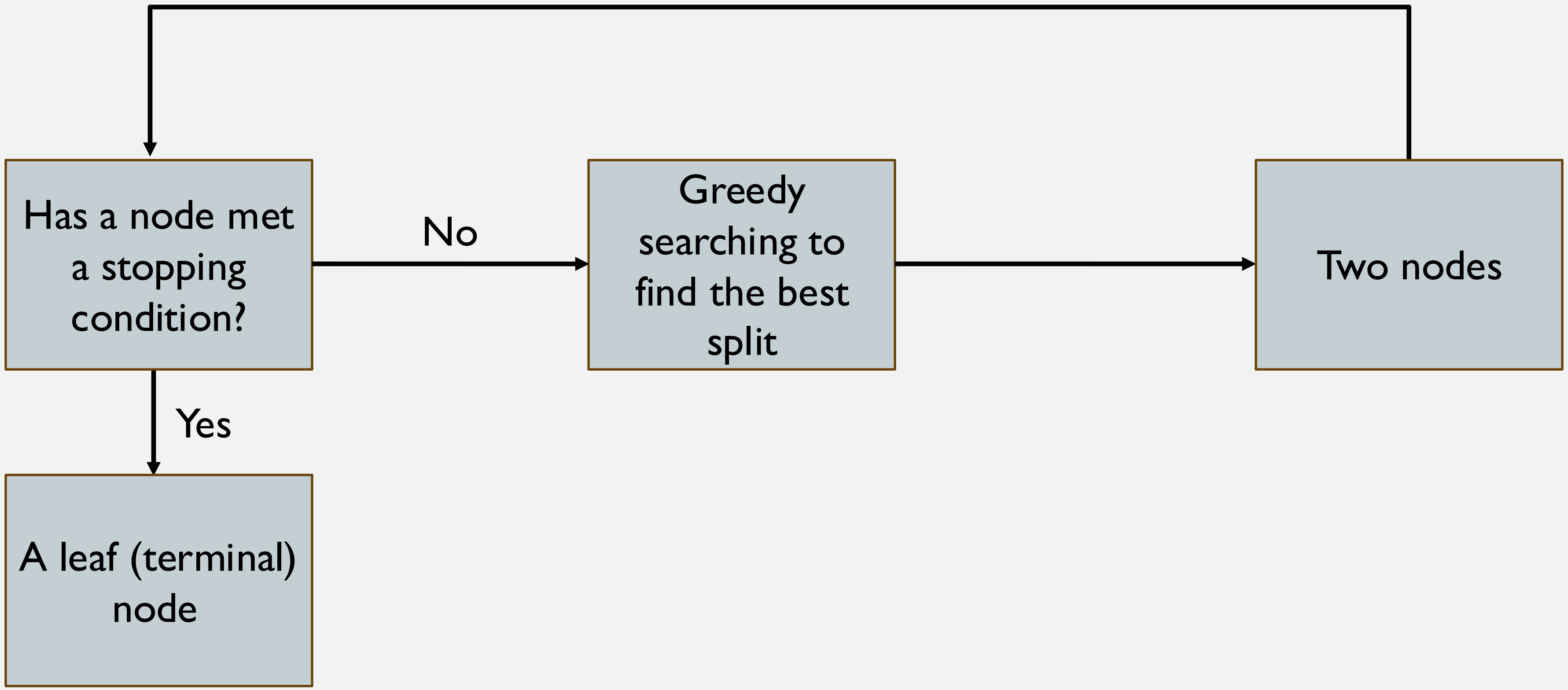
CART

- **Goal:** implement Scikit-learn's `DecisionTreeClassifier` from scratch, with support for the parameters `max_depth` and `min_samples_split`, achieving similar accuracy and replicating the tree structure
  - Consider `max_depth` and `min_samples_split` as stopping conditions when building the tree
  - Use **cross-validation** when pruning to select the best tree
  - Conduct **cost-complexity** pruning
  - Run the model with different random states

ML ALGORITHMS

Aspects	Details
Representation	Binary tree which splits based on features, thresholds and gains (greedy search)
Loss	**Gini impurity**, entropy, or misclassification error
Optimizer	Recursive **top-down** cost-complexity pruning

REPRESENTATION



REPRESENTATION

Algorithm 1 CART for Binary Classification

Inputs: (i) training data S; (ii) feature subset $A \subseteq [d]$; (iii) max_depth; (iv) min_samples_split

    if all samples in S are labeled by 1 **then** return a leaf node labeled by 1
    if all samples in S are labeled by 0 **then** return a leaf node labeled by 0
    if $A = \emptyset$ **then** return a leaf whose value = majority of labels in S
    if max_depth $= 0$ **then** return a leaf whose value = majority of labels in S
    if $|S| <$ min_samples_split **then** return a leaf whose value = majority of labels in S
    else
        Compute $\mathrm{Gini}(S) \leftarrow 1 - \{[\Pr(y=1|S)]^2 + [\Pr(y=0|S)]^2\}$
        foreach feature i in A **do**
            Generate a sequence of thresholds θ based on the sorted unique values of $\mathbf{x}_i$
            foreach threshold θ in the sequence **do**
                $S_1 = \{(\mathbf{x}, y) \in S : x_i \leq \theta\}$
                $S_2 = \{(\mathbf{x}, y) \in S : x_i > \theta\}$
                $\mathrm{Gini}(S_1, S_2, S) = \frac{|S_1|}{|S|} \cdot \mathrm{Gini}(S_1) + \frac{|S_2|}{|S|} \cdot \mathrm{Gini}(S_2)$
                $\mathrm{Gain}(S_1, S_2, S) = \mathrm{Gini}(S) - \mathrm{Gini}(S_1, S_2, S)$
        Let $j = \mathrm{argmax}_{i \in A} \mathrm{Gain}(S_1, S_2, S)$
        $S_1 = \{(\mathbf{x}, y) \in S : x_j \leq \theta\}$
        $S_2 = \{(\mathbf{x}, y) \in S : x_j > \theta\}$
        Let T_1 be the tree returned by CART(S_1, A, max_depth $- 1$, min_samples_split)
        Let T_2 be the tree returned by CART(S_2, A, max_depth $- 1$, min_samples_split)
    Return the tree

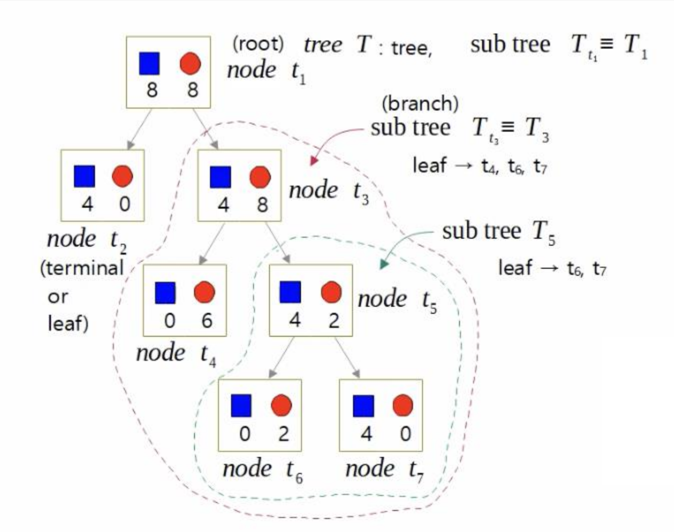
- This algorithm works for feature matrix with continuous, ordinal and **binary categorical** features
- Works fine for the Heart Disease Dataset since features are either continuous, ordinal, or binary
- Ordinal features are **encoded** to numeric values so they are treated as continuous features
- For binary categorical features, this algorithm will generate only one threshold, which is **0.5** (the midpoint of 1 and 0)
- For an arbitrary binary feature $\mathbf{x}_i$, after one split, the subset dataset will only contain one unique value, so it's unnecessary to remove $\mathbf{x}_i$ from A ($\mathbf{x}_i$ will not be used to split again for sure)

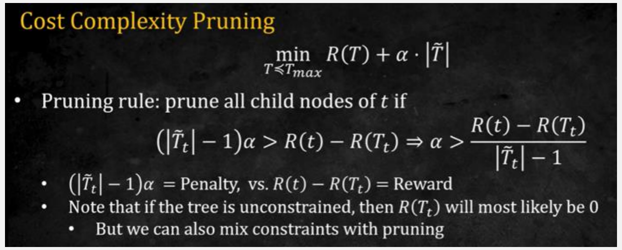
LOSS

- Parent Gini: $\mathrm{Gini}(S) = 1 - \sum_{k=1}^{K} [\Pr(y = k|S)]^2 = 1 - \sum_{k=1}^{K} p_k^2$
  - $\Pr(y = k|S)$ is the proportion of samples in the dataset S that belong to class k
    - $\Pr(y = k|S) = p_k$
  - K is the number of unique classed in the dataset
  - In a binary classification problem, $K = 2$ and $\mathrm{Gini}(S) = 1 - (p_1^2 + p_2^2)$
    - If set $p_1 = a$, $\mathrm{Gini}(S) = 1 - (a^2 + (1-a)^2) = 2a(1-a)$
- Weighted Gini (or Gini for Split): $\mathrm{Gini}(S_1, S_2, S) = \frac{|S_1|}{|S|} \cdot \mathrm{Gini}(S_1) + \frac{|S_2|}{|S|} \cdot \mathrm{Gini}(S_2)$
- Gain: $\mathrm{Gain}(S_1, S_2, S) = \mathrm{Gini}(S) - \mathrm{Gini}(S_1, S_2, S)$

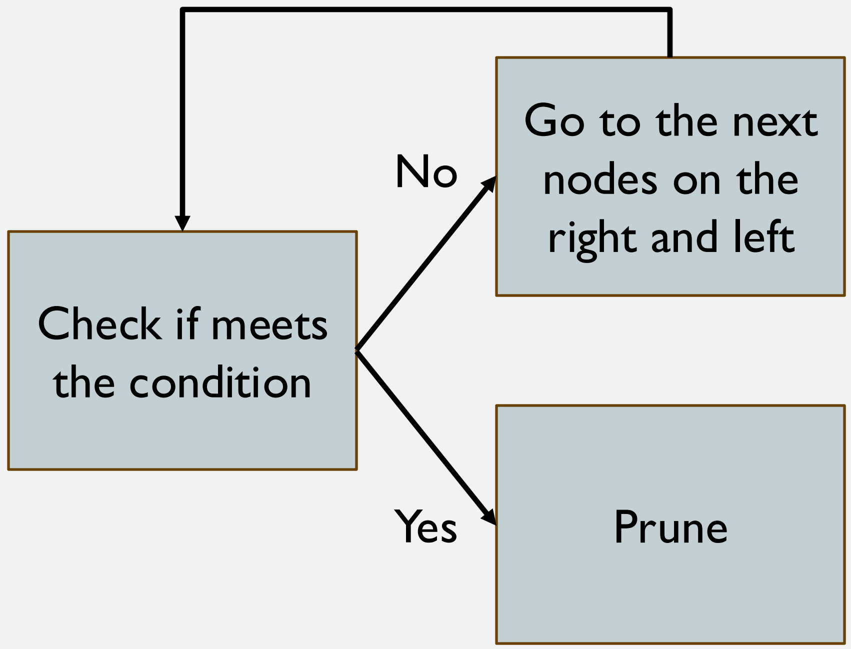
OPTIMIZER: PRUNING

- Global cost function: $\boldsymbol{R_\alpha(T) = R(T) + \alpha \cdot |T|}$
  - For a given α, the optimal pruned tree T^* is $T^* = \underset{T}{\operatorname{argmin}} R_\alpha(T)$
- Local pruning rule:
  - Prune T_t if $\boldsymbol{\alpha} > \frac{\boldsymbol{R(t) - R(T_t)}}{\boldsymbol{|T_t| - 1}}$
  - Prune T_t if $(|T_t| - 1) \cdot \alpha > R(t) - R(T_t)$
- Notations:
  - T: the entire decision tree
  - T_t: the subtree rooted at node t
  - $|T|$: number of leaf nodes in T
  - $R(T)$: misclassification cost of T, $R(T) = \sum_{t \in \text{leaves of } T} R(t)$
  - $R(t)$: misclassification cost if t is a leaf node (0-1 loss)
  - $\alpha \geq 0$: complexity parameter, penalizing the number of leaf nodes



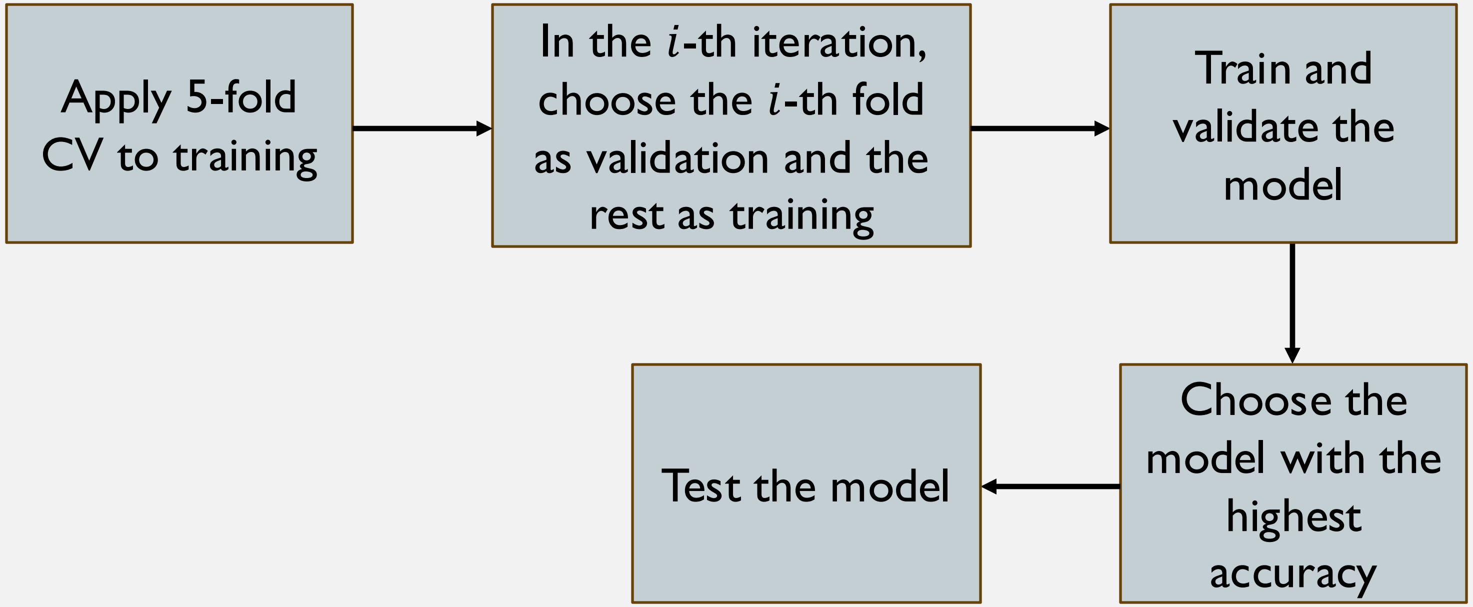


OPTIMIZER: PRUNING



CROSS VALIDATION

- Split the entire dataset into train/test = 8/2
- Apply 5-fold cross-validation to the training data to ensure robustness
- For each model/set of parameters:

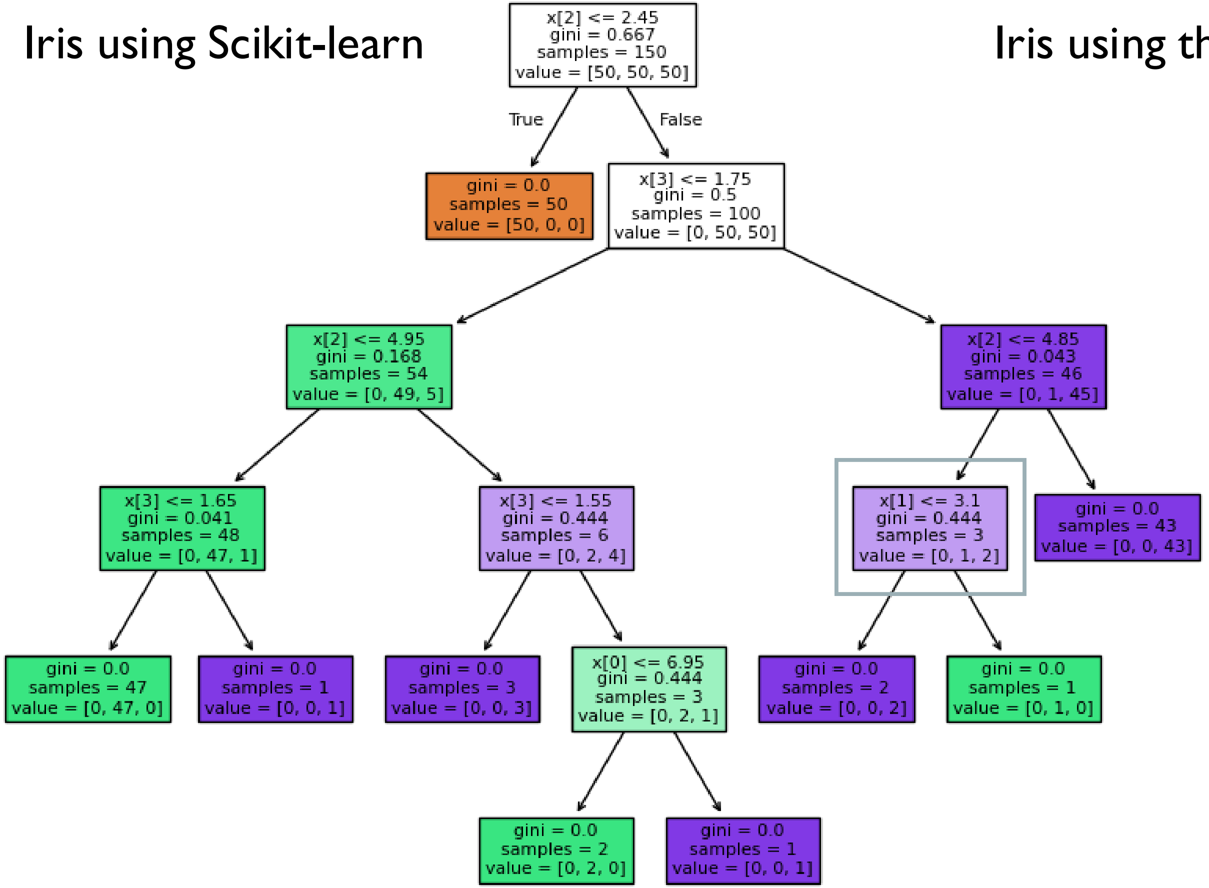


PREVIOUS WORK: IRIS

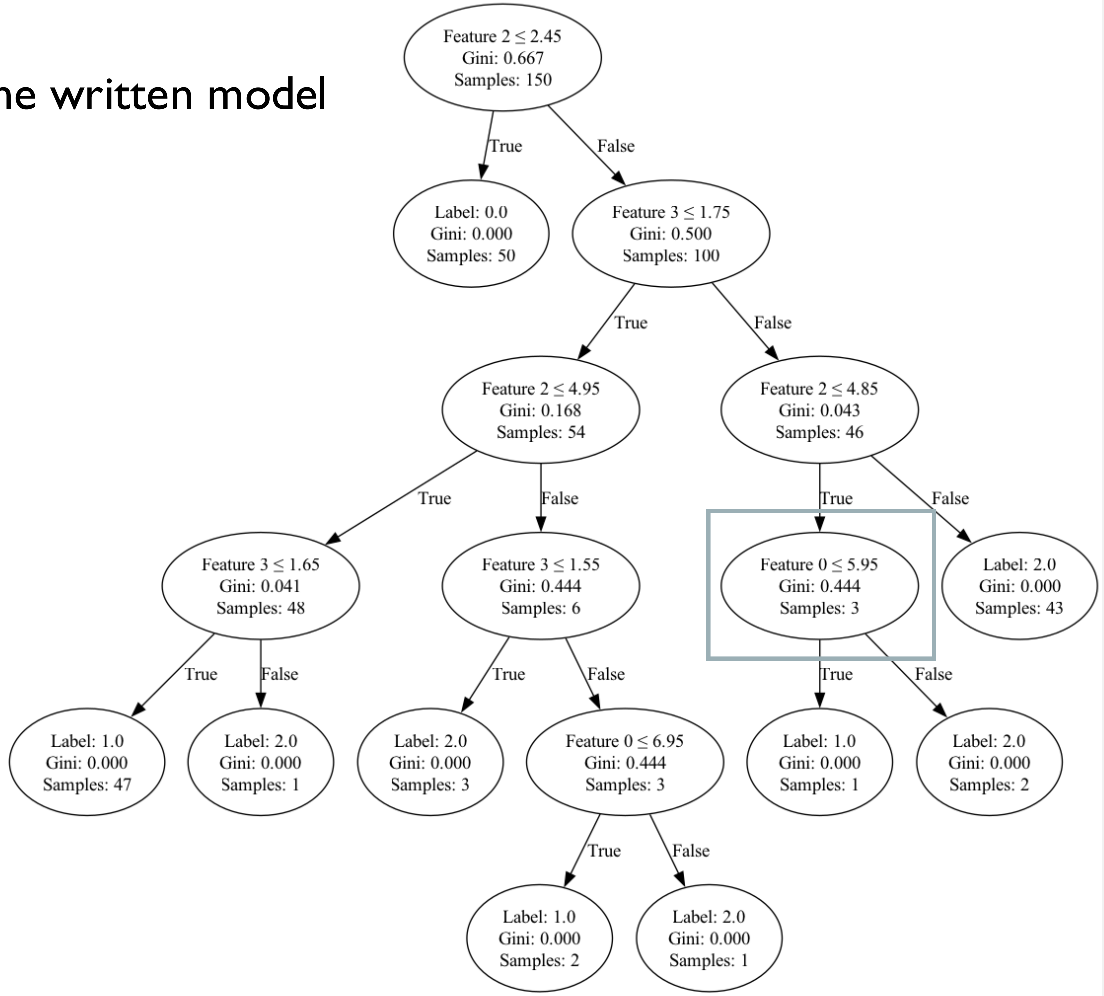
- On the Scikit-learn website, there's an implementation of the IRIS dataset with `DecisionTreeClassifier` [3]
- Our goal is to compare our result to the Scikit-learn's implementation
  - Compare the accuracy
  - Compare the tree structure
- IRIS contains 150 observations and 3 classes

PREVIOUS WORK: IRIS

Iris using Scikit-learn

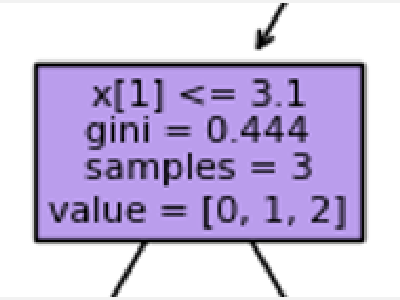


Iris using the written model



PREVIOUS WORK: IRIS

- Issue with the tie (same Gini)
- In Scikit-learn's `DecisionTreeClassifier`, there is a parameter called `splitter` and by default, `splitter = "best"`
- However, when having ties, Scikit-learn will **randomly** choose which feature and threshold to split [4]
- This selection process is random and **hidden** (cannot be controlled by setting the random state)



RESULTS

Dataset	Model	Average training accuracy	Testing accuracy
Iris	Scikit-learn not pruned	0.979998	0.946662
	CART not pruned	0.979998	0.946662
Heart Disease	Scikit-learn not pruned	0.978046	0.959024
	CART not pruned	0.972928	0.943414

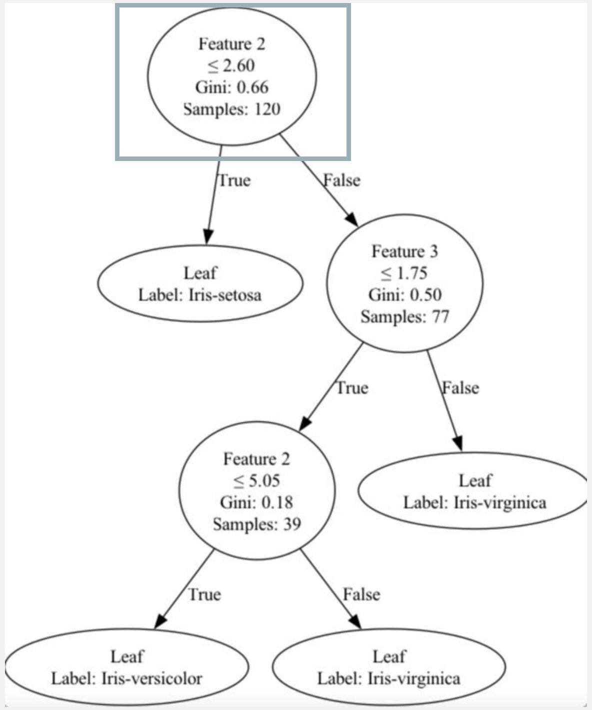
Dataset	Model	Average training accuracy	Testing accuracy
Iris	Scikit-learn pruned	0.9750	0.9666
	CART pruned	0.9733	0.9400
Heart Disease	Scikit-learn pruned	0.8561	0.8634
	CART pruned	0.8629	0.8283

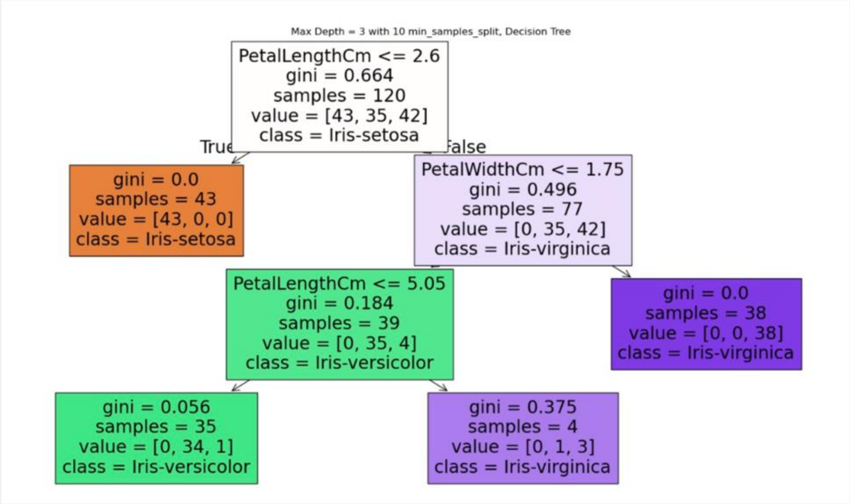
WHY PRUNING MAKES THE TREE LESS ACCURATE?

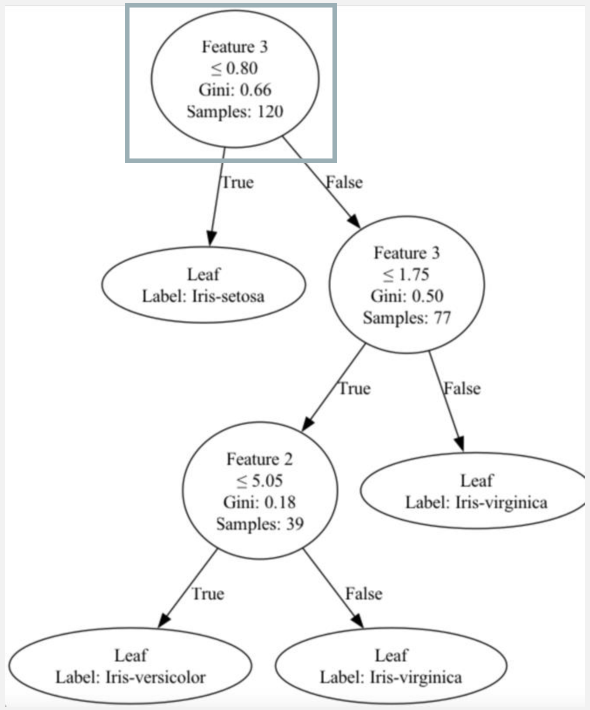
- How pruning affects our result?
  - Pruning reduces the number of features used in the tree
  - The removed features may be important for the testing set
  - Pruning enhances generalization but may fail to capture patterns that could increase the testing accuracy
- Some key aspects of pruning
  - Remove unnecessary branches
  - Produce a less complex tree
- 🤔 Bias-complexity tradeoff

INTERESTING STORY

- Before, we put the entire Iris data into training
- Now, we use train/test = 8/2 splitting







CHALLENGES

- Hard to make everything the same
- The tie-choosing mechanism in Scikit-learn is unknown
- Different random states have different results
- Even if setting the random state, still hard to replicate the exact tree

SUMMARY

- Scikit-learn's `DecisionTreeClassifier` will randomly choose which feature and threshold to split when having ties
- There is no way to fully reproduce `DecisionTreeClassifier` since this mechanism is completely random
- We only considered 2 parameters in `DecisionTreeClassifier` [5]
- The tie problem is crucial, we believe this drags down our model's accuracy since different splitting strategies will lead to misclassifications in the testing set

DecisionTreeClassifier

```
class sklearn.tree.DecisionTreeClassifier(*, criterion='gini',
splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1,
min_weight_fraction_leaf=0.0, max_features=None, random_state=None,
max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None,
ccp_alpha=0.0, monotonic_cst=None)
```

[source]

A decision tree classifier.

REFERENCES

- [1] https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset
- [2] https://scikit-learn.org/1.5/auto_examples/datasets/plot_iris_dataset.html
- [3] https://scikit-learn.org/1.5/auto_examples/tree/plot_iris_dtc.html
- [4] https://github.com/scikit-learn/scikit-learn/issues/12259
- [5] https://scikit-learn.org/1.5/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Q & A

THANK YOU