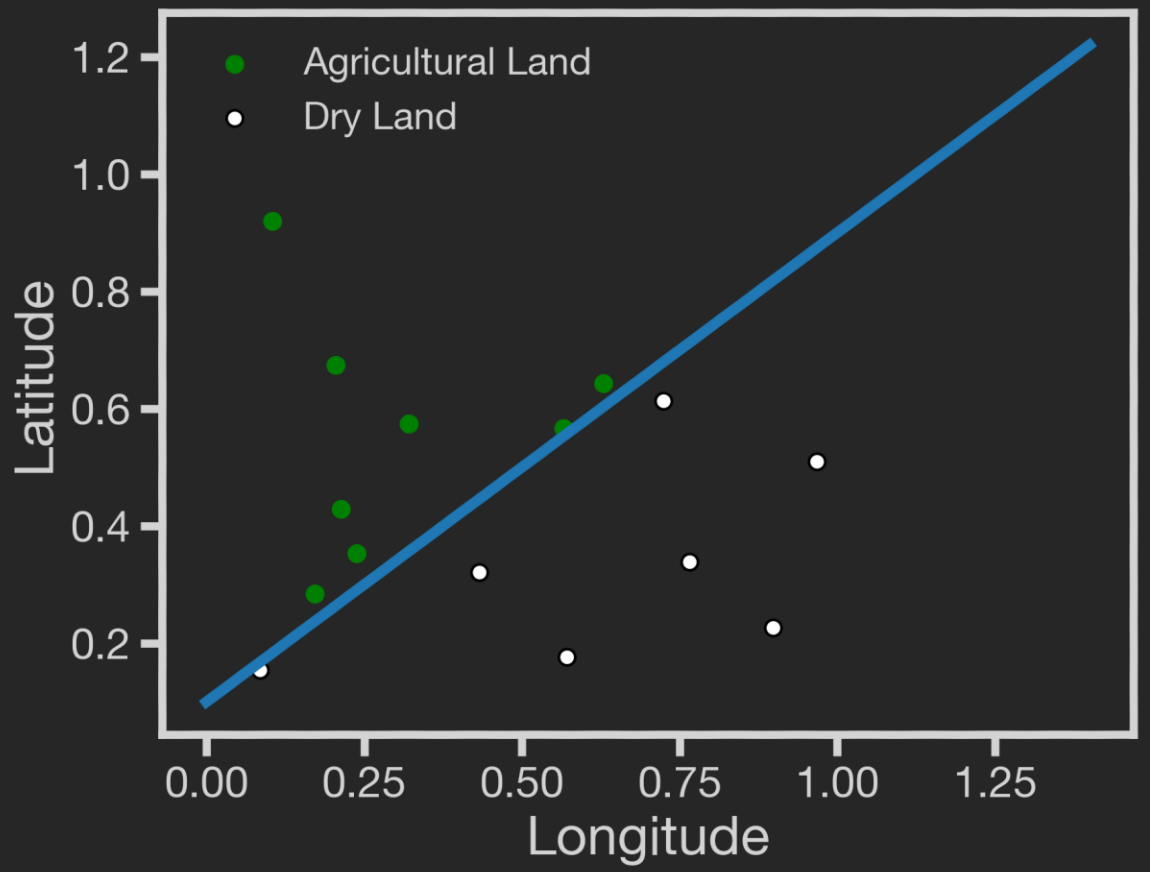
Decision Trees

Motivation

For example, the equation that defines the decision boundary shown with a blue line in the figure is:



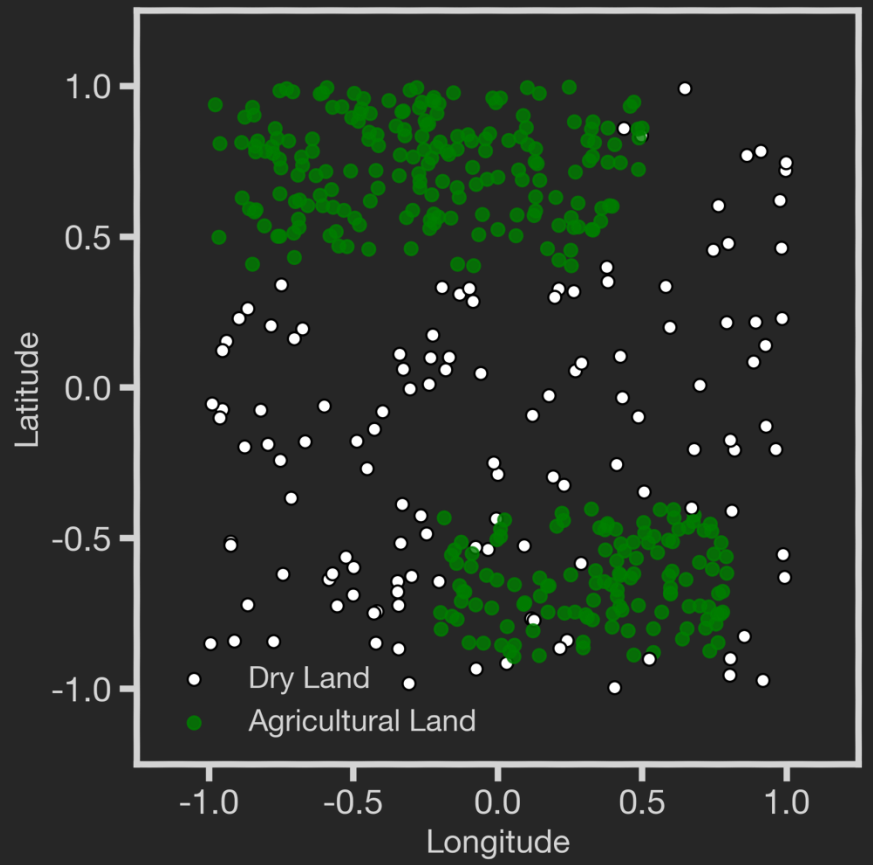
$$Latitude = 0.8\ Longitude + 0.1$$

$$or$$

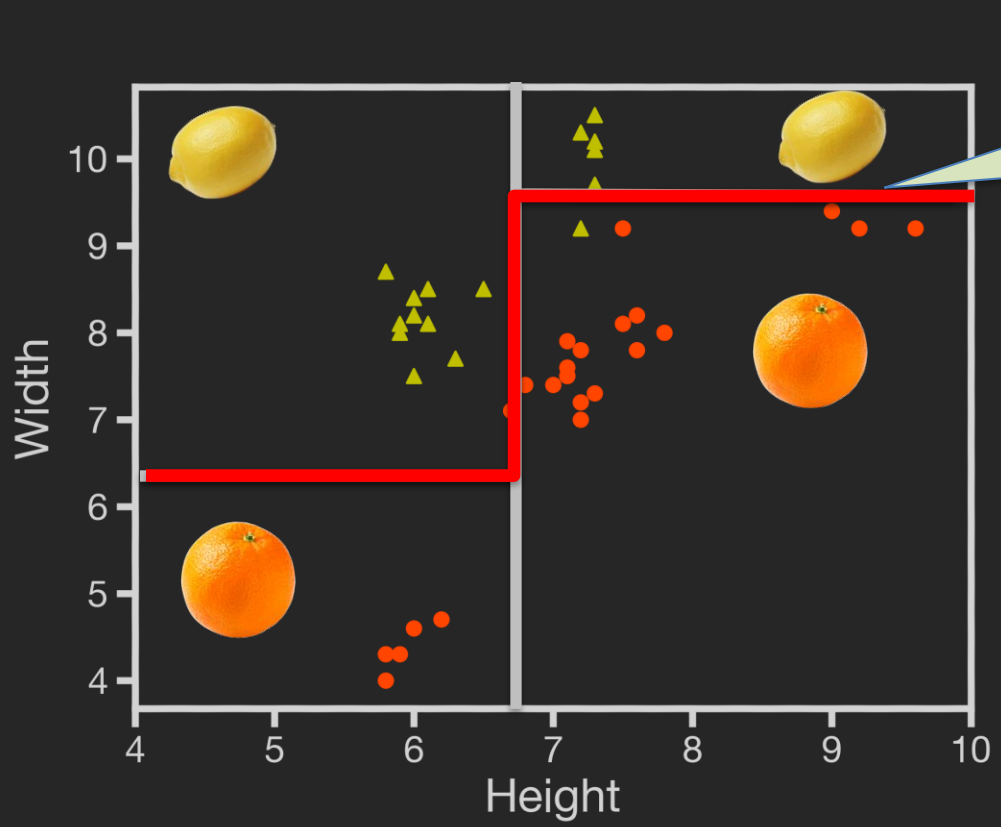
$$-0.8\ Longitude + Latitude - \ 0.1 = 0$$

Motivation

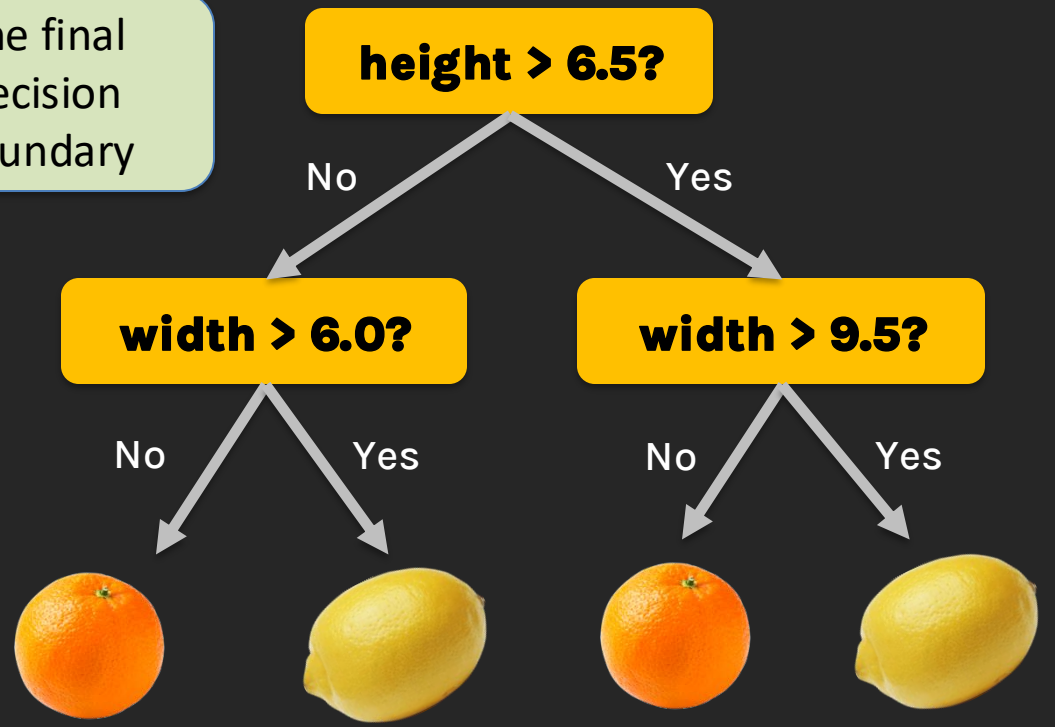
In such datasets the classes are still well-separated in the feature space, but ***the decision boundaries cannot easily be described by a single equation***.



Example

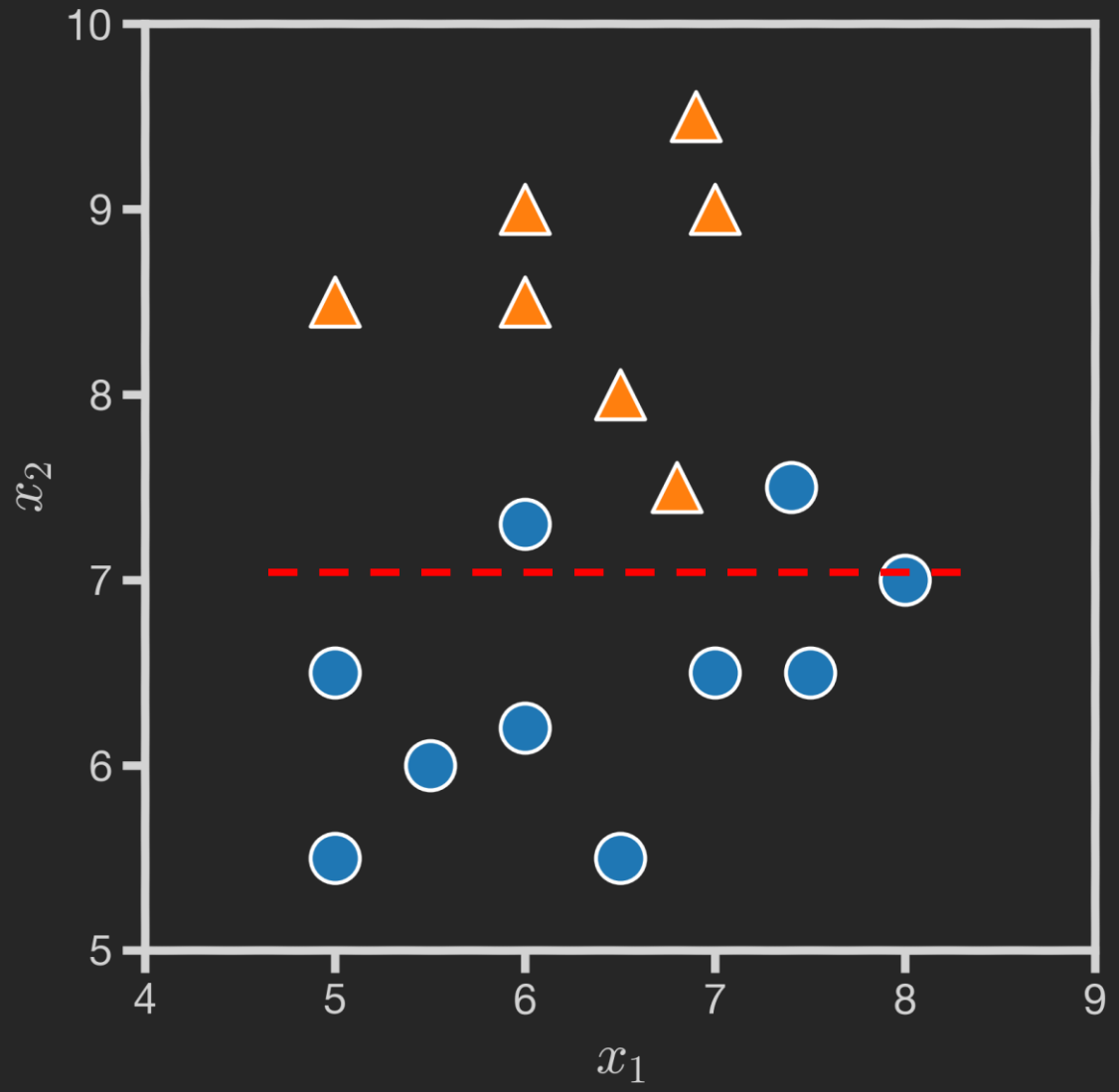


The final decision boundary



Splitting Criteria

Splitting Criteria



Which gives a better split?

- Should it be split along x_1 or x_2?
- Where should we split?

Classification Error

In general:

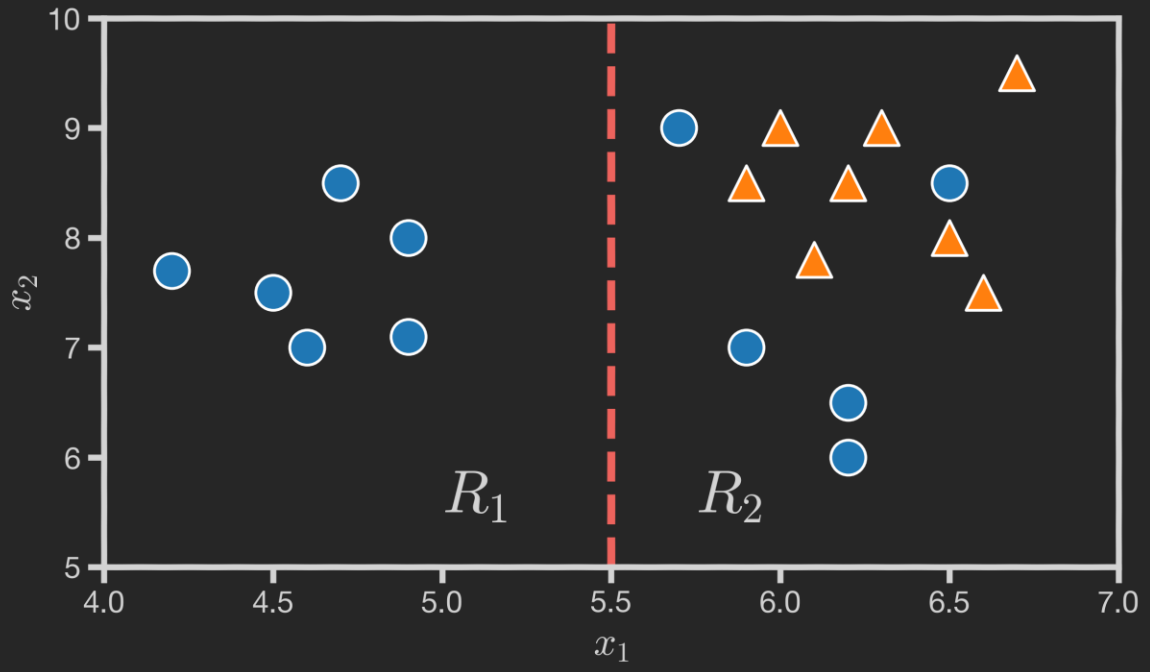
Assume we have P **predictors (or features)** and K **classes**. Suppose we select the p^{th} predictor and split a region along the **threshold** t_p.

We can assess the quality of this split by calculating the **classification error** made by each newly created region:

$$Error\ (R_r|\ p, t_p) = 1\ - \max_k(\Psi(k|R_r))$$

where $\Psi(k|R_r)$ is the proportion of training points in R_r that are labeled class k.

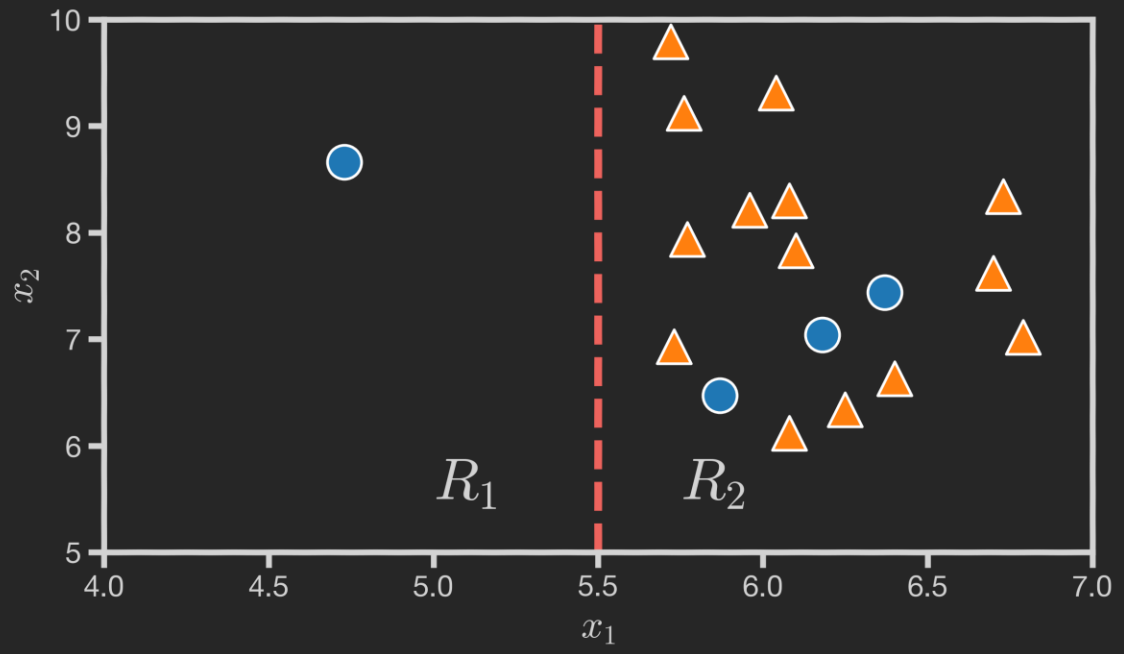
Classification Error



	●	▲	$Error = 1 - \max_k(\Psi(k\|R_r))$
R_1	6	0	$1 - max\left(\frac{6}{6}, \frac{0}{6}\right) = 0$
R_2	5	8	$1 - max\left(\frac{5}{13}, \frac{8}{13}\right) = \frac{5}{13} = 0.38$

Classification Error

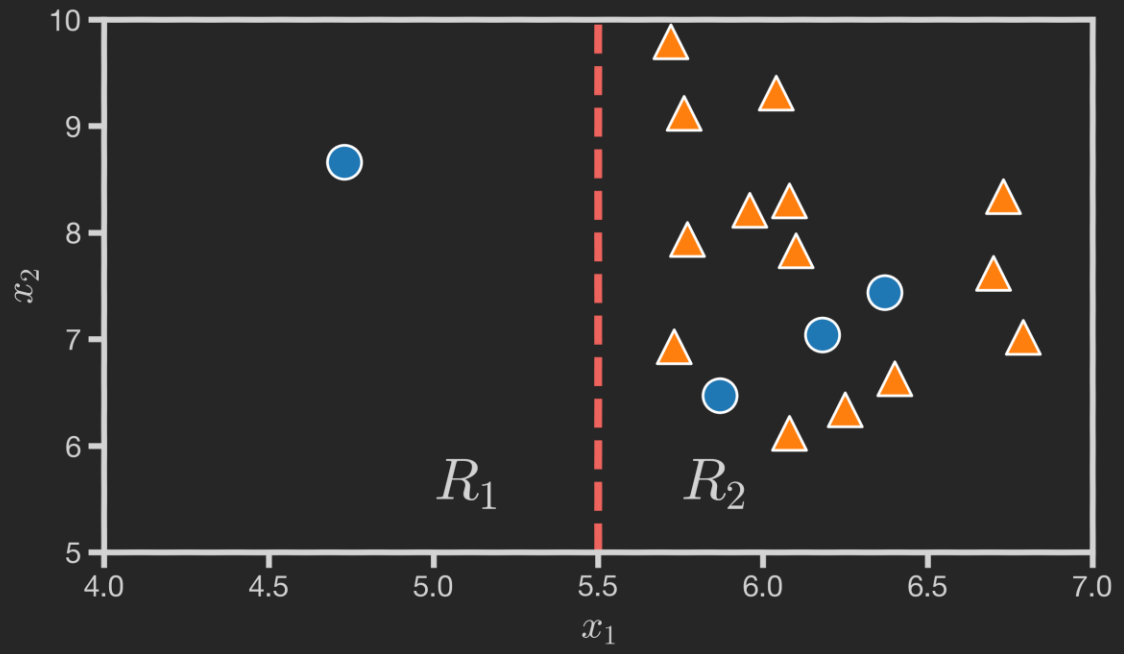
Now calculate the error for the following split:



	●	▲	$Error = 1 - \max_k(\Psi(k\|R_r))$
R_1	1	0	$1 - max\left(\frac{1}{1}, \frac{0}{1}\right) = 0$
R_2	3	14	$1 - max\left(\frac{3}{17}, \frac{14}{17}\right) = \frac{3}{17} = 0.18$

R_1 has a smaller error that R_2.
Does that mean this is a good split?

Classification Error



We need to take the **weighted average** over both regions so the number of points in each region is taken into consideration:

$$\min_{p,\, t_p} \left[\frac{N_1}{N} Error\left(R_1 \middle| p\,, t_p\right) + \frac{N_2}{N} Error\left(R_2 \middle| p\,, t_p\right) \right]$$

where N_r is the number of training points inside region R_r.

Gini Index

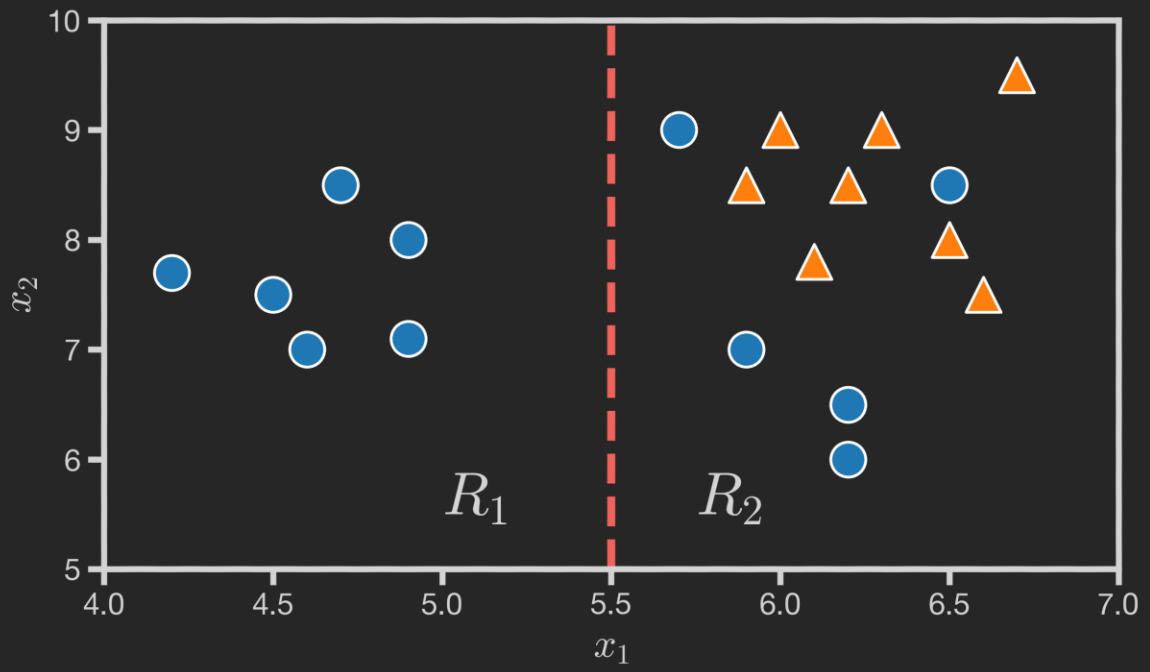
Assume we have P predictors and K classes. Suppose we select the p^{th} predictor and split a region along the threshold t_p.

We can assess the quality of this split by measuring the Gini Index made by each newly created region by calculating:

1 – sum of the squares of the proportions of points from the k-th class in the region r (PSI).

$$Gini\ (R_r | \ p, t_p) = 1\ - \sum_k \Psi(k|R_r)^2$$

Gini Index



	●	▲	$Gini = 1 - \sum_k \Psi(k\|R_r)^2$
R_1	6	0	$1 - \left[\left(\frac{6}{6}\right)^2 + \left(\frac{0}{6}\right)^2\right] = 0$
R_2	5	8	$1 - \left[\left(\frac{5}{13}\right)^2 + \left(\frac{8}{13}\right)^2\right] = \frac{80}{169} = \mathbf{0.47}$

Gini Index

We can now try to find the predictor p and the threshold t_p that minimize the **weighted average Gini Index** over the two regions:

$$\min_{p,t_p}\left[\frac{N_1}{N}Gini\left(R_1|p\,,t_p\right)+\frac{N_2}{N}Gini\left(R_2|p\,,t_p\right)\right]$$

where $N_{1,2}$ is the number of training points inside region $R_{1,2}$.

Entropy

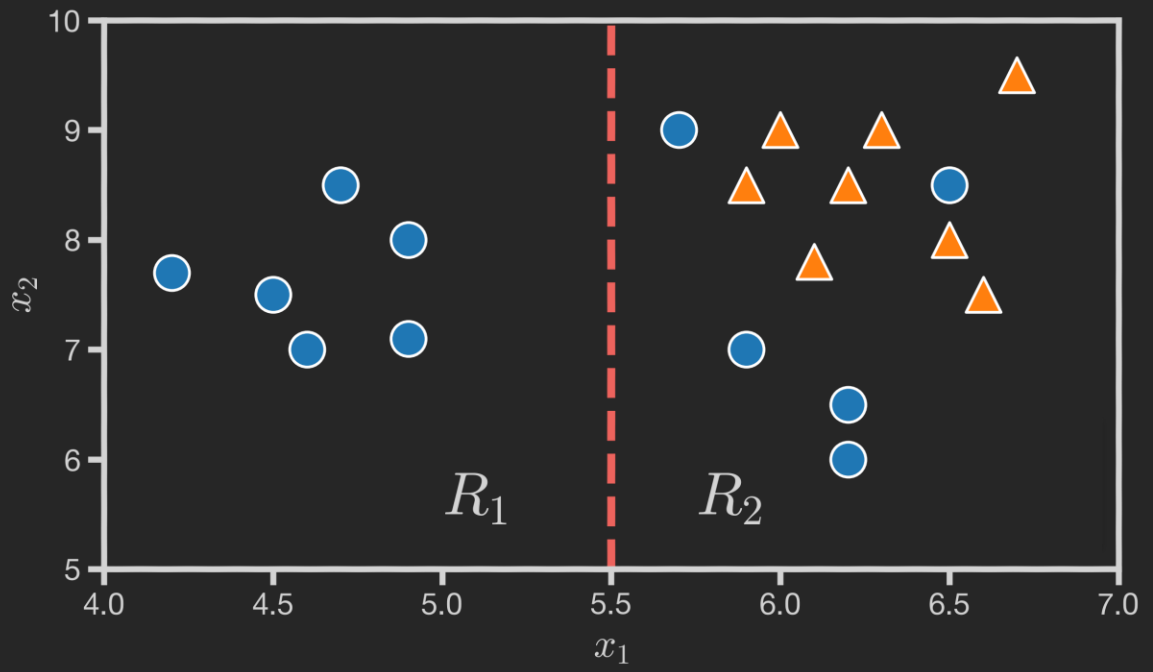
Assume we have P predictors and K classes. Suppose we select the p^{th} predictor and split a region along the **threshold** t_p.

We can assess the quality of this split by measuring the entropy of the class distribution in each newly created region by calculating:

$$Entropy\,(R_r|\,p,\,t_p) = -\sum_k \Psi(k|R_r)\log_2\Psi(k|R_r)$$

Note: We are actually computing the conditional entropy of the distribution of training points amongst the K classes given that the point is in region r.

Entropy



	●	▲	$Entropy = -\sum_k \Psi(k\|R_r)\log_2 \Psi(k\|R_r)$
R_1	6	0	$-\left[\frac{6}{6}\log_2\frac{6}{6}+\frac{0}{6}\log_2\frac{0}{6}\right]=0$
R_2	5	8	$-\left[\frac{5}{13}\log_2\frac{5}{13}+\frac{8}{13}\log_2\frac{8}{13}\right]\approx 1.38$

Entropy

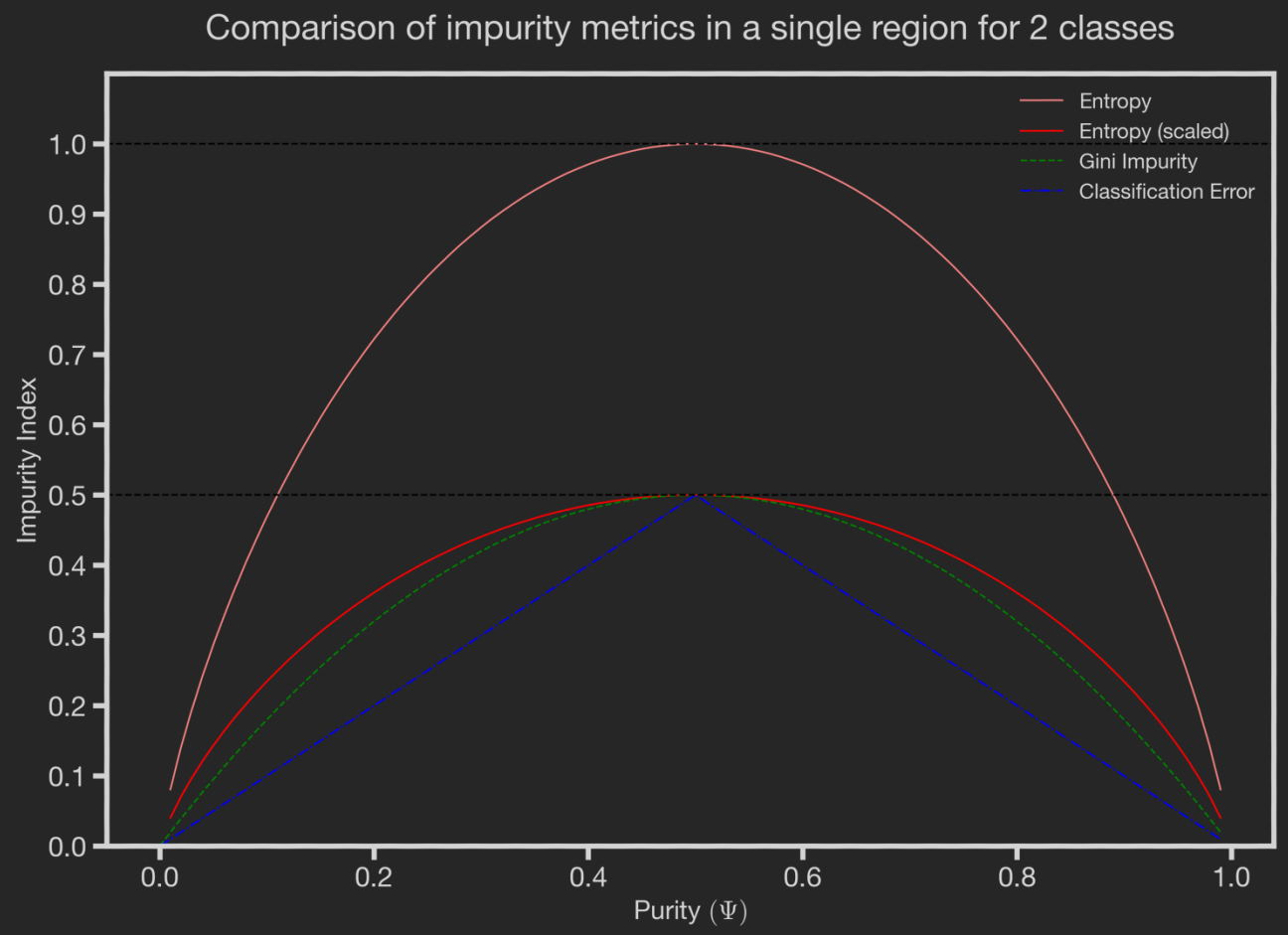
We can now try to find the predictor p and the threshold t_p that minimize the **weighted average entropy** over the two regions:

$$\min_{p,t_p}\left[\frac{N_1}{N}Entropy(R_1|p\,,t_p)+\frac{N_2}{N}Entropy(R_2|p\,,t_p)\right]$$

where $N_{1,2}$ is the number of training points inside region $R_{1,2}$.

Comparison of Criteria

Question: Which of the three criteria fits our guideline the best?

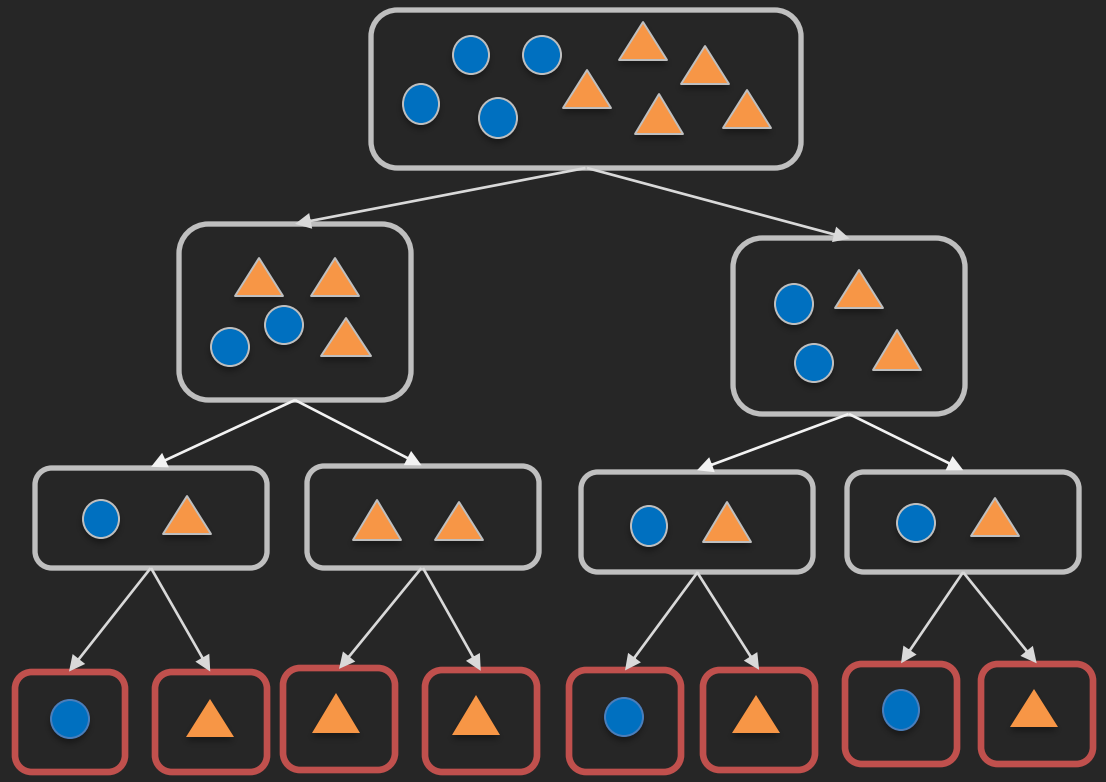


Entropy **penalizes impurity the most**

Stopping Conditions

Stopping Conditions

Question: If we don't terminate the decision tree algorithm manually, what will the leaf nodes of the decision tree look like?

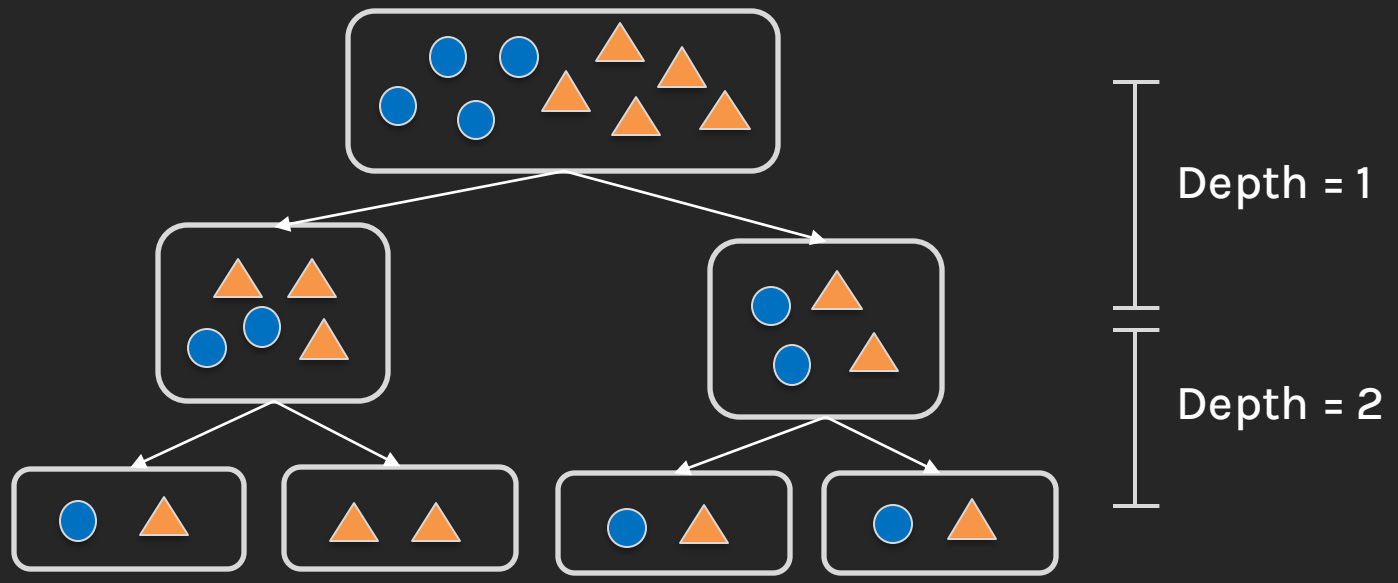


The tree will continue to grow until each region contains **exactly one training point** and the model attains 100% **training** accuracy.

Stopping Conditions

The most common stopping condition is to limit the maximum depth *(max_depth)* of the tree.

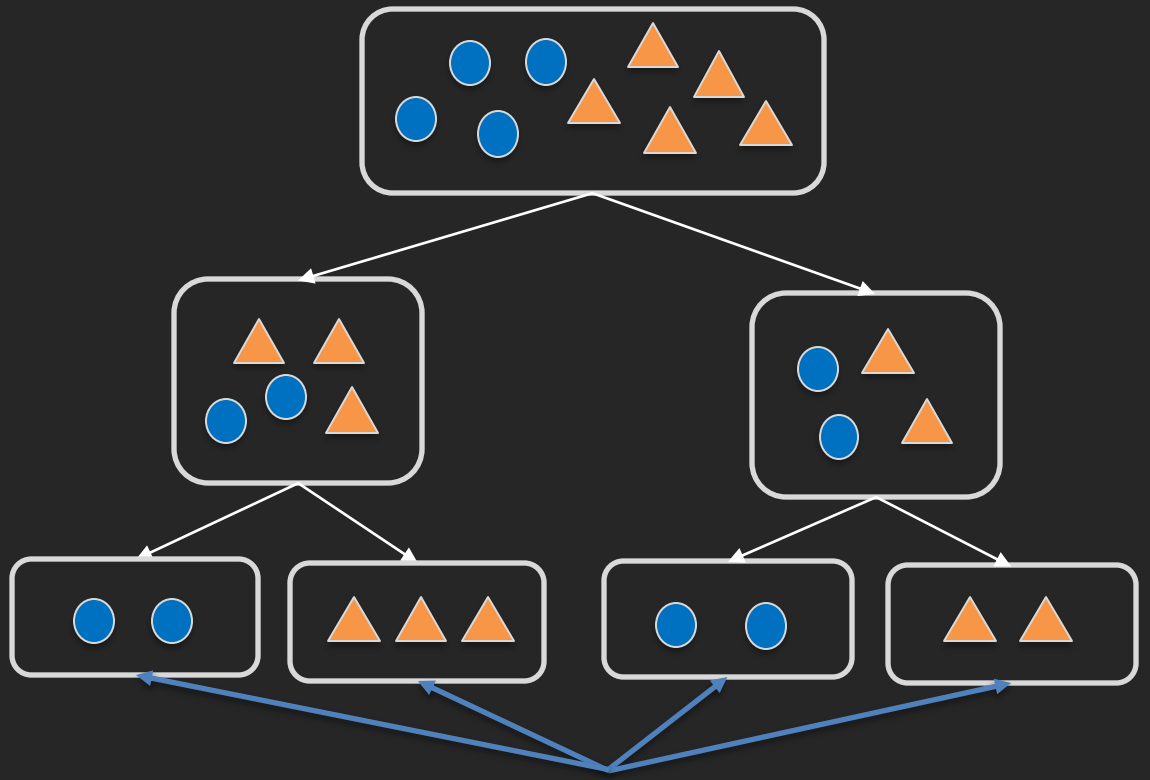
max_depth = 2



Stopping Conditions

Other common simple stopping conditions are:

- Don’t split a region if all instances in the region belong to the same class.

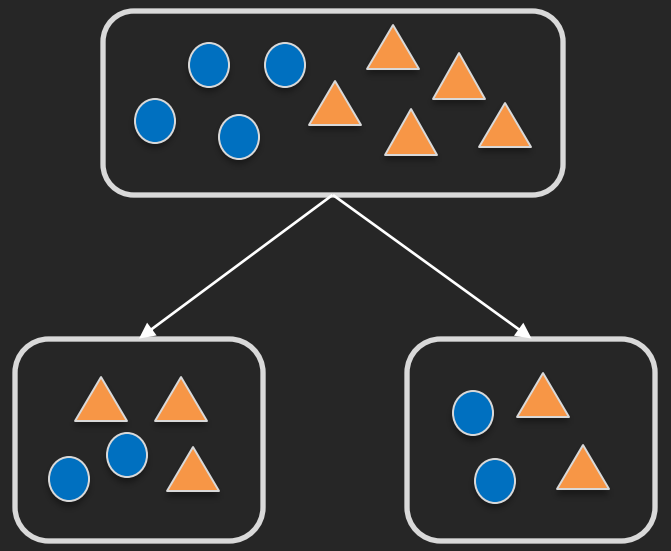


Stopping Conditions

Other common simple stopping conditions are:

- Don't split a region if the number of instances in any of the sub-regions will fall below pre-defined threshold *(min_samples_leaf)*.

min_samples_leaf = 4



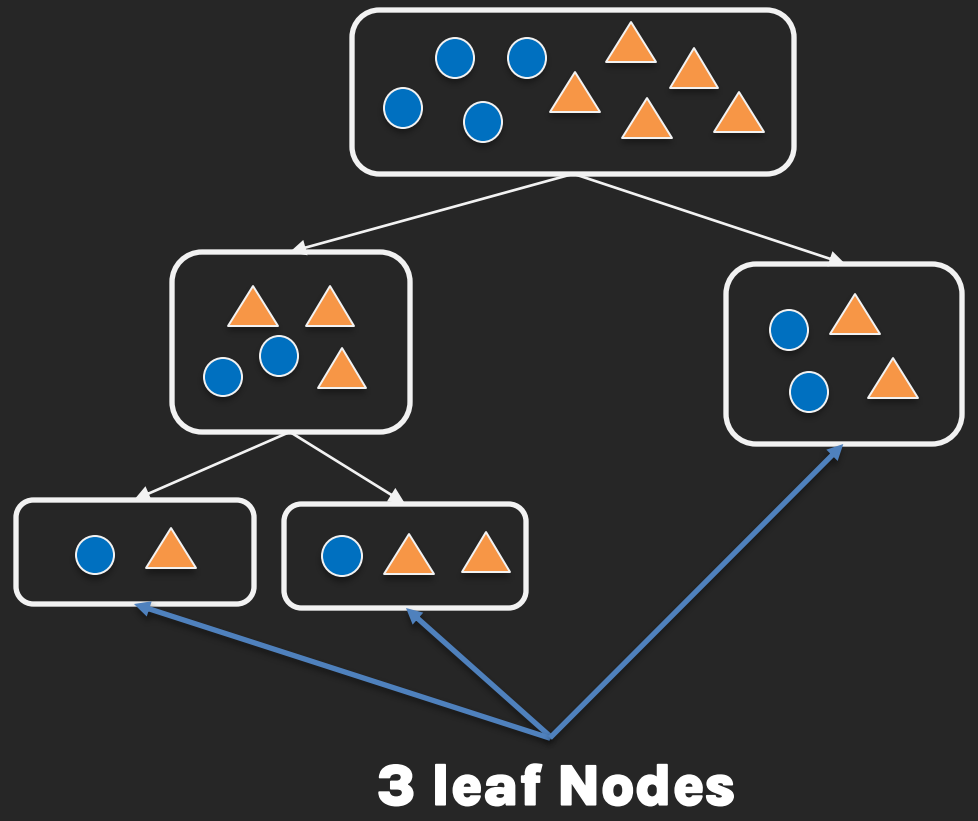
Stopping Conditions

Other common simple stopping conditions are:

- Don't split a region if the total number of leaves in the tree will exceed a pre-defined threshold *(max_leaf_nodes)*.

max_leaf_nodes = 3

Do you see any issue with this?



Stopping Conditions

Normally, Sklearn grows trees in what is called **'level-order'**-fashion until a stopping condition such as *max_depth* is met.

However, if a value for *max_leaf_nodes* is specified, Sklearn will instead grow the tree in a '**best-first**' fashion.

But what do level-order and best-first growth ***mean***?

Stopping Conditions

Normally, Sklearn grows trees in what is called **'level-order'**-fashion until a stopping condition such as *max_depth* is met.

However, if a value for *max_leaf_nodes* is specified, Sklearn will instead grow the tree in a '**best-first**' fashion.

But what do level-order and best-first growth ***mean***?

Level-order is also sometimes called '*breadth*-first'-search but we will use the term 'level-order' so it won't be confused with the similar sounding '*best*-first' search.

Stopping Conditions

A more restrictive stopping condition is:

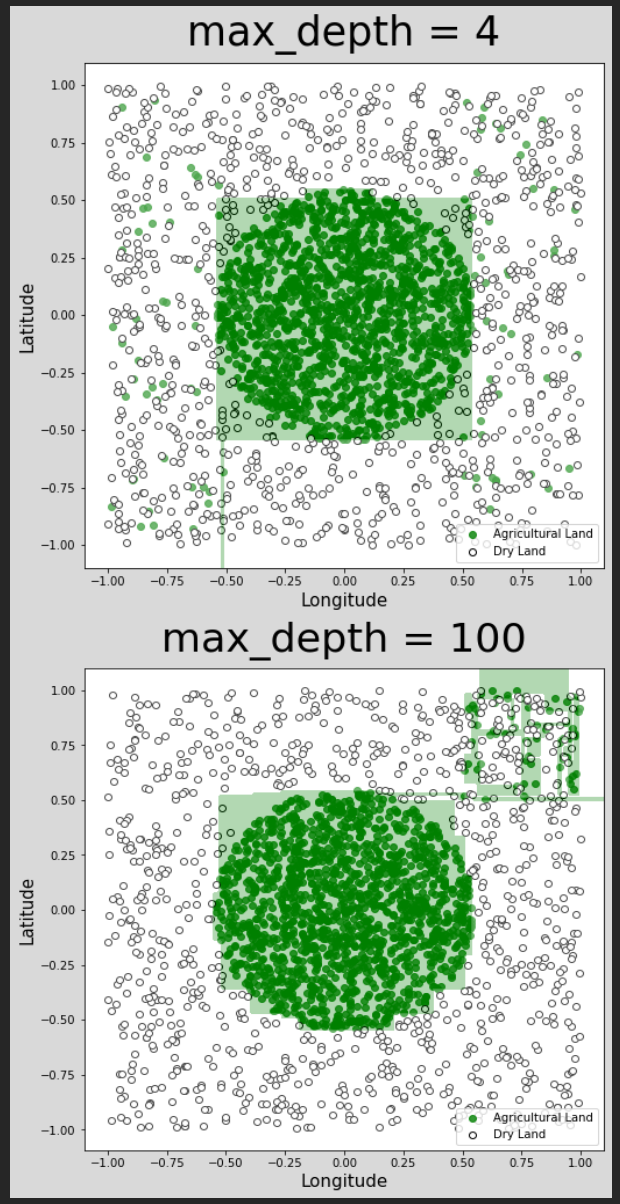
Compute the gain in purity of splitting a region R into R_1 and R_2:

$$Gain(R) = \Delta(R) = m(R) - \frac{N_1}{N} m(R_1) - \frac{N_2}{N} m(R_2)$$

Classification Error/Gini Index/Entropy

Don't split if the gain is less than some pre-defined threshold (`min_impurity_decrease`).

Variance vs Bias



- **High Bias:** Trees of low depth are not a good fit for the training data - it's unable to capture the nonlinear boundary separating the two classes.
- **Low Variance:** Trees of low depth are robust to slight perturbations in the training data - the square carved out by the model is stable if you move the boundary points a bit.
- **Low Bias:** With a high depth, we can obtain a model that correctly classifies all points on the boundary (by zig-zagging around each point).
- **High Variance:** Trees of high depth are sensitive to perturbations in the training data, especially to changes in the boundary points.

How do we decide what is the appropriate stopping condition or stopping method?

Stopping Conditions

max_depth | *min_samples_leaf* | *max_leaf_nodes* | min_impurity_decrease

How can we determine the appropriate hyperparameters?

cross-validation

Regression Trees

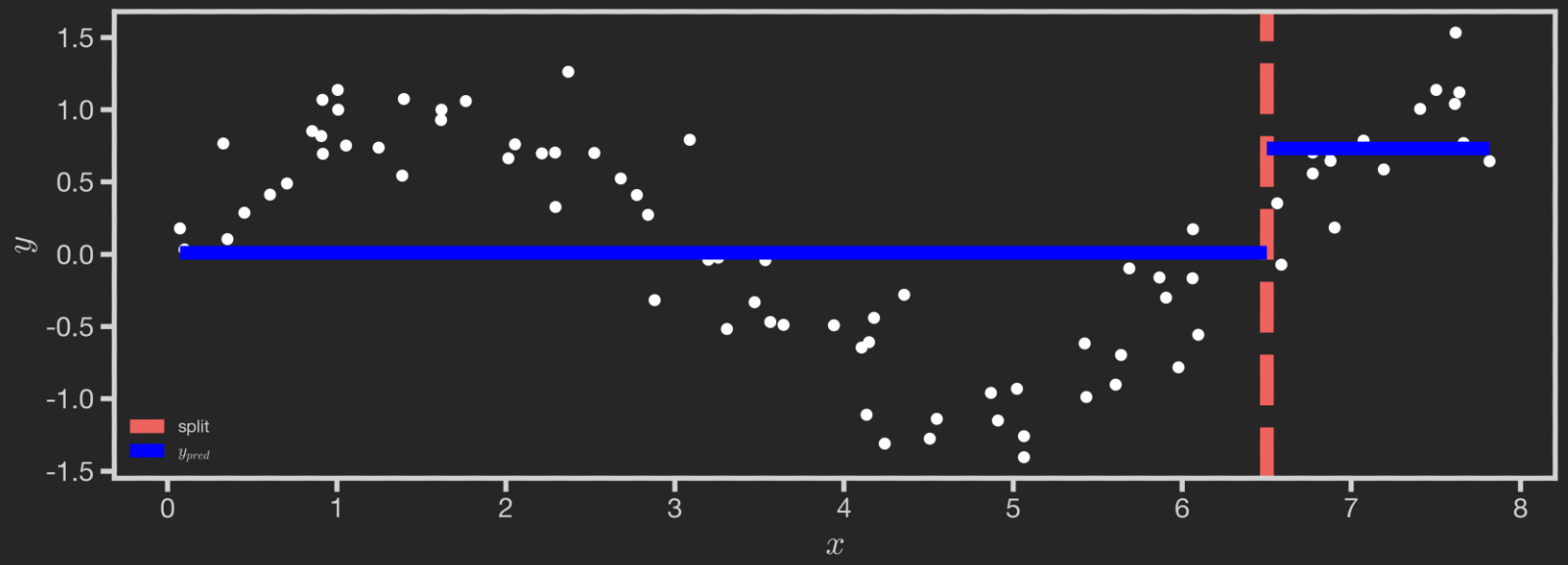
Regression Trees

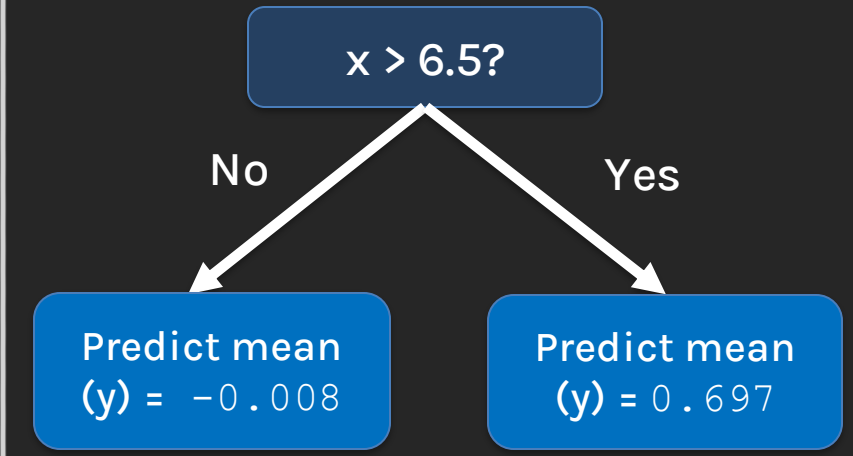
How can this decision tree approach apply to a *regression problem* (quantitative outcome)?

Questions to consider:

- How would you determine any **splitting criteria**?
- What would be a reasonable **objective function**?
- How would you perform **prediction at each leaf**?

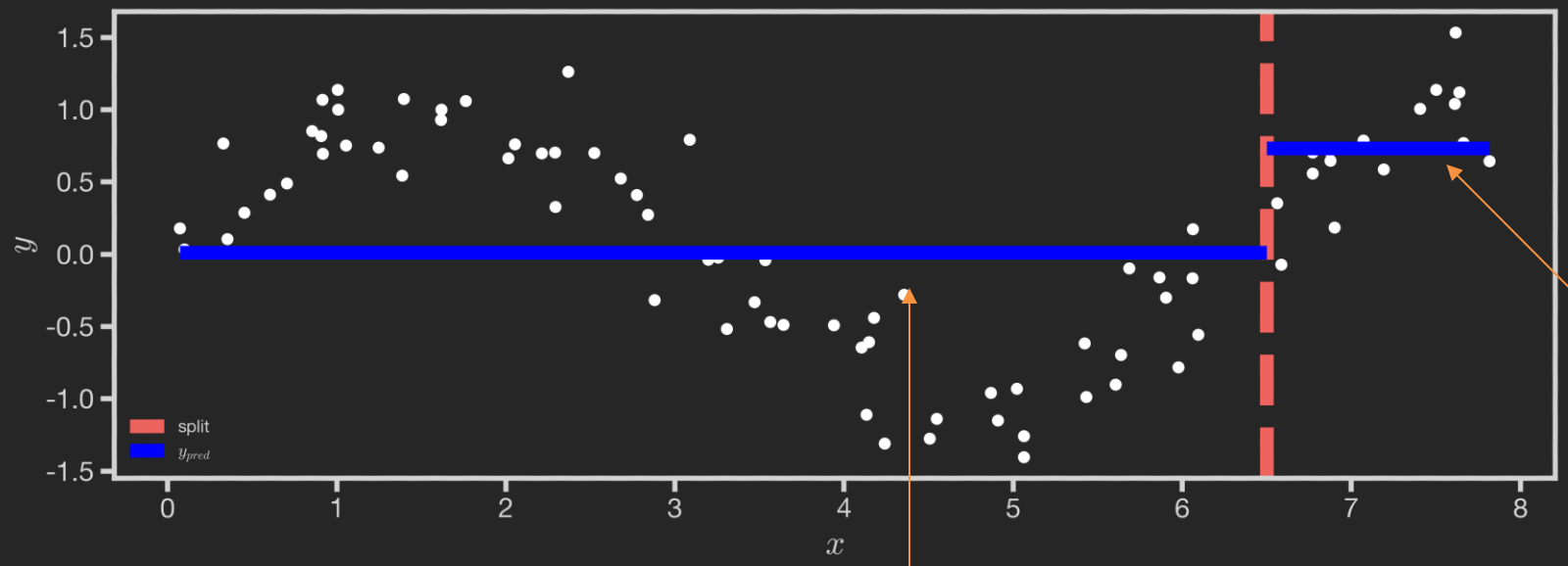
Splitting Criteria





Splitting Criteria

We can assess the quality of this split by calculating the mean squared error for each newly created region:



$$MSE(R_1) = \frac{1}{n_1} \sum_{i \in R_1} (y_i - \bar{y}_{R_1})^2$$

$$MSE(R_2) = \frac{1}{n_2} \sum_{i \in R_2} (y_i - \bar{y}_{R_2})^2$$

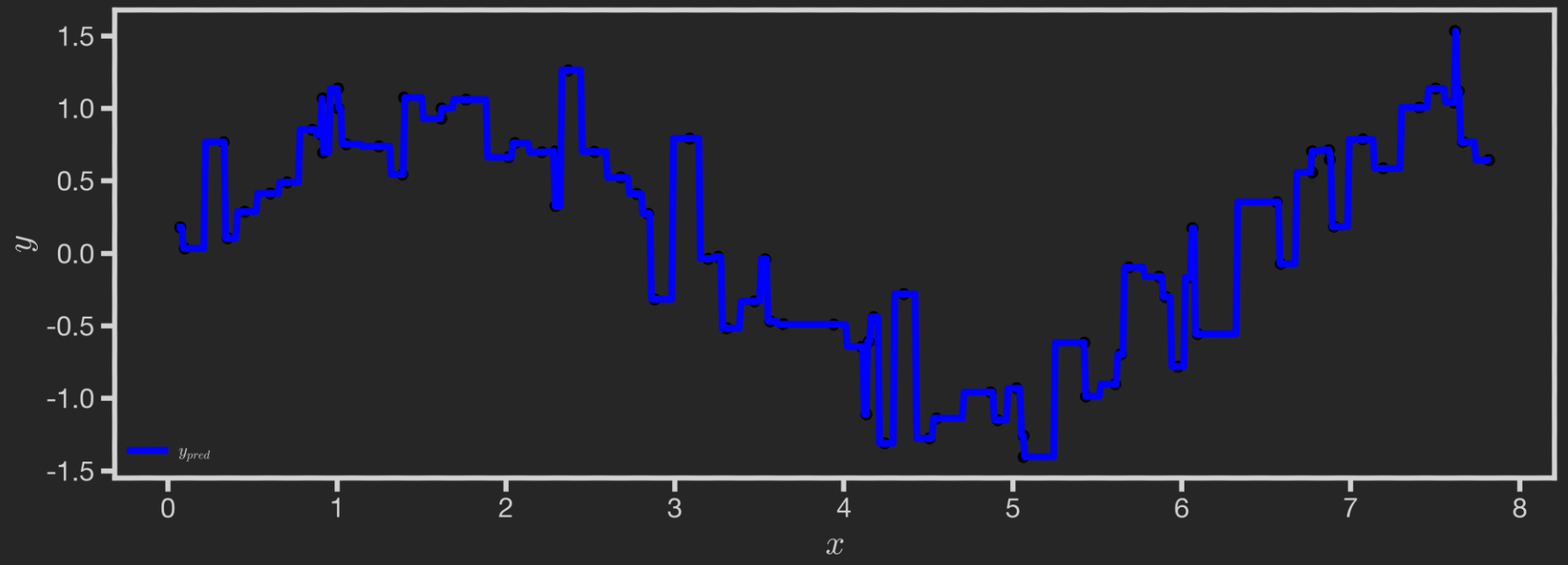
Splitting Criteria

When calculating the MSE, we need to consider the number of points in each region. So, we take the **weighted** average over both regions.

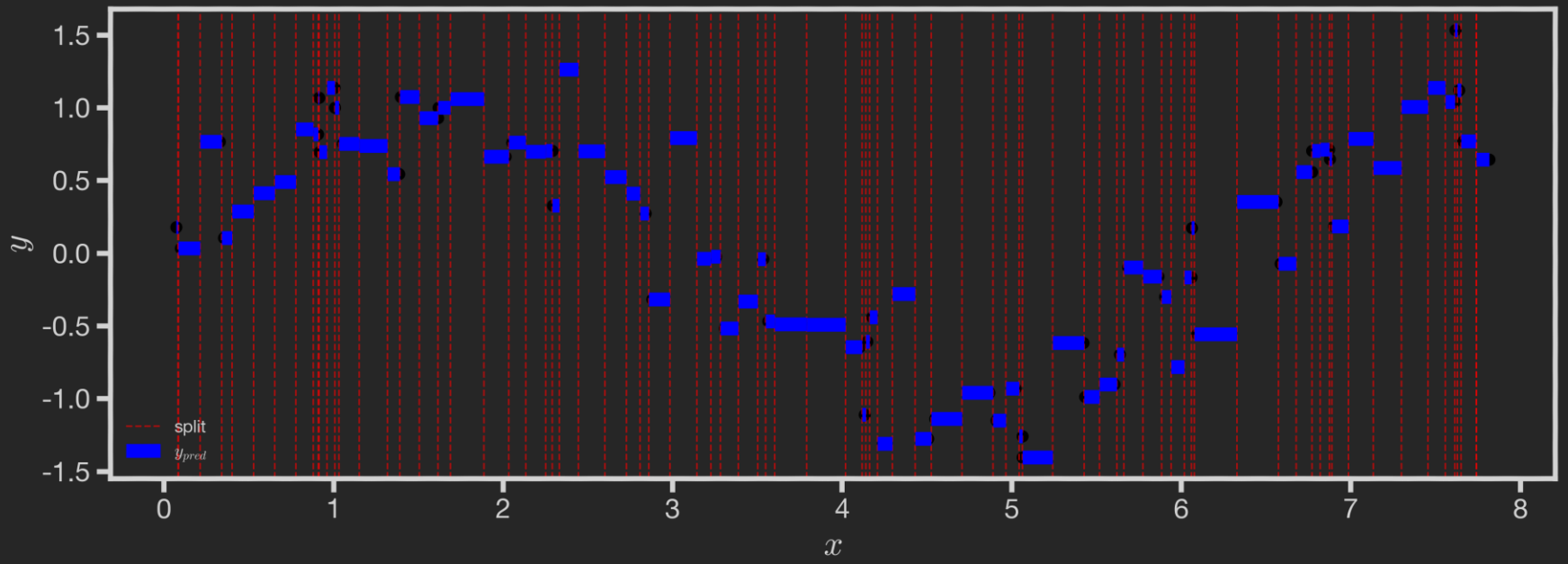
$$\min_{p,\, t_p} \left[\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2) \right]$$

Reminder: p denotes the predictor, and t_p denotes the threshold.

Regression Tree (max_depth = 5)



Regression Tree (max_depth = 10)



Stopping Conditions

Most of the stopping conditions we saw for classification trees, such as **maximum depth** or **minimum number** of points in a region, can also be applied to regression.

Instead of purity gain, we can compute **accuracy gain** **(in this case MSE reduction)** for splitting a region R and stop the tree when the gain is less than some pre-defined threshold.

$$Gain(R) = \Delta(R) = \underbrace{MSE(R)}_{\text{MSE without split}} - \underbrace{\left(\frac{N_1}{N} MSE(R_1) + \frac{N_2}{N} MSE(R_2)\right)}_{\text{MSE after split}}$$

Categorical Attributes

Numerical vs Categorical Attributes

Consider the following data:

Sepal width	Color	Flower
3.0 mm	Yellow	Sunflower
3.5 mm	Red	Rose
3.7 mm	Purple	Tulip
4.5 mm	Purple	Orchid

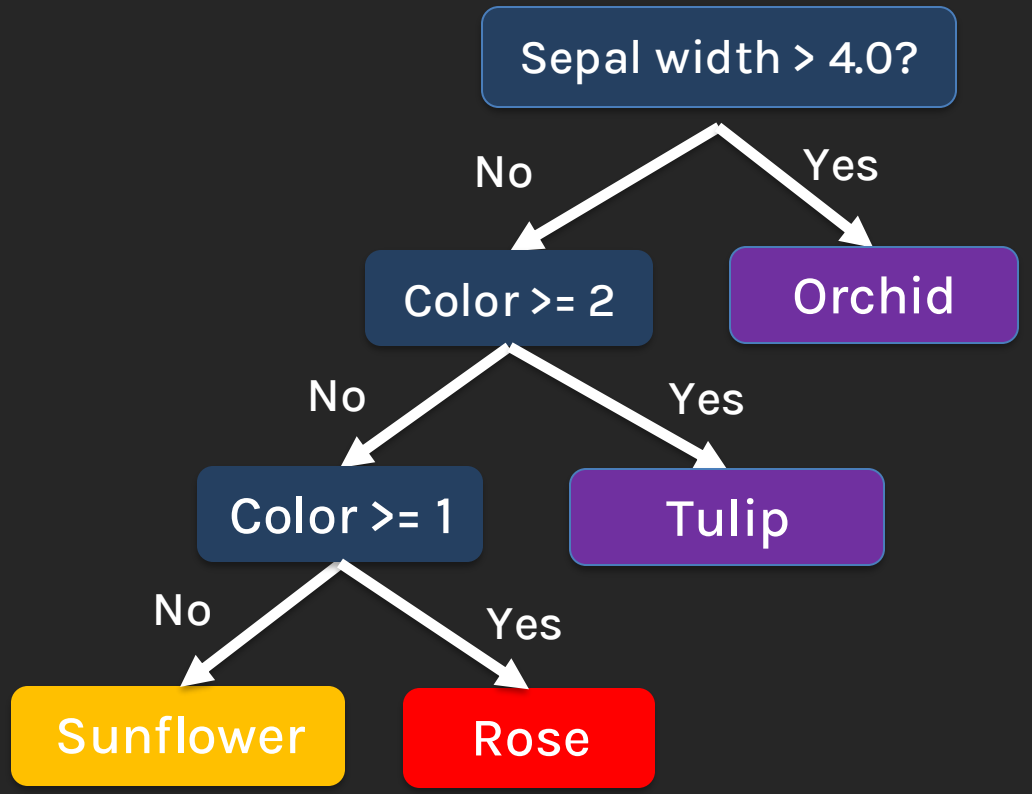
Question: How do we construct a decision tree for this data?

Numerical vs Categorical Attributes

A simple solution is to encode the values of a categorical feature using numbers and treat this feature like a numerical variable.

If we encode **Yellow** = 0, **Red** = 1, **Purple** = 2, our decision tree can be:

Sepal width	Color	Flower
3.0 mm	0	Sunflower
3.5 mm	1	Rose
3.7 mm	2	Tulip
4.5 mm	2	Orchid



Numerical vs Categorical Attributes

In the example, we used **ordinal encoding.** If your categorical data is not ordinal, this is not good. You'll end up with splits that do not make sense. How do we encode this?

One-hot-encoding or dummy encoding!

Numerical vs Categorical Attributes

Sepal width	Color	Flower
3.0 mm	Yellow	Sunflower
3.5 mm	Red	Rose
3.7 mm	Purple	Tulip
4.5 mm	Purple	Orchid

OHE →

Sepal width	C1	C2	C3	Flower
3.0 mm	0	0	1	Sunflower
3.5 mm	0	1	0	Rose
3.7 mm	1	0	0	Tulip
4.5 mm	1	0	0	Orchid

Pruning

Alternatives to Using Stopping Conditions

What is the major issue with pre-specifying a stopping condition?

- You may stop too early or stop too late.

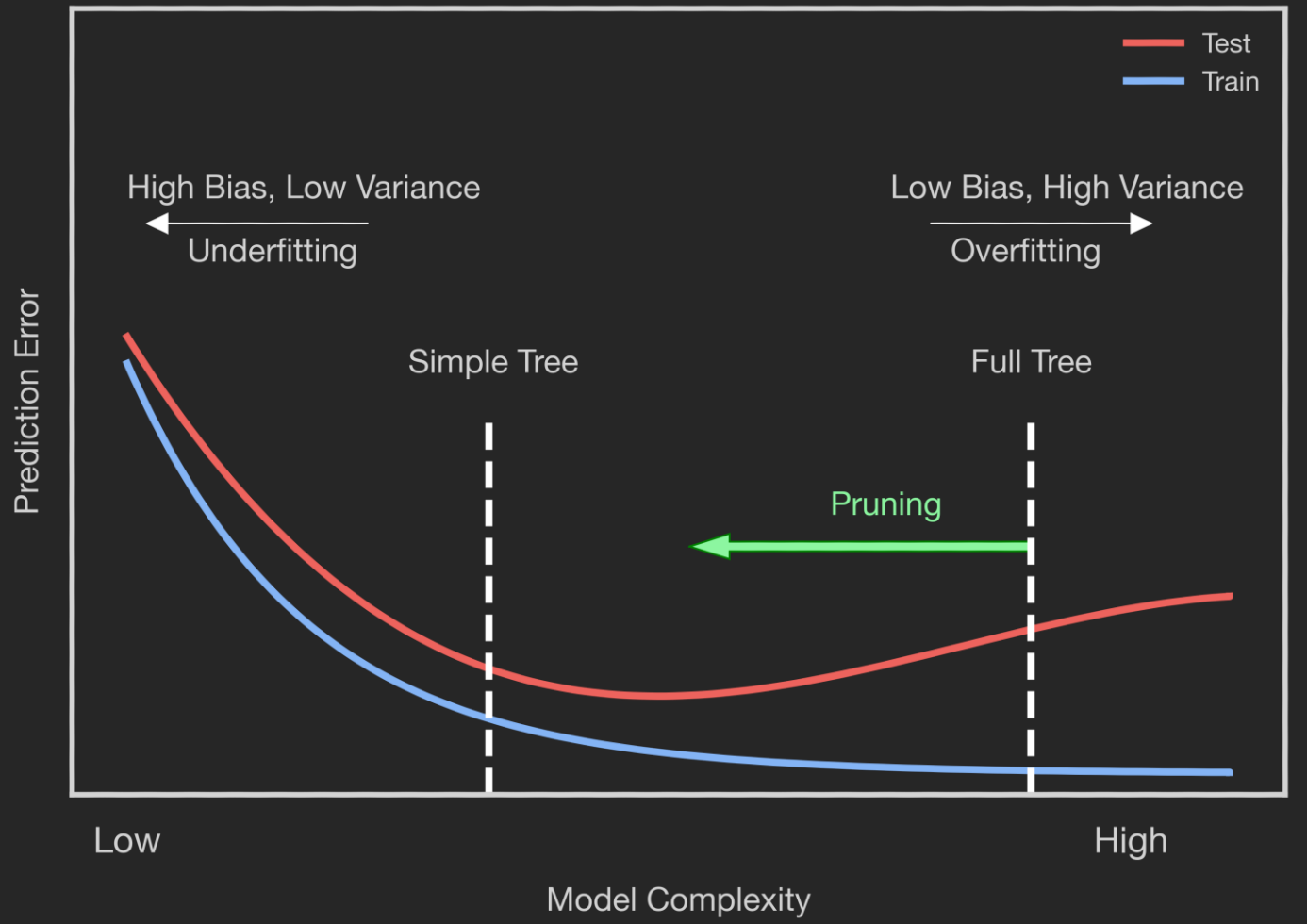
How can we fix this issue?

- Choose several stopping criteria (e.g., set minimal Gain(R) at various levels) and cross-validate to decide which one is the best.

What is an alternative approach to this method?

- Don’t stop. Instead, prune the tree!

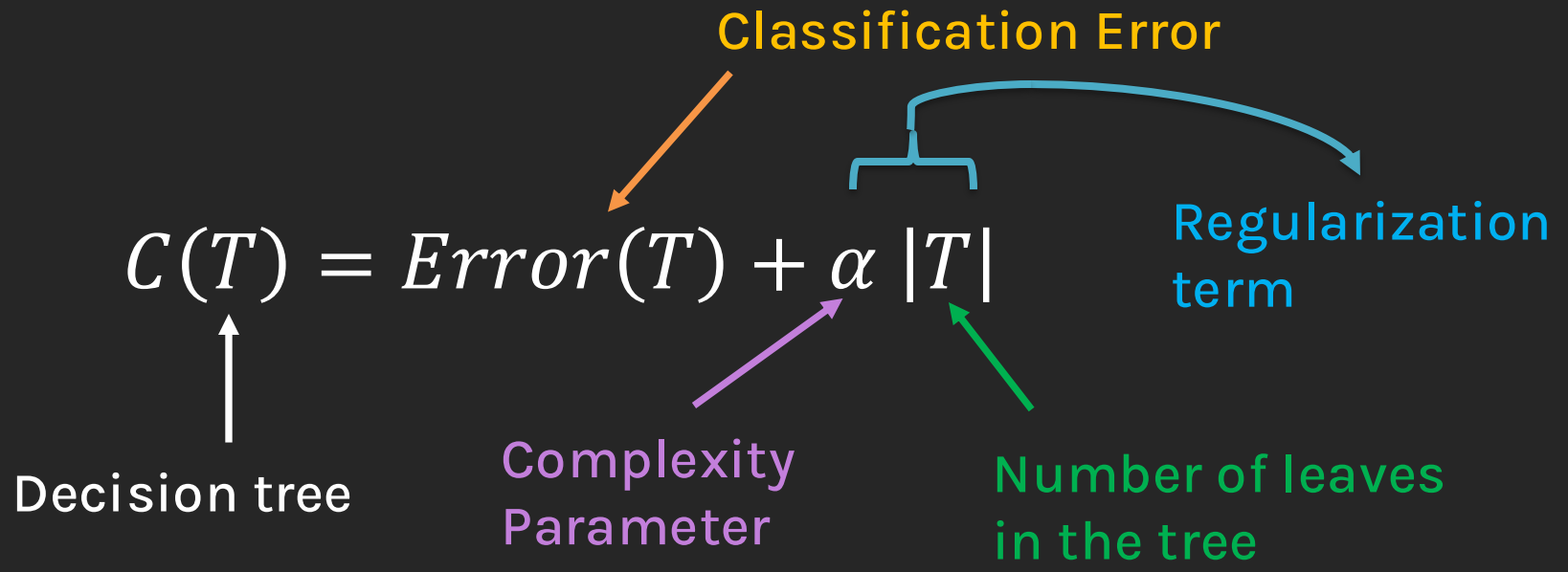
Motivation for Pruning



Rather than preventing a complex tree from growing, we can obtain a simpler tree by 'pruning' a complex one.

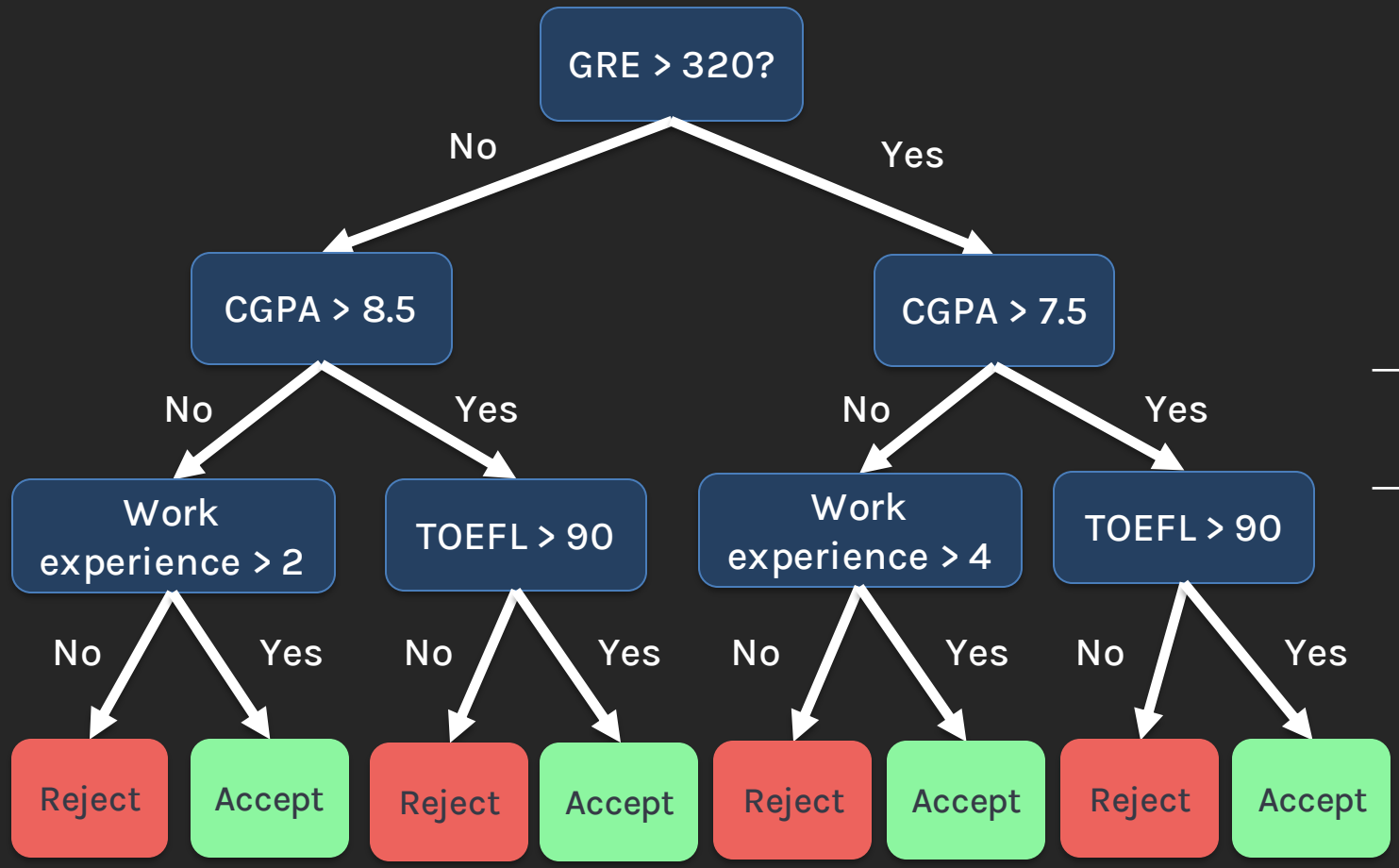
Pruning

There are many methods of pruning. A common one is the **cost complexity pruning:**



In other words, we add a 'regularization' term!

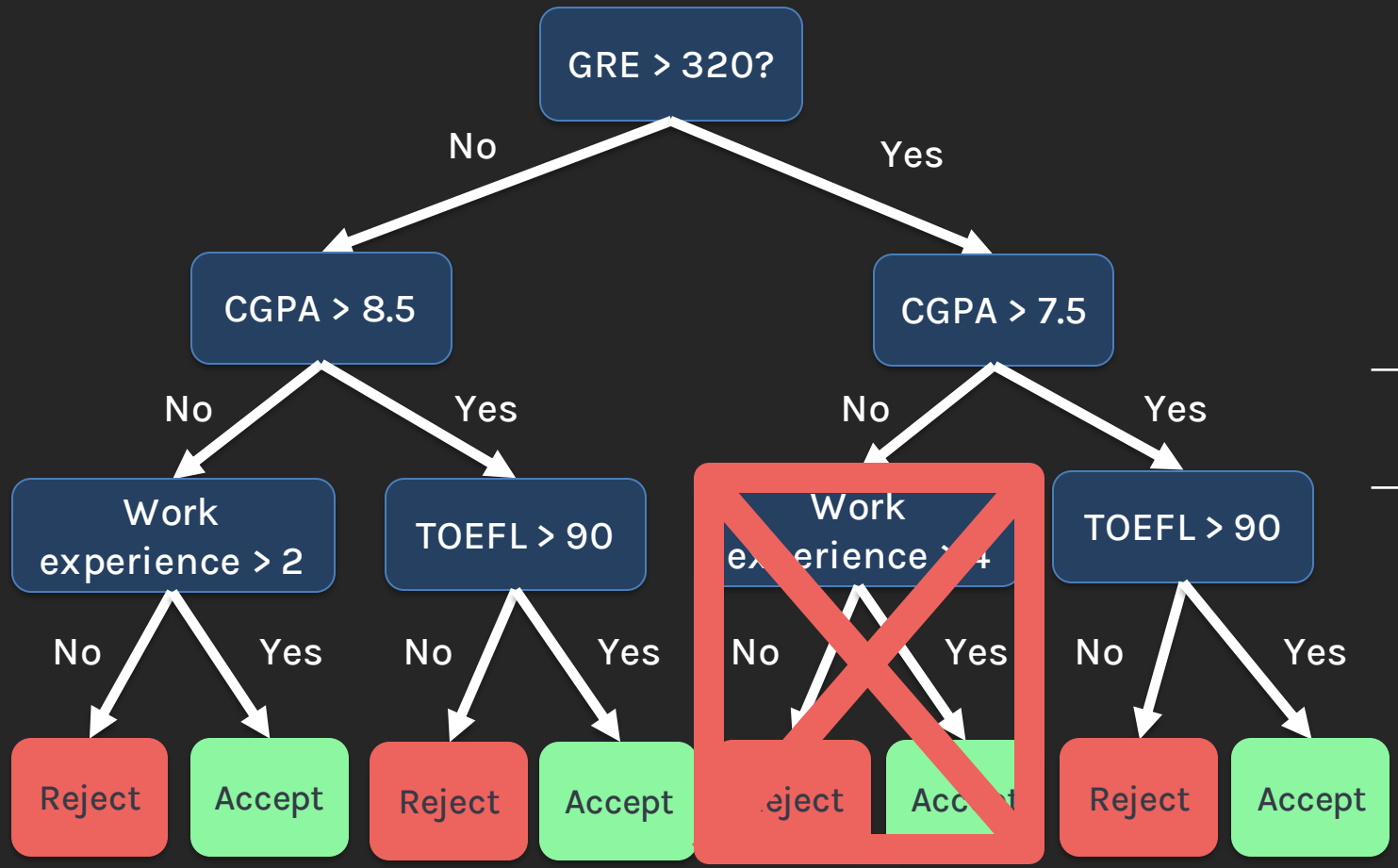
Cost-Complexity Pruning: Example



$\alpha = 0.2$

Tree	Error(T)	| T |	$Error(T) + \alpha \lvert T \rvert$
T	0.32	8	0.32 + 0.2*8 = 1.92

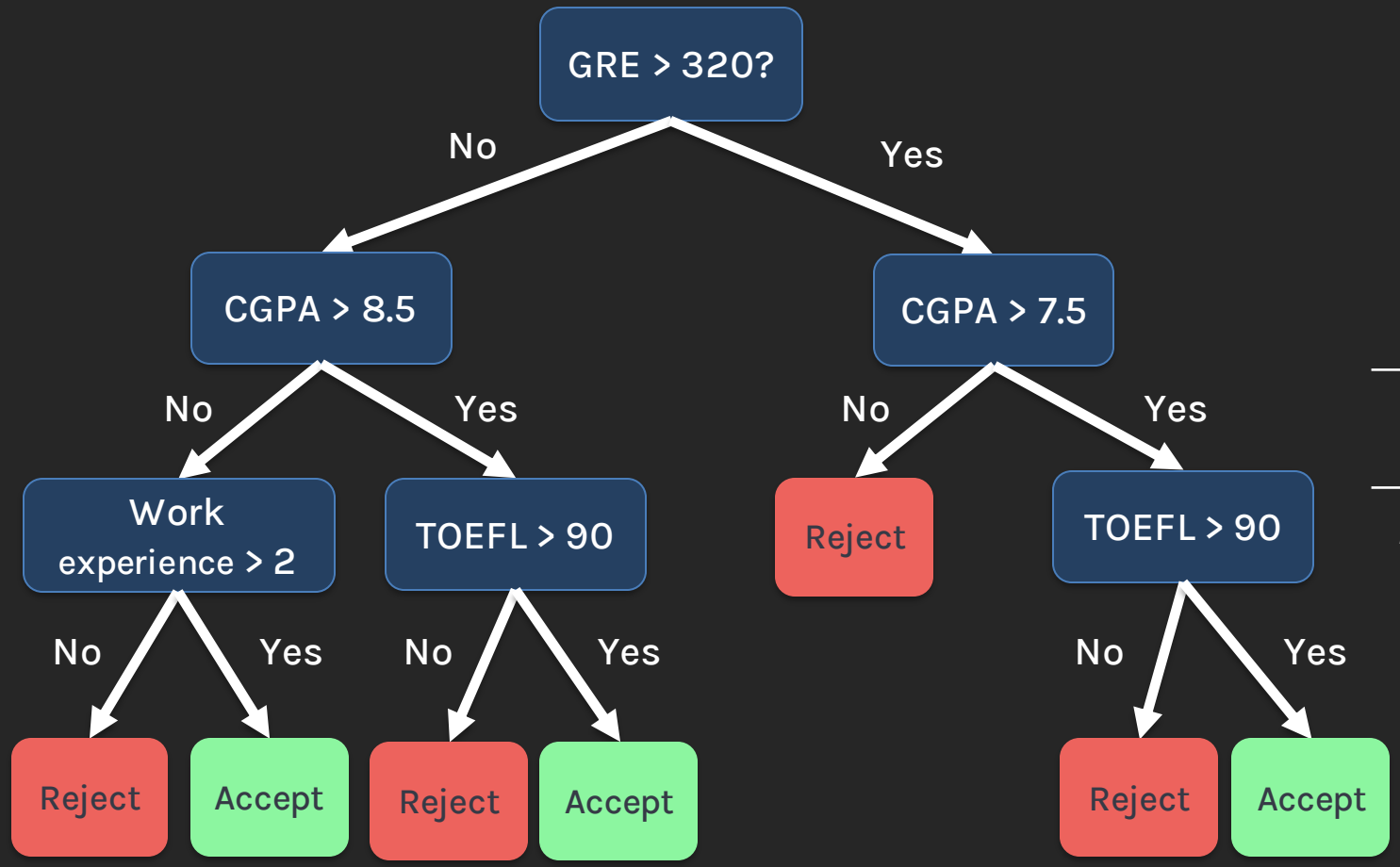
Cost-Complexity Pruning: Example



$\alpha = 0.2$

Tree	Error(T)	| T |	$Error(T) + \alpha \lvert T \rvert$
T	0.32	8	0.32 + 0.2*8 = 1.92

Cost-Complexity Pruning: Example

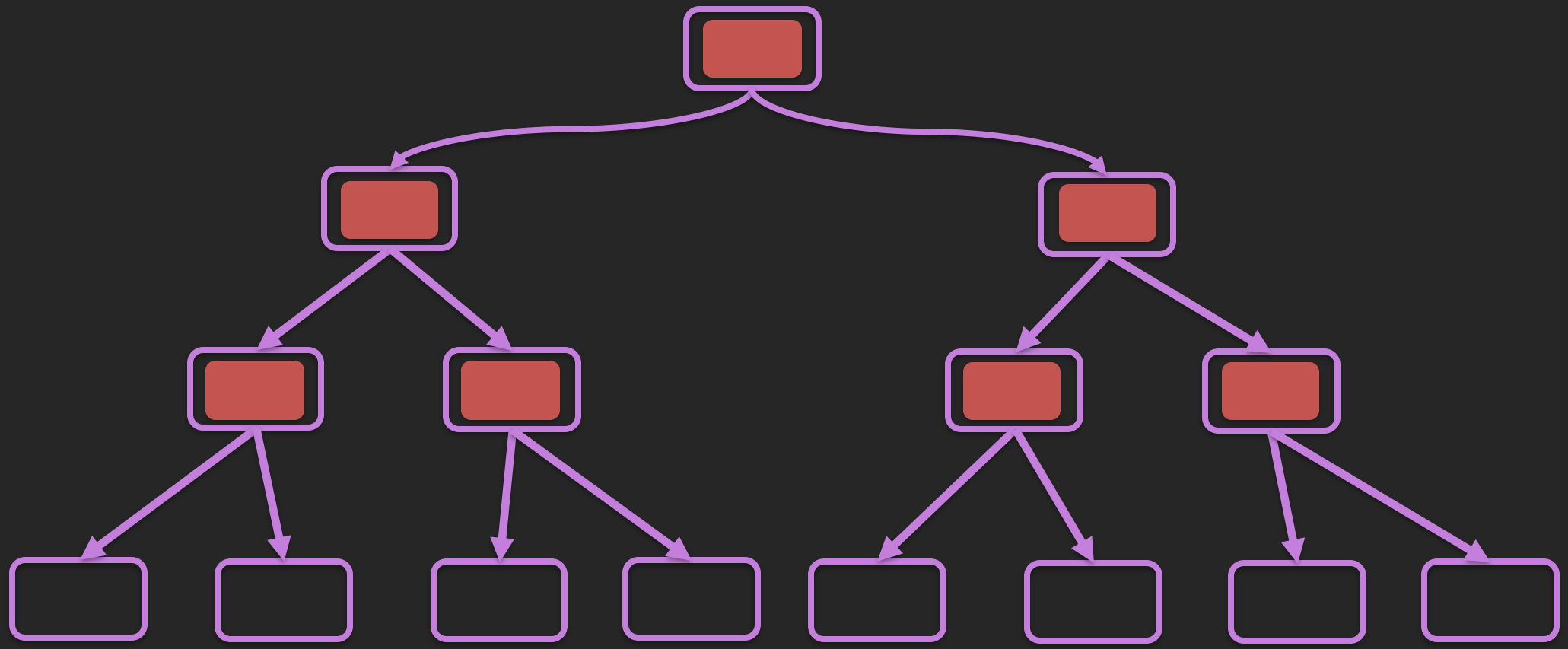


$\alpha = 0.2$

Tree	Error(T)	| T |	$Error(T) + \alpha \lvert T \rvert$
T	0.32	8	1.92
T_{small}	0.33	7	0.33 + 0.2*7 = 1.73

Pruning

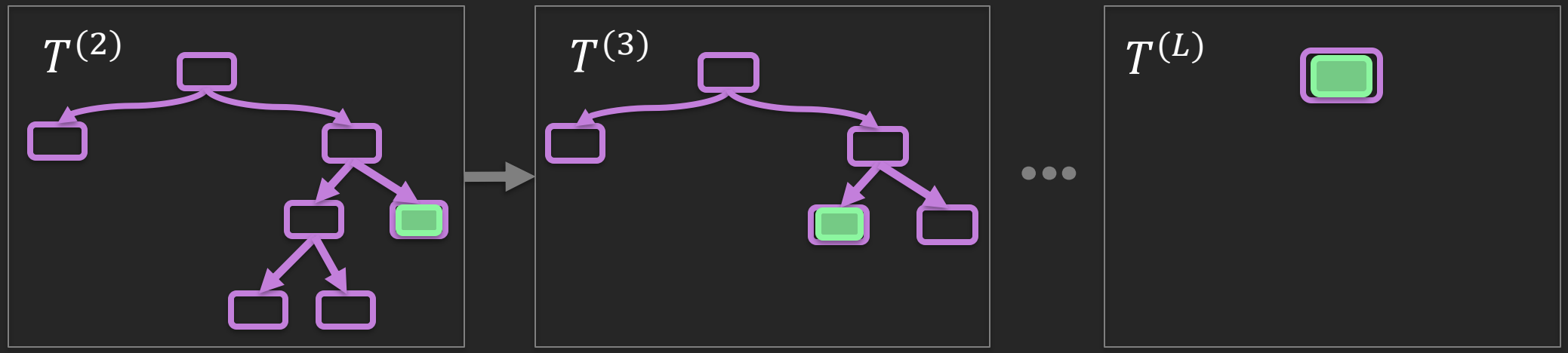
There are **7** possible pruning locations, which are shown with red rectangles.



We will choose the one that maximizes the difference of **cost complexity score** between a full tree and a pruned tree.

Pruning

We iterate this pruning process to obtain $T^{(2)}, T^{(3)}, \ldots, T^{(L)}$ where $T^{(L)}$ is the tree containing just the root of $T^{(0)}$.



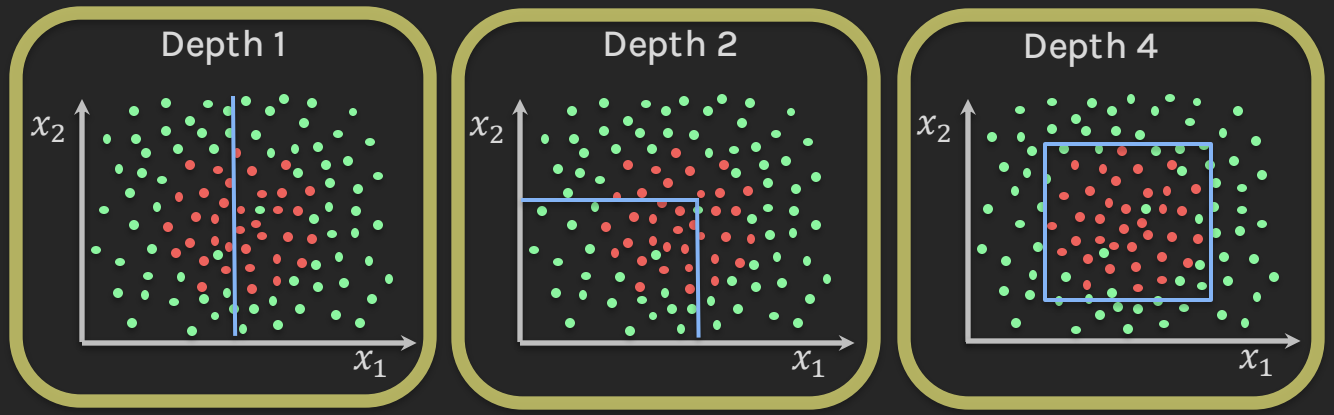
We select the optimal tree $T^{(i)}$ by cross validation.

This is the T^* given α. Finally, we choose **the optimal** α by cross validation!

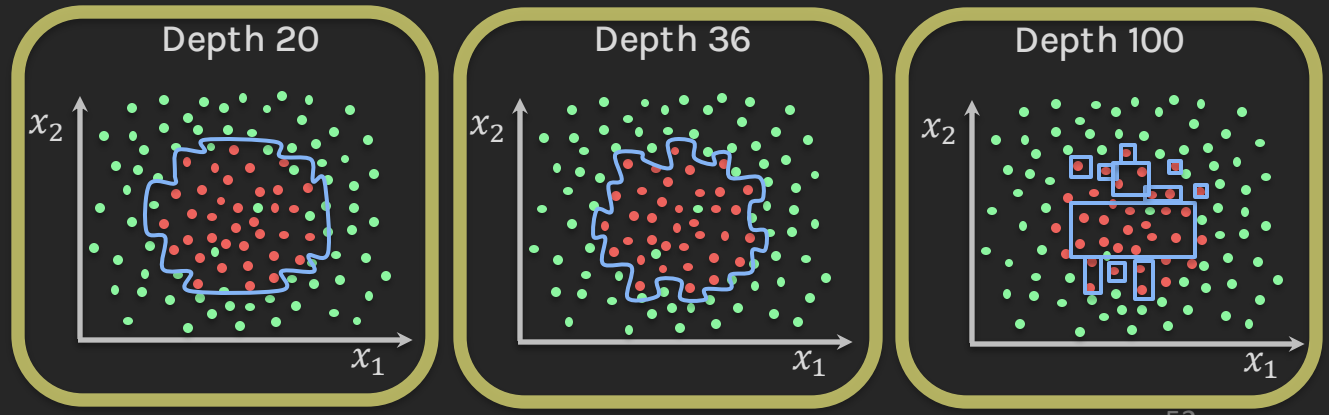
Bagging

Underfitting and Overfitting

When a tree is too shallow, it cannot divide the input data into enough regions, so the model underfits. When the tree is too deep, it cuts the input space into too many regions and fits to the noise of the data, so it overfits.



Underfit ⟷ Overfit



Ensemble Learning - Intuition

Instead of relying on the prediction from one specialist we consult multiple doctors.

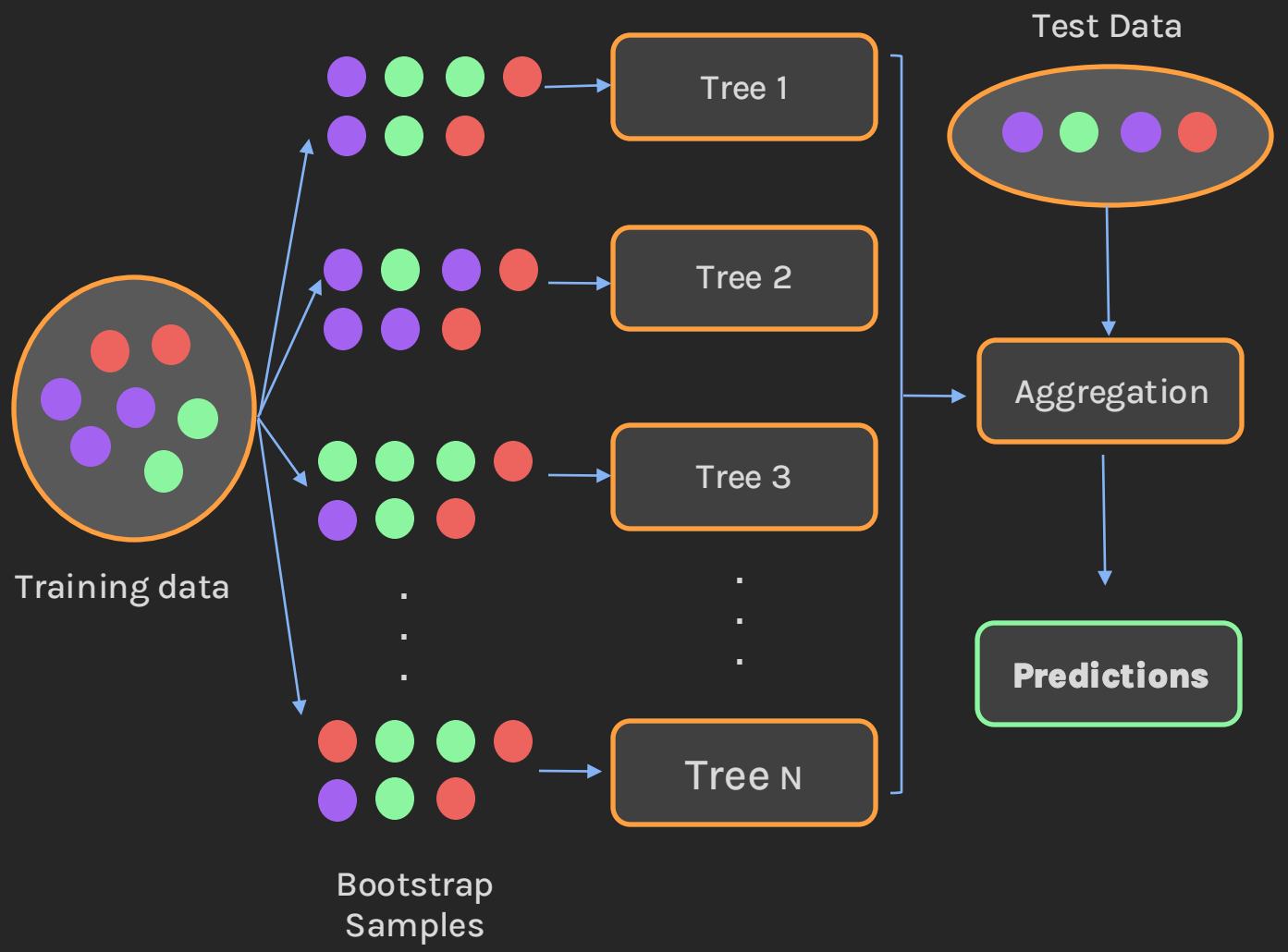
We individually ask each of these doctors to predict whether the scan indicates the presence of brain tumor.

Each 'specialist' or model views the MRI scans and makes a prediction. Their collective decisions are then aggregated to form a final verdict. This approach helps in mitigating individual model errors, increasing the accuracy of the prediction.

Bagging - Bootstrap + Aggregating

1. **Bootstrap:** we generate multiple samples of training data, via bootstrapping. We train a deep decision tree on each sample of data.

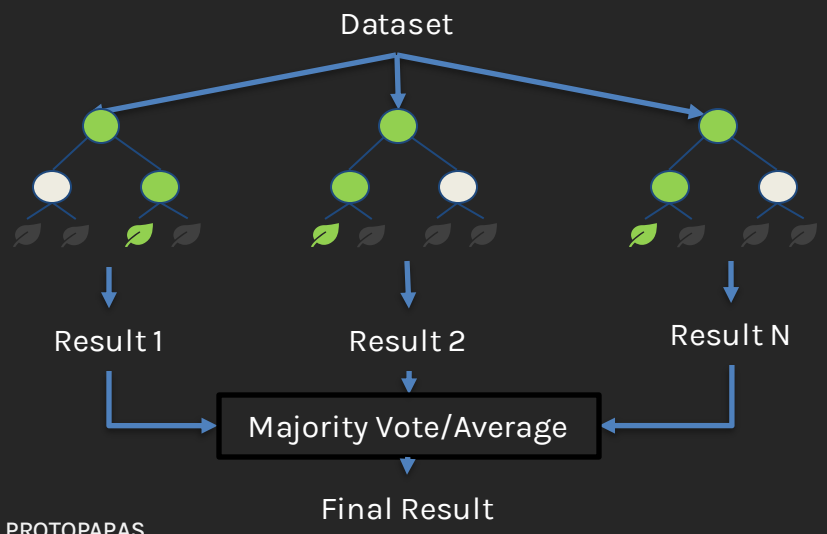
2. **Aggregating:** for a given input, we output the averaged outputs of all the models for that input.

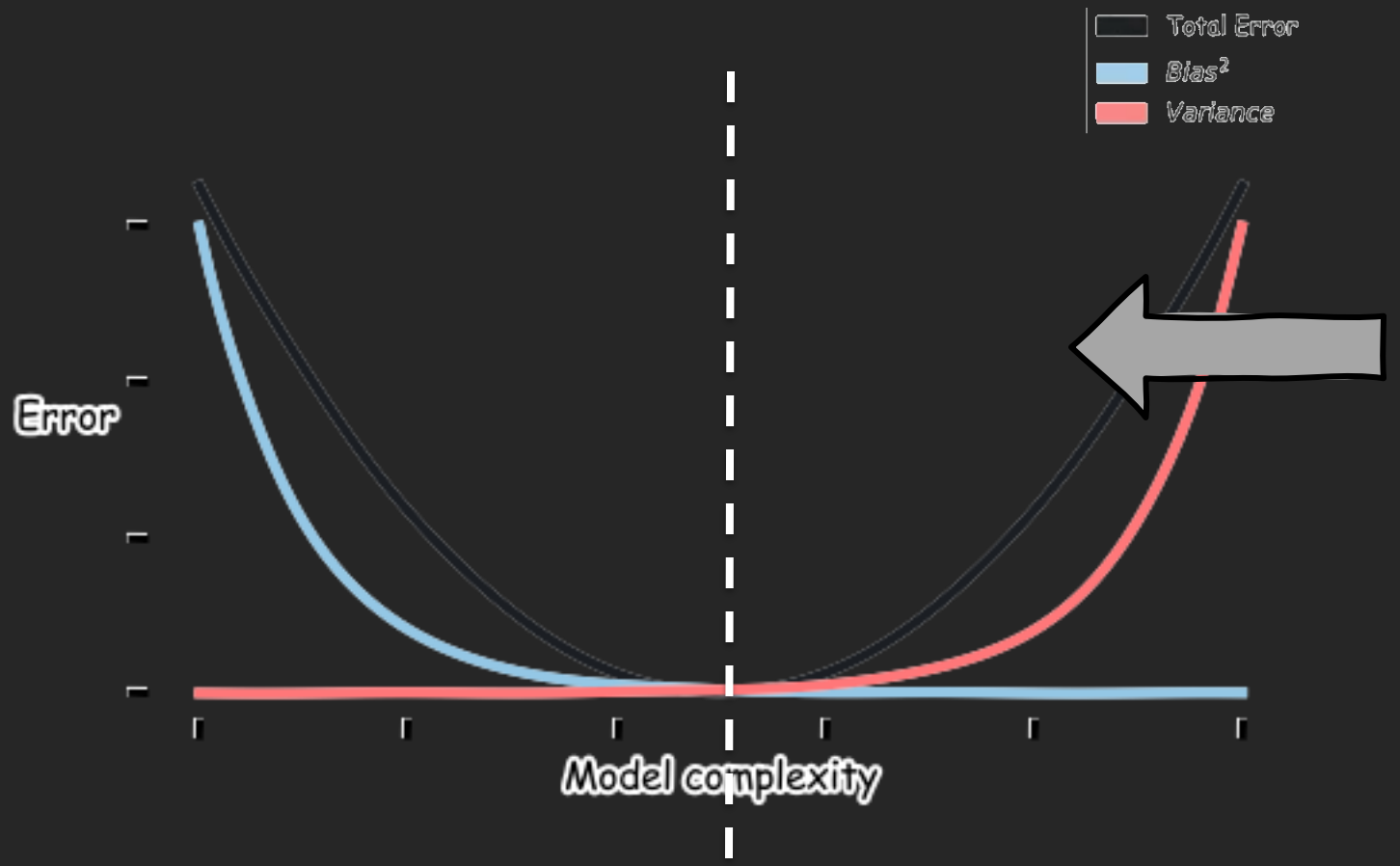


Bagging – Reducing Variance

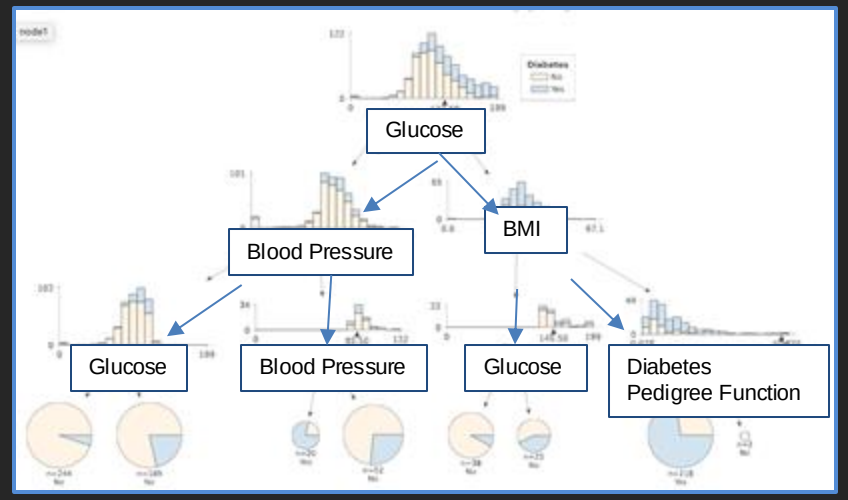
Reduce variance

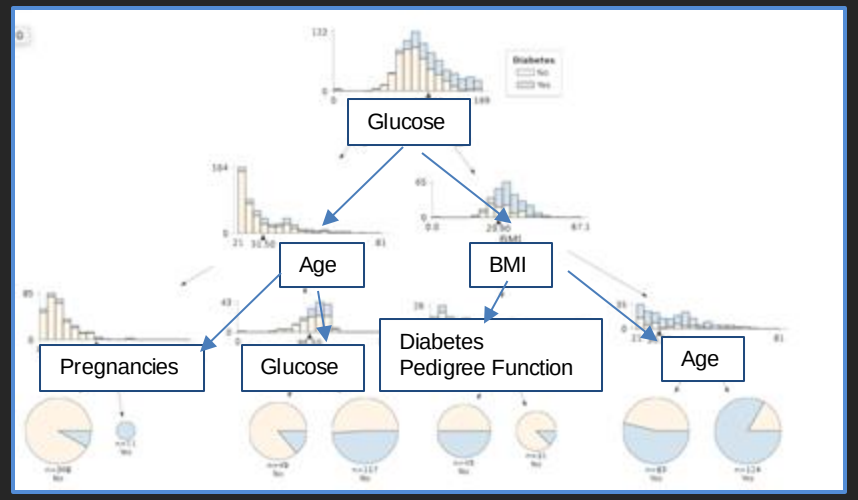
$$d_{\text{trees}} \to \infty$$
$$var \to 0$$

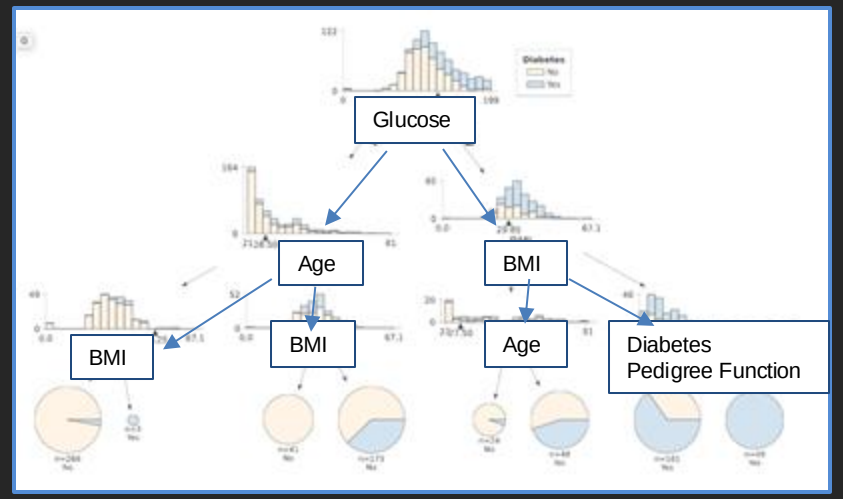


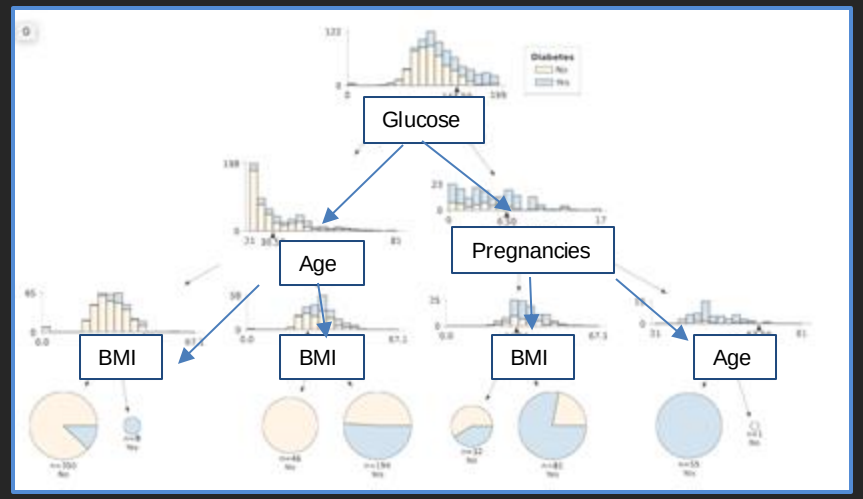


Classification in Bagging



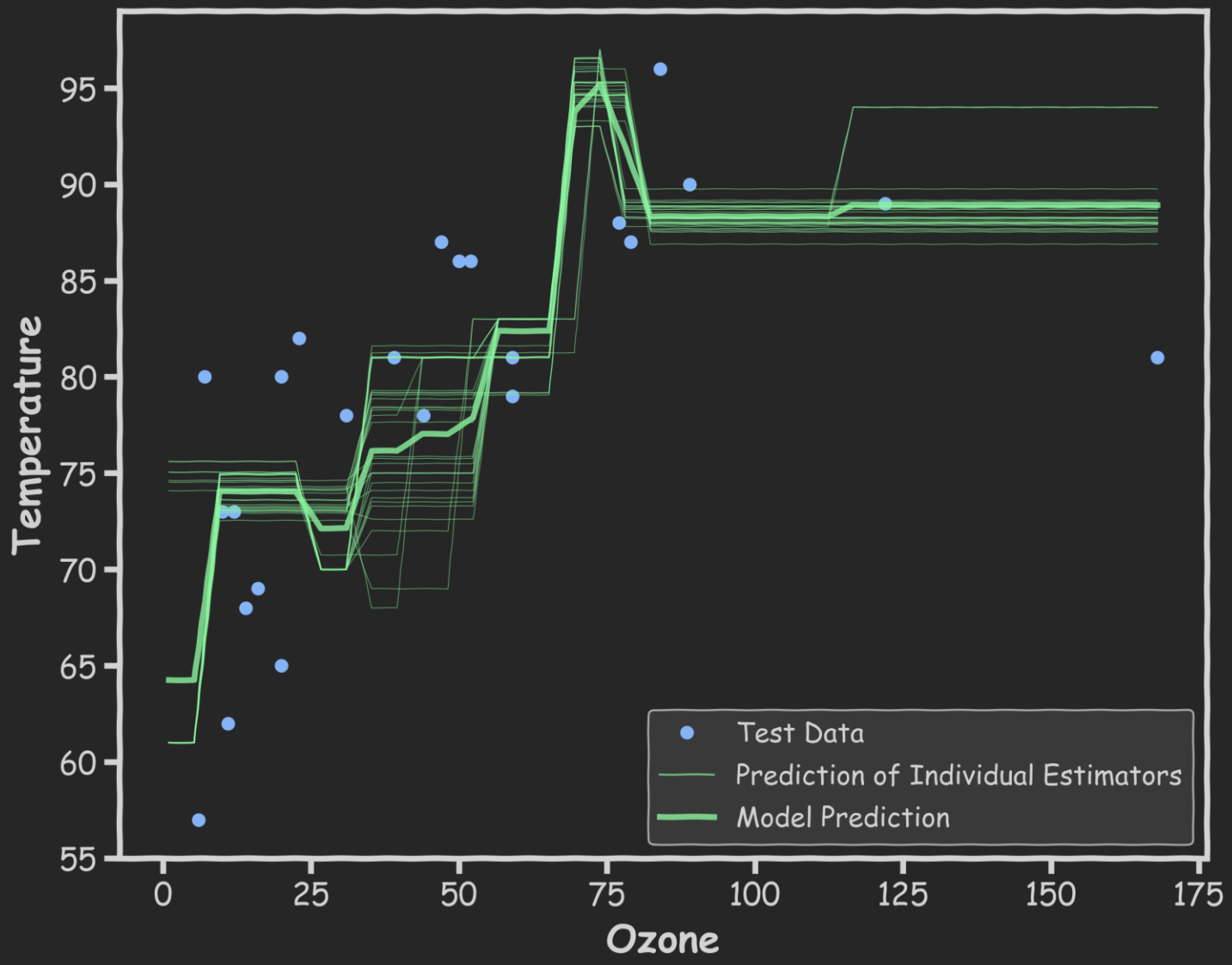






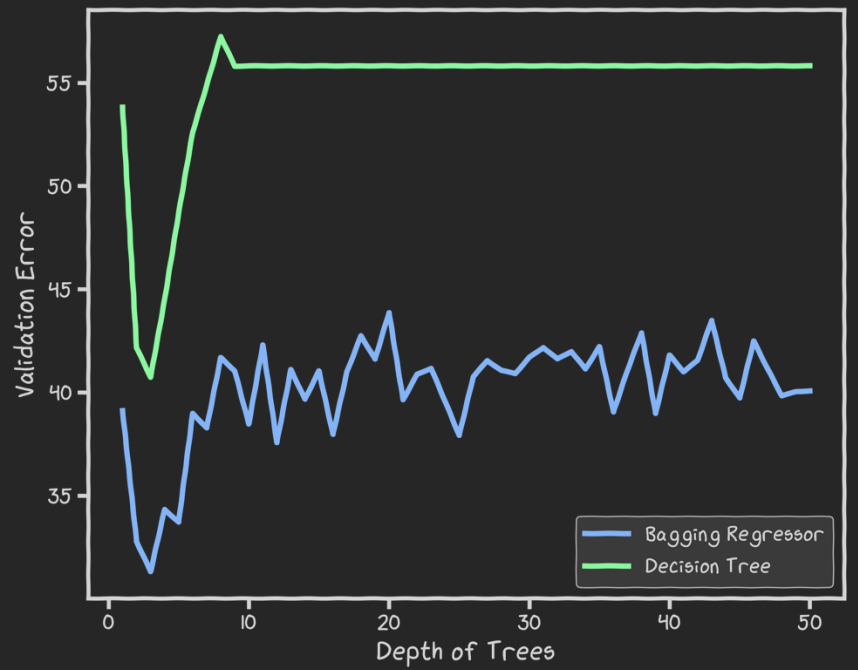
For each bootstrap, we build a decision tree. The results is a combination (majority) of the predictions from all trees.

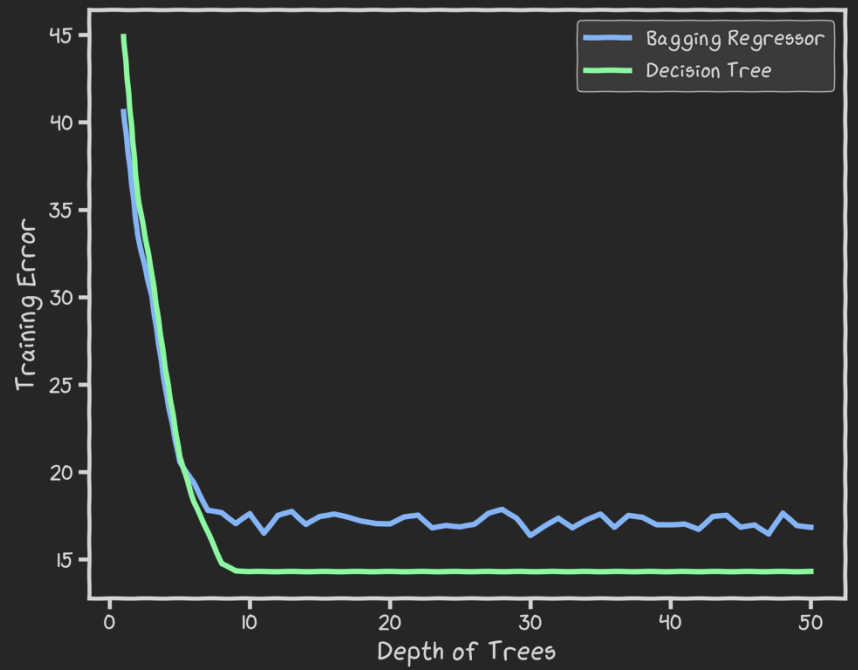
Prediction from all Decision Trees



The prediction by the bagging regressor model is the average of all the individual predictions of the trees.

Underfitting and Overfitting in Decision Tree vs. Bagging





- The graphs shows that the **more depth we add to Decision Trees, the faster the model overfits**

- However, for Bagging, there is still some form of **better model performance maintained after increase of depth of trees** although eventually it does **lead to no improvement of performance after a certain point as well**

Cases of underfitting and overfitting in Bagging

Cross validation is cool but computationally expensive and requires a larger dataset.

Before we conclude the bagging session, we will present another method of measuring the performance of ensemble methods.

Out-of-bag error

Out-of-bag Error (OOB)

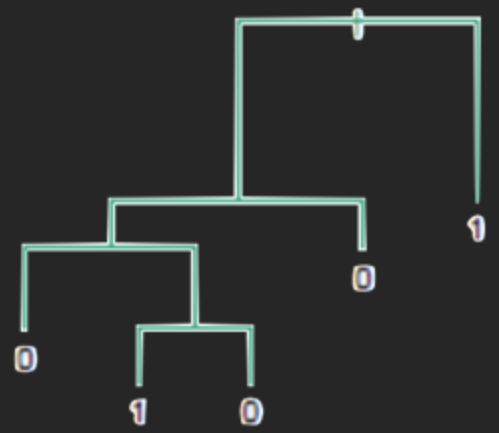
Original Data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
.	.
.	.
.	.
X_n	y_n

Bootstrap Sample 1

X	Y
X_1	y_1
X_3	y_3
X_5	y_5
X_{21}	y_{21}
X_{35}	y_{35}
.	.
.	.
.	.
X_k	y_k

Decision Tree 1



Used and unused data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
.	.
.	.
.	.
X_n	y_n

Predictor/Feature

Response/Target

Out-of-bag Error (OOB)

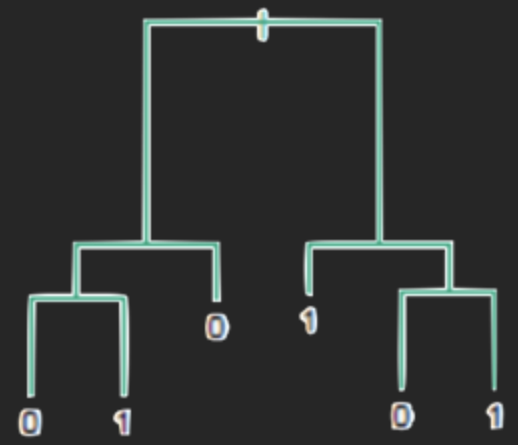
Original Data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
.	.
.	.
.	.
X_n	y_n

Bootstrap Sample 2

X	Y
X_5	y_5
X_7	y_7
X_{13}	y_{13}
X_{27}	y_{27}
X_{32}	y_{32}
.	.
.	.
.	.
X_k	y_k

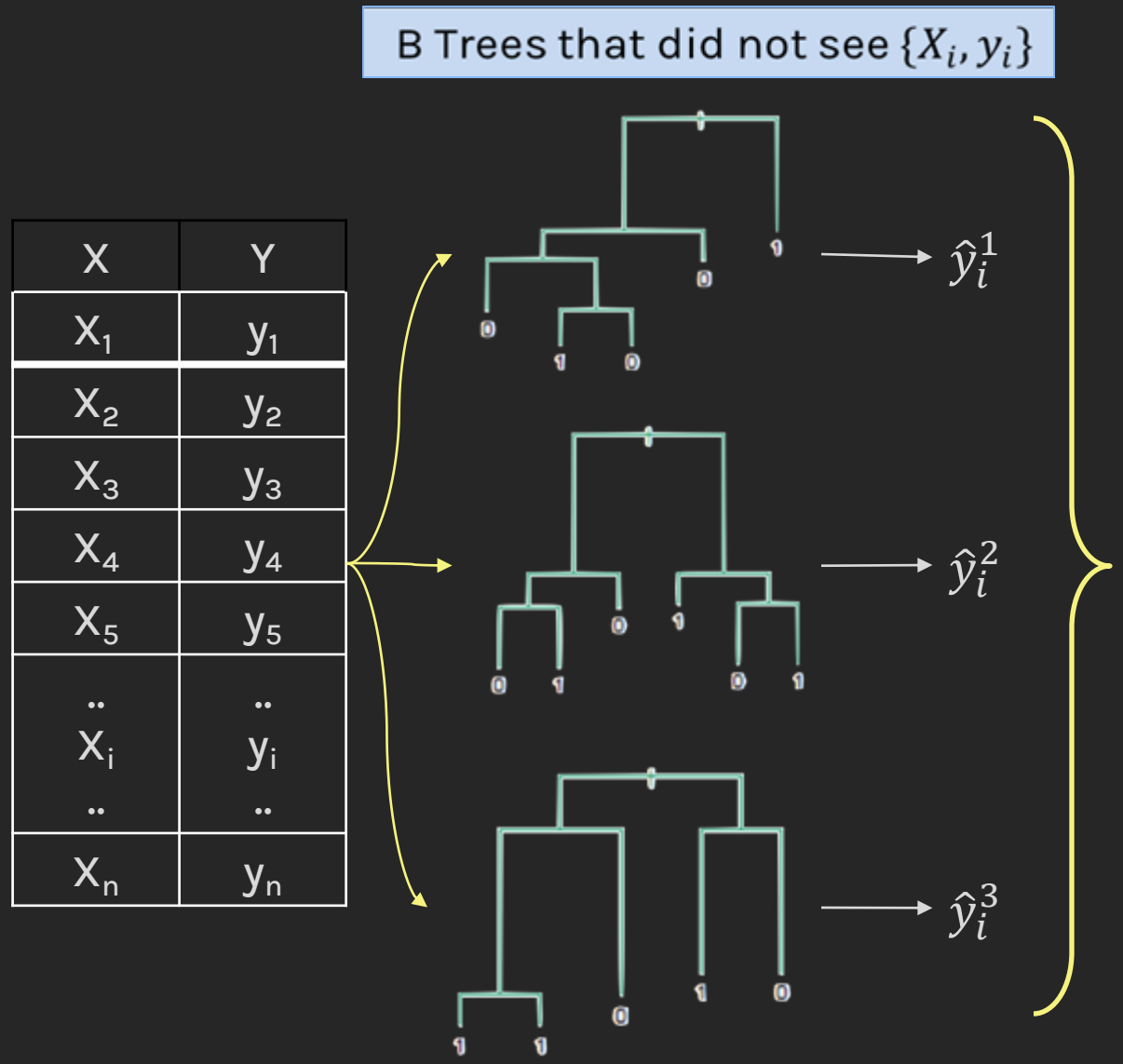
Decision Tree 2



Used and unused data

X	Y
X_1	y_1
X_2	y_2
X_3	y_3
X_4	y_4
X_5	y_5
.	.
.	.
.	.
X_n	y_n

Point-wise out-of-bag error



- Identify observations the trained models have not seen
- Get the predictions for these observations from the models

Point-wise out-of-bag error

Take majority for classification and average for regression tasks as the validation prediction for that observation

Point-wise prediction

Point-wise out-of-bag error

Classification

$$\hat{y}_{i,pw} = majority(\hat{y}_i^j)$$

$$e_i = \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$$

Regression

$$\hat{y}_{i,pw} = \frac{1}{B}\sum_{j \in B} \hat{y}_{i,j}$$

$$e_i = (y_i - \hat{y}_{i,pw})^2$$

OOB Error

We average the point-wise out-of-bag errors over the full training set.

Classification

$$Error_{OOB} = \frac{1}{N}\sum_{i}^{N} e_i = \frac{1}{N}\sum_{i}^{N} \mathbb{I}(\hat{y}_{i,pw} \neq y_i)$$

Regression

$$Error_{OOB} = \frac{1}{N}\sum_{i}^{N} e_i = \frac{1}{N}\sum_{i}^{N} (y_i - \hat{y}_{i,pw})^2$$

Improving on Bagging

In practice, the trees in Bagging tend to be **highly correlated**.

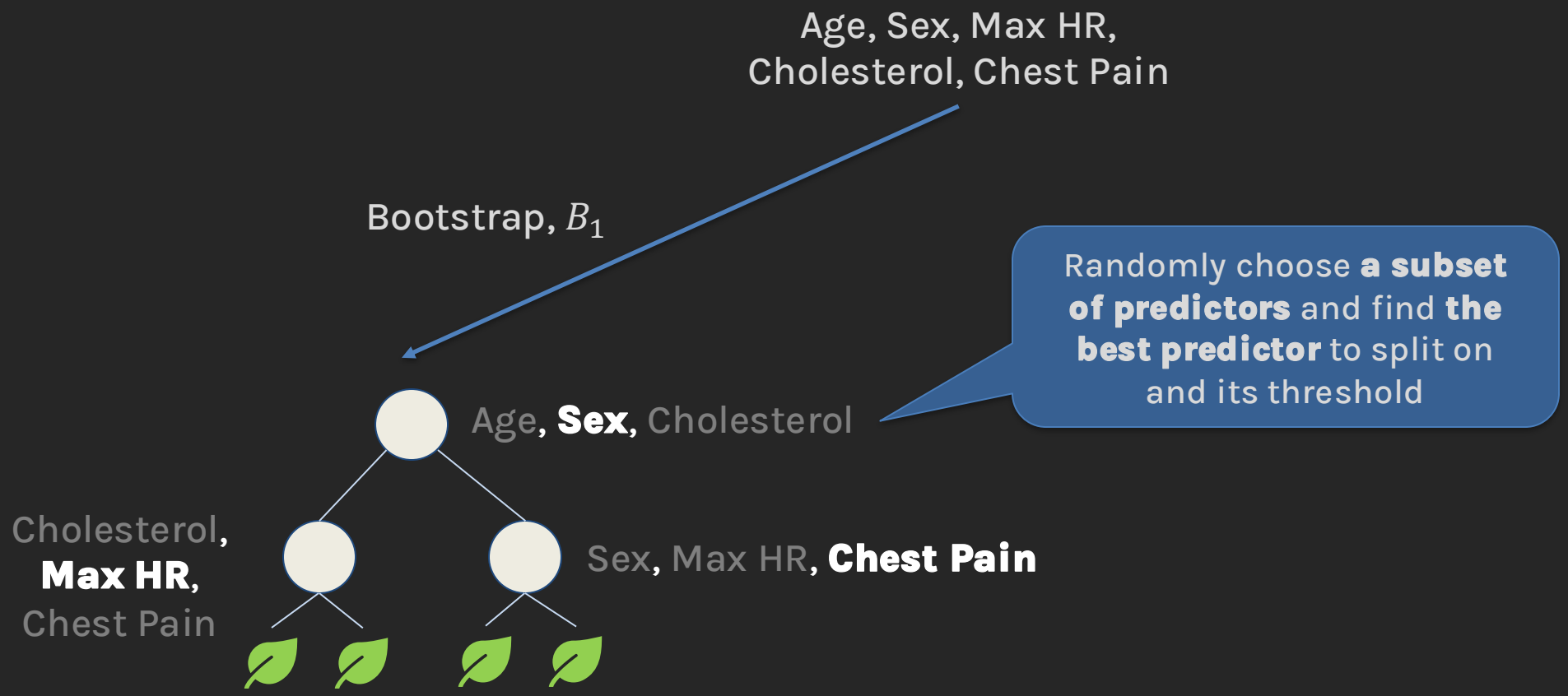
- Suppose we have an **extremely strong predictor**, x_j, in the training set amongst **moderate predictors**. Then the greedy learning algorithm ensures that most of the models in the ensemble will choose to split on x_j in early iterations.

- However, we assumed (or hope) that each tree in the ensemble is **independently and identically distributed**.

Random Forest

Random Forest

Consider a dataset that contains the following predictors:



Tuning Random Forests

Random forest models have multiple **hyper-parameters** to tune:

1. The **number of predictors** to randomly select at each split.

2. The **total number of trees** in the ensemble.

3. The **stopping criteria** - maximum depth, minimum leaf node size, etc.

4. The **splitting criterium** – gini, entropy

Tuning Random Forests

There are standard (default) values for each of random forest hyper-parameters recommended by long time practitioners.

For **THE NUMBER OF PREDICTORS**

$\sqrt{N_j}$ predictors for classification

$\frac{N_j}{3}$ predictors for regression

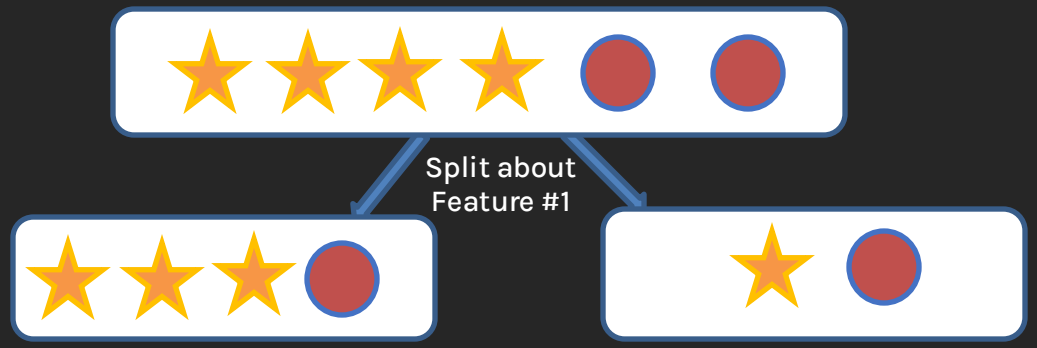
For **THE NUMBER OF TREES**, we use out-of-bag errors. With OOB, training and validation can be done in a single sequence - once the out-of-bag error stabilizes, adding more trees may not be beneficial.

Also, generally these parameters should be tuned through **OOB** (making them data and problem dependent).

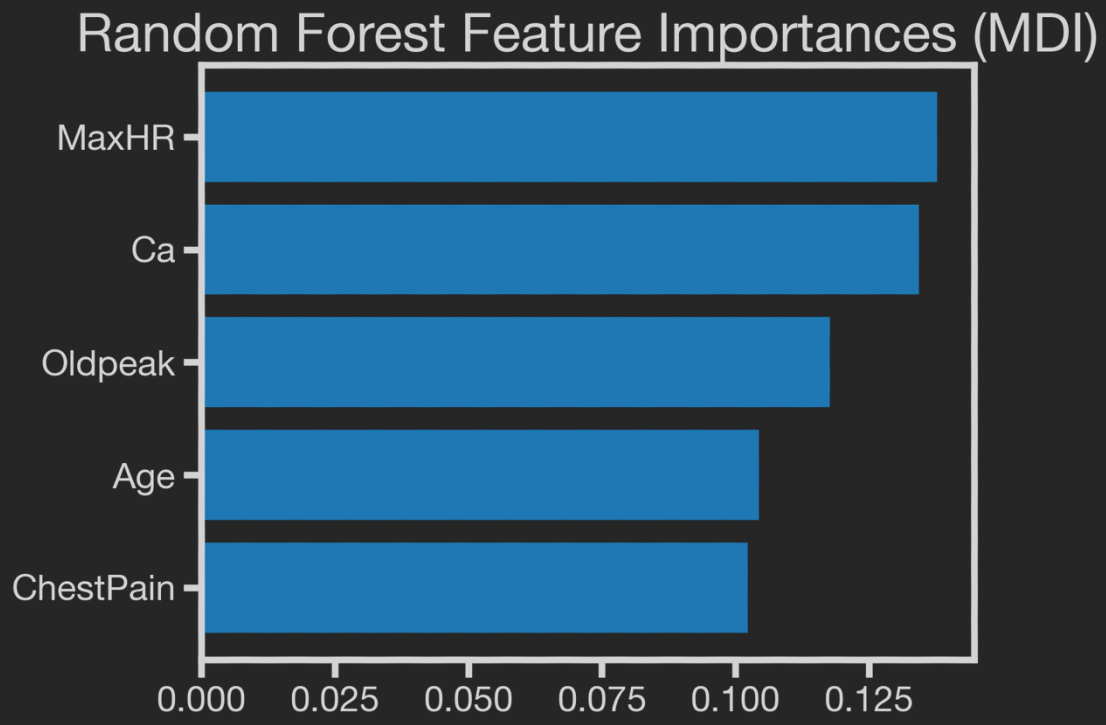
Variable Importance

Mean Decrease in Impurity (MDI)

Decision trees make splits that maximize the decrease in impurity. By calculating the mean decrease in impurity for each feature across all trees, we arrive at the variable importance for a bagging or random forest model.



Average this over all the trees



Mean Decrease in Impurity (MDI)

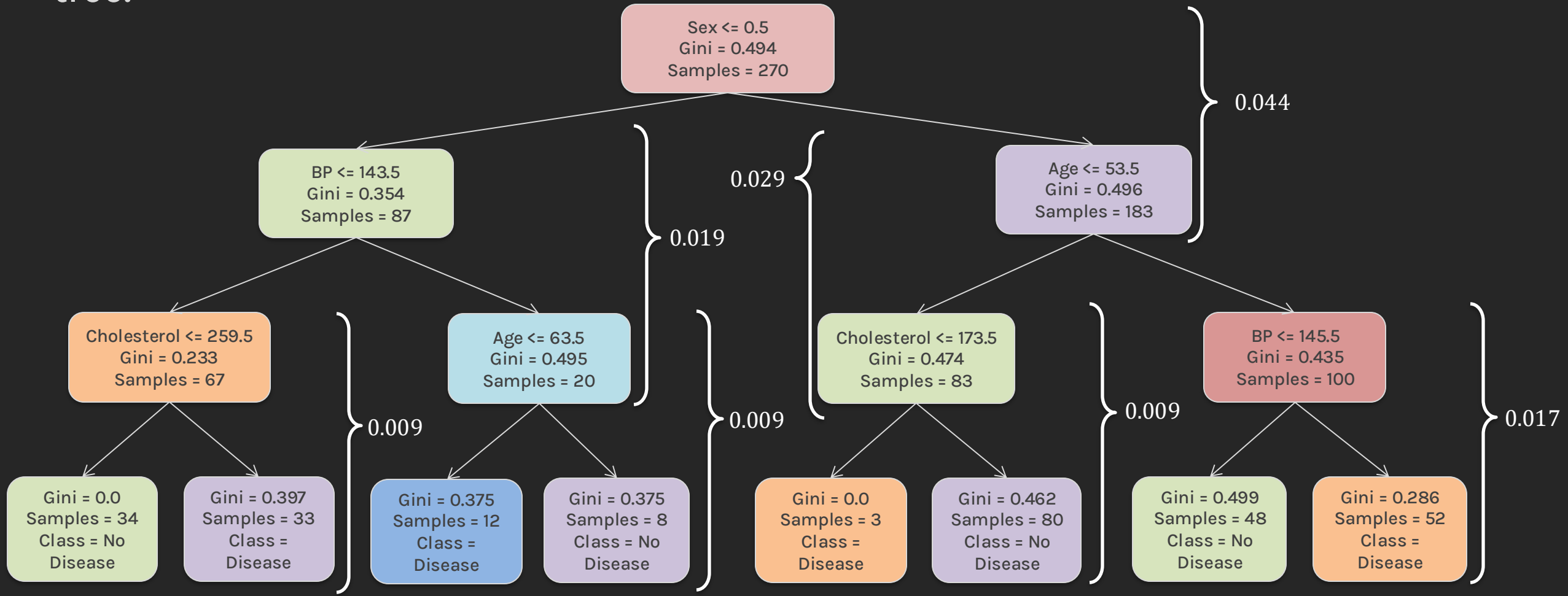
Step 1: Calculate the mean decrease in impurity for each node q in the decision tree.

$$\Delta I_q = \left(\frac{n}{N}\right)\left[Gini_n - \left(\frac{m_L}{n}\right) Gini_{m_L} - \left(\frac{m_R}{n}\right) Gini_{m_R}\right]$$

Mean Decrease in Impurity (MDI)

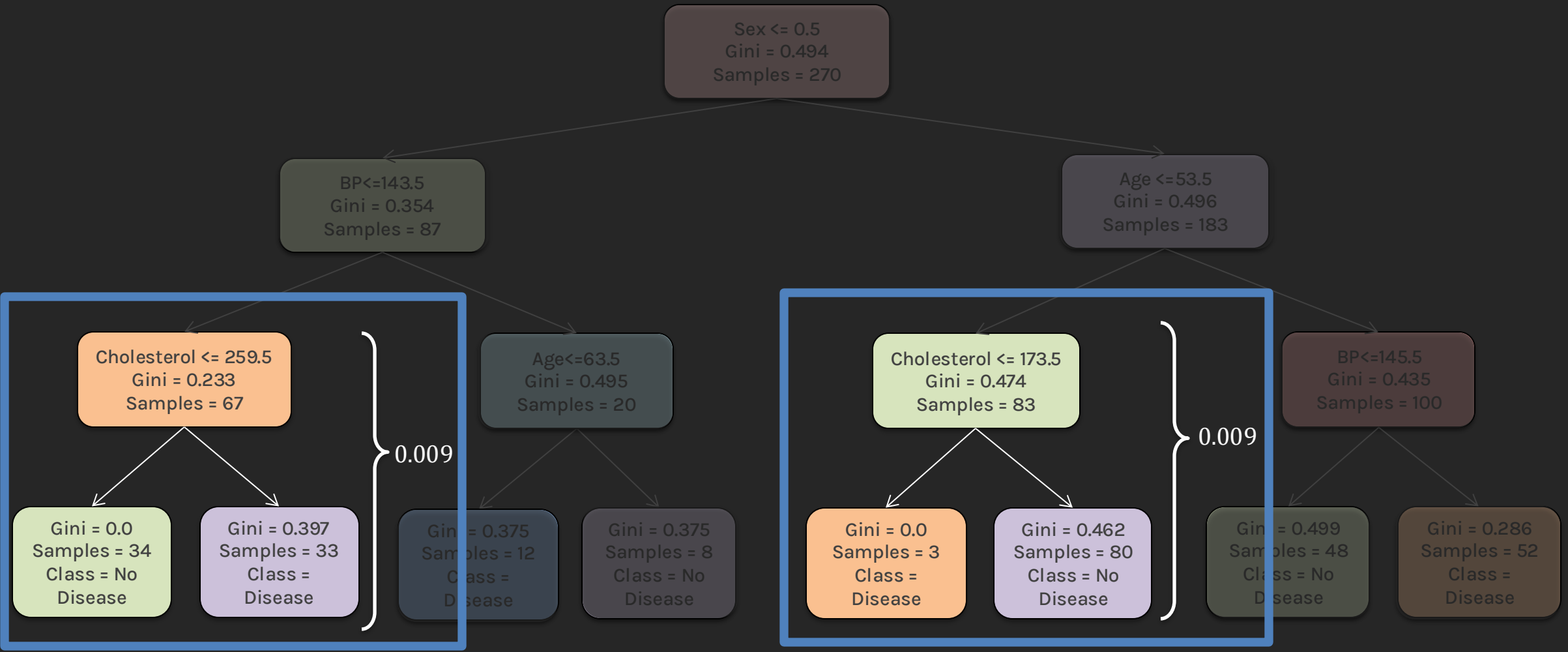
$$\Delta I_q = \left(\frac{n}{N}\right)\left[Gini_n - \sum\left(\frac{m}{n}\right) Gini_m\right]$$

Step 1: Calculate the mean decrease in impurity for each node q in the decision tree.



Mean Decrease in Impurity (MDI)

Let us try to compute the importance of the feature **cholesterol**:



Mean Decrease in Impurity (MDI)

Step 2: Calculate the importance of the feature by **summing up** the impurity decrease calculated in step 1 for each node in which that feature occurs.

$$\text{Feature Importance}_{cholesterol} = \sum_{n \in nodes\ split\ on\ cholesterol} Impurity\ Decrease_n$$

$$= 0.009 + 0.009 = 0.018$$

Mean Decrease in Impurity (MDI)

Step 3: **Normalize** this value between 0 and 1 by dividing by the sum of all feature importance values.

This ensures the sum of all feature importance in a decision tree adds up to 1.

$$\text{Norm Feature Importance}_{cholesterol} = \frac{\text{Feature Importance}_{cholesterol}}{\sum_{j \in all\ features} \text{Feature Importance}_{j}}$$

Mean Decrease in Impurity (MDI)

Step 4: To calculate the feature importance at the Random Forest or Bagging level, we **average** the normalized feature importance of the given predictor over **all the trees**.

$$\text{Norm RF/Bagging Feature Importance}_{cholesterol} = \frac{\sum_{t \in all\ trees} \text{Norm Feature Importance}}{Total\ number\ of\ trees}$$

Summary: Mean Decrease in Impurity (MDI)

For each feature j in the dataset:

Step 1: Calculate the **mean decrease in impurity** for **each node** n in the decision tree t.

Step 2: Calculate the **feature importance** by **summing up** the impurity decrease calculated in step 1 for each node n in which that feature j occurs.

$$F_j^{(t)} = \sum_{n \in nodes_j} I_n^{(t)}$$

Step 3: **Normalize** this value between 0 and 1 by dividing by the sum of all feature importance values.

$$\hat{F}_j^{(t)} = \frac{F_j^{(t)}}{\sum_i F_i^{(t)}}$$

Step 4: At the Random Forest level, **average** over all the T trees.

$$\mathcal{F}_j = \frac{\sum_t \hat{F}_j^{(t)}}{T}$$

Permutation Importance

Consider the following dataset:

Height (cm)	Weight (kg)	…	Fitness Level (1 – 5)
150	65	…	2
140	50	…	3
…	…	…	…
170	70	…	4
160	80	…	1

Step 1: Record the validation/OOB accuracy of RF model:
Accuracy = 0.88

Permutation Importance

Step 1: Record the validation/OOB accuracy of RF model:
Accuracy = 0.88

Height (cm)	Weight (kg)	…	Fitness Level (1 – 5)
150	65	…	2
140	50	…	3
…	…	…	…
170	70	…	4
160	80	…	1

Step 2: Randomly permute the data for column j in the validation/OOB set.

Permutation Importance

Step 1: Record the validation/OOB accuracy of RF model:
Accuracy = 0.88

Step 2: **Randomly permute** the data for the column j (in this case Weight).

Record the validation/OOB accuracy of the RF model on the modified dataset:
Permuted Accuracy = 0.87

Step 3: Repeat the previous step K number of times and average all the accuracies.

Assume we permute the feature 3 times, we get:

Permuted Accuracy$_1$ = 0.82

Permuted Accuracy$_2$ = 0.87 ⟶ **Avg Permuted Accuracy=** $\frac{0.82+0.87+0.86}{3}$ **= 0.85**

Permuted Accuracy$_3$ = 0.86

Permutation Importance

Step 1: Record the validation/OOB accuracy of RF model:
Accuracy = 0.88

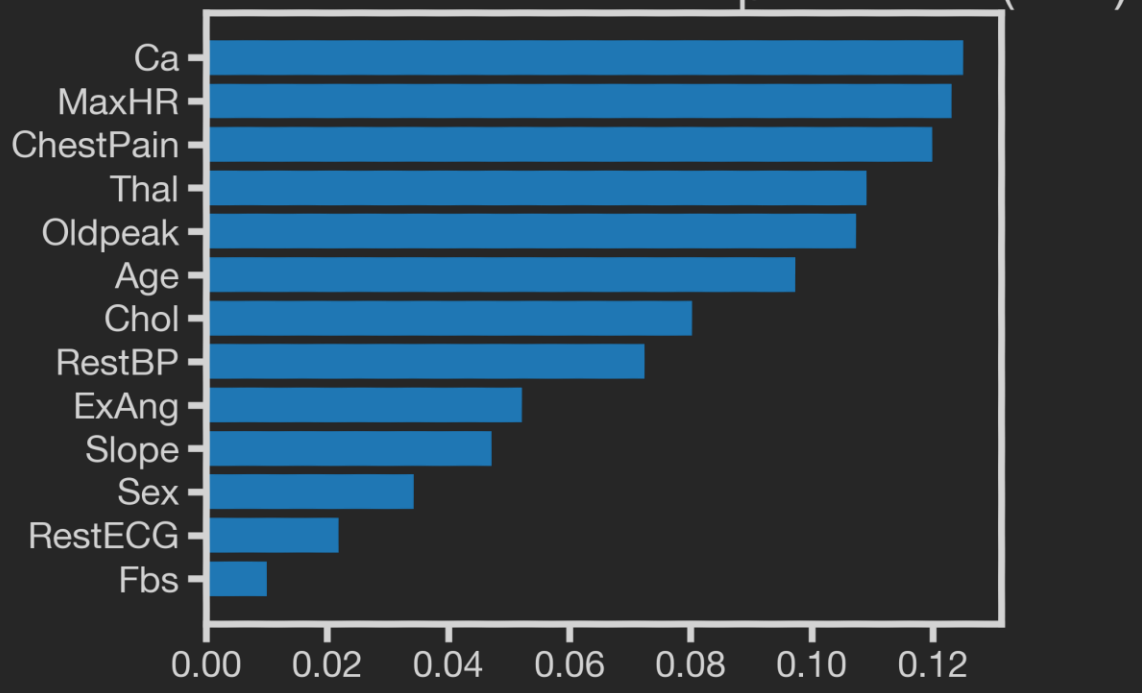
Step 2: **Randomly permute** the data for the column j (in this case Weight).

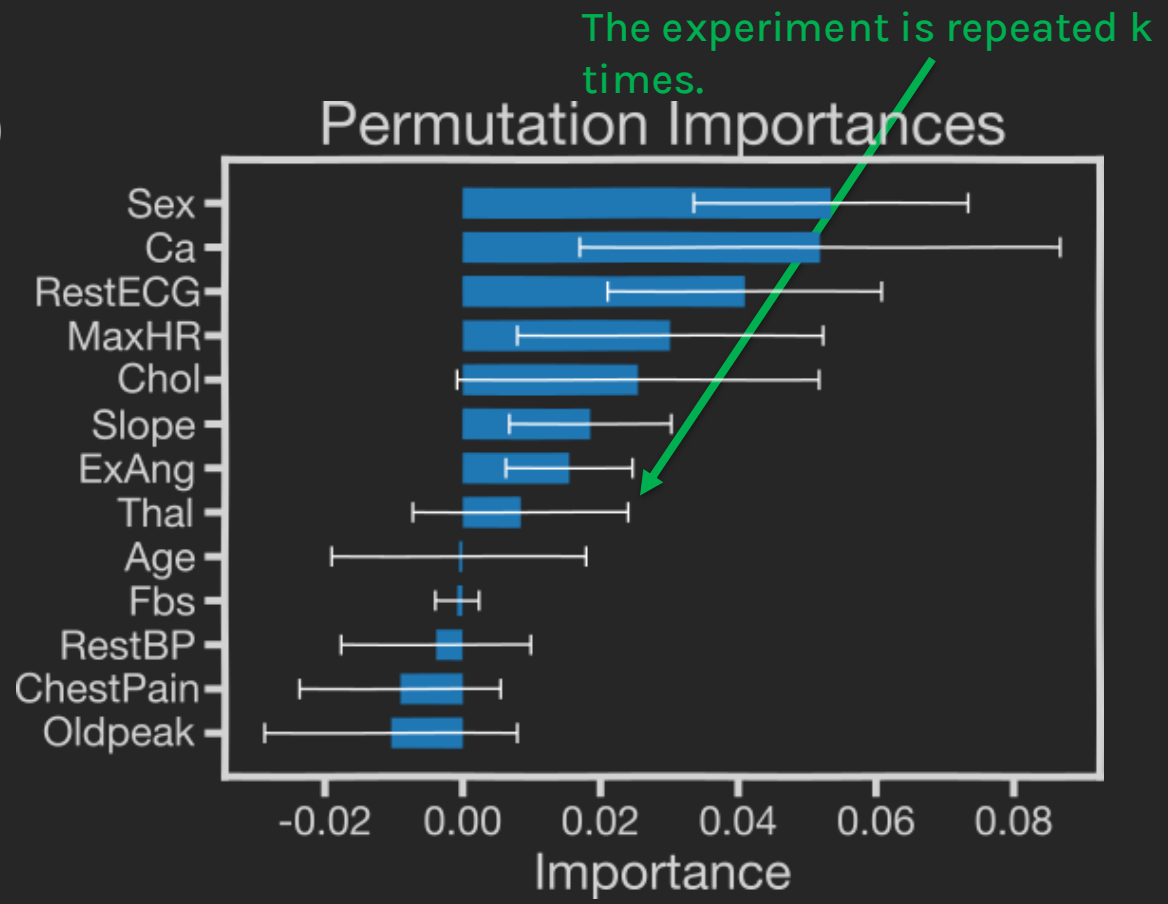
Record the validation/OOB accuracy of the RF model on the modified dataset:
Permuted Accuracy = 0.87

Step 3: Repeat the previous step with K number of times and average all the accuracies:
Avg Permuted Accuracy= 0.85

Step 4: Calculate the difference between unpermuted and average permuted accuracy to get the importance of the feature in the random forest:
Difference = 0.88 - 0.85 = 0.03

MDI vs Permutation Importance



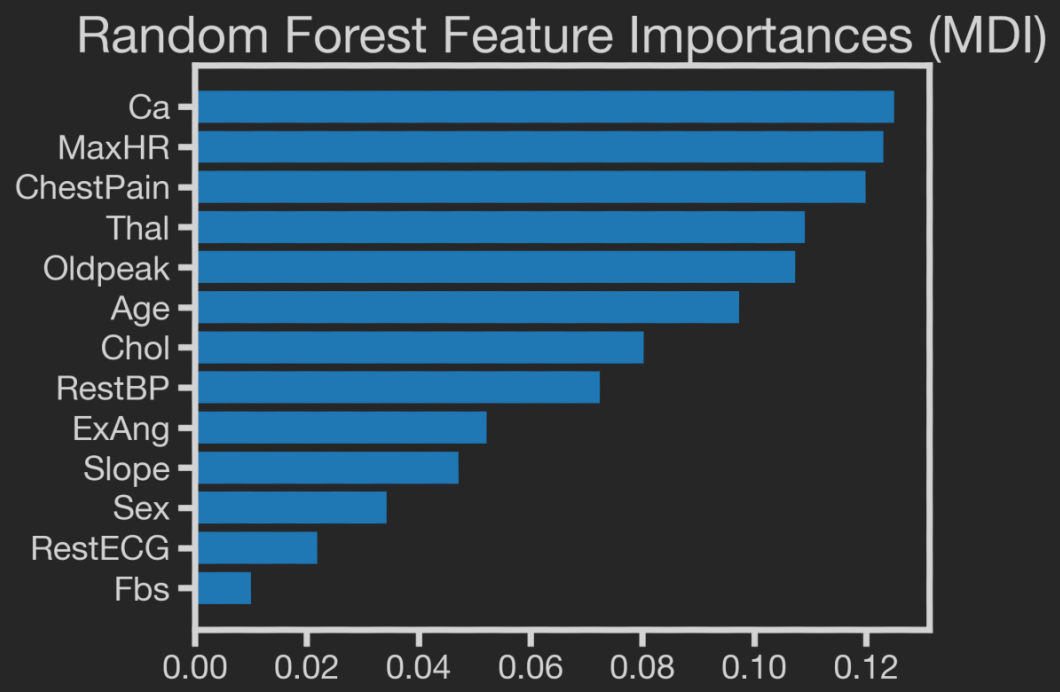


Features like ***ChestPain***, ***OldPeak***, ***Thal*** are ranked most important in MDI importance plot, but they are ranked low in permutation importance plot.

MDI vs Permutation Importance

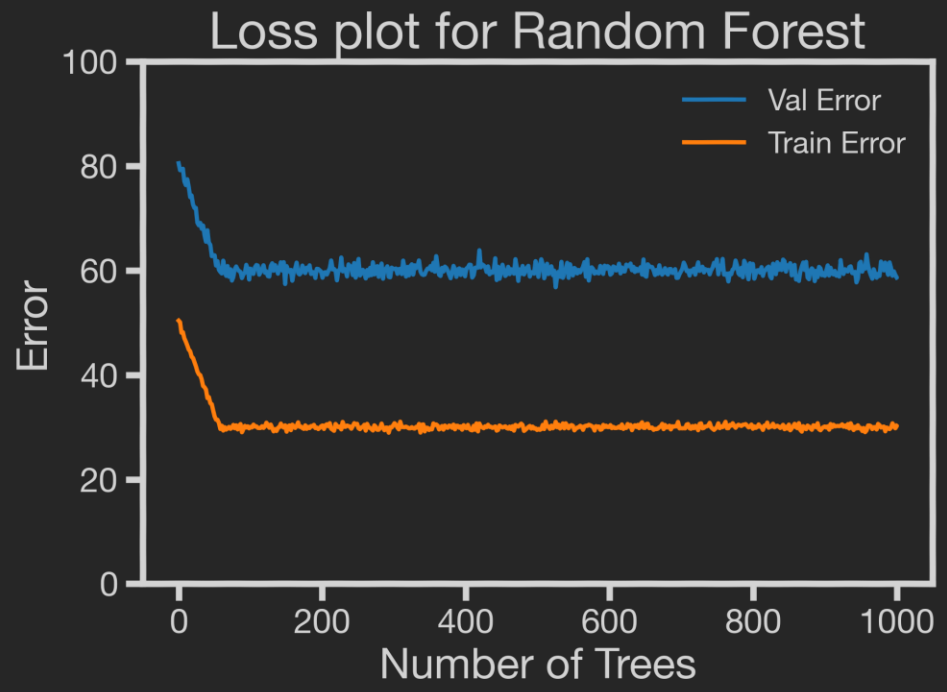
- The biggest advantage of the MDI is **speed of computation**. All needed values are computed during the Random Forest training.

- The drawbacks of the method is its tendency to prefer (select as important) numerical features and categorical features with **high cardinality**. In the example shown, *Max HR* is selected as an important feature because of the high cardinality.



Final Thoughts on Random Forests

Increasing the number of trees in the ensemble generally does not increase the risk of overfitting.



By decomposing the generalization error in terms of bias and variance, we see that increasing the number of trees produces a model that is at least as robust as a single tree.