

Decision Trees – Classification

Pavlos Protopapas

Outline

- Motivation
- Decision Trees - Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions
- Decision Trees - Regression

Outline

- **Motivation**
- Decision Trees - Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions
- Decision Trees - Regression

When I hear someone mention
food



Motivation

The simplest and most straightforward classification model we know is logistic regression.

Logistic regression works best when:

- a) The classes are well-separated in the feature space, and
- b) The classification boundary has a nice geometry



Motivation

Classification Boundaries or **Decision Boundaries** are defined where the probabilities of being in class 1 and class 0 are equal, i.e.

$$P(Y = 1) = P(Y = 0)$$

Motivation

Classification Boundaries or **Decision Boundaries** are defined where the probabilities of being in class 1 and class 0 are equal, i.e.

$$P(Y = 1) = P(Y = 0) = 1 - P(Y = 1)$$

$$\Rightarrow P(Y = 1) = 0.5$$

which is equivalent to when the log-odds are zero:

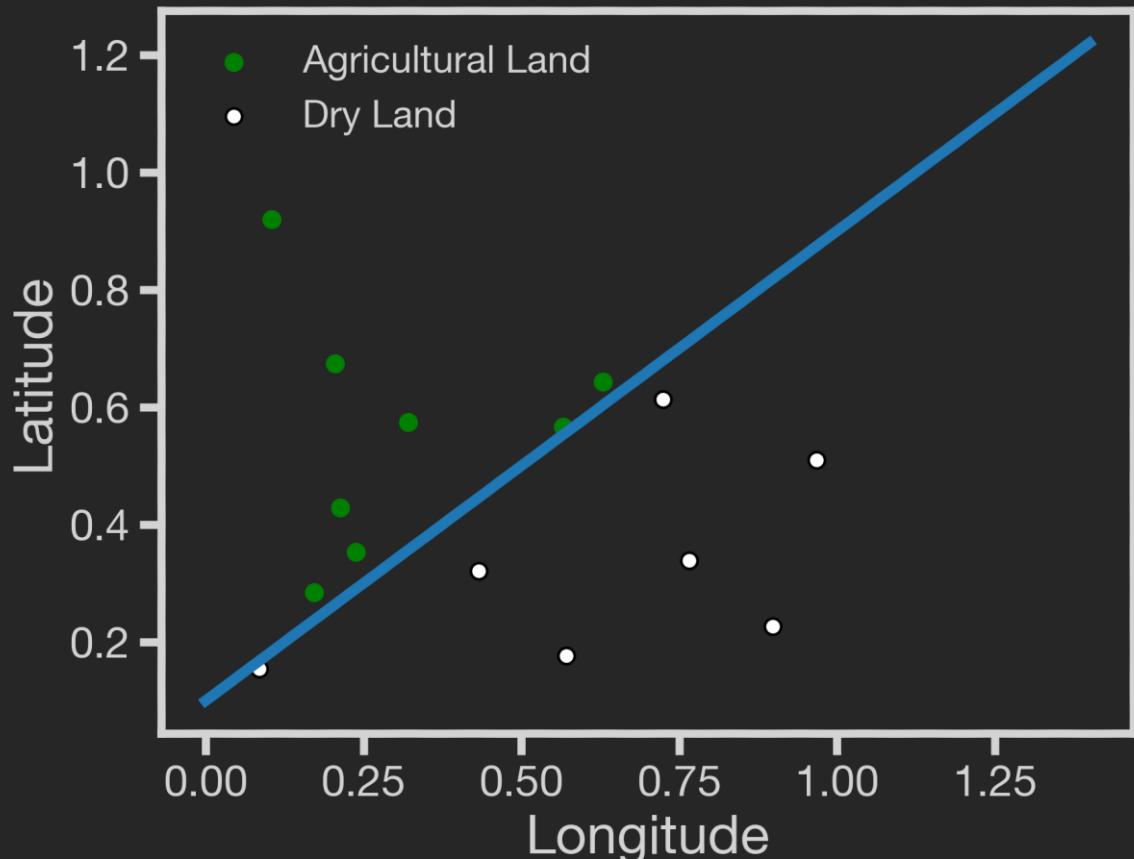
$$X\beta = 0$$

This equation defines a line or a hyperplane.



Motivation

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



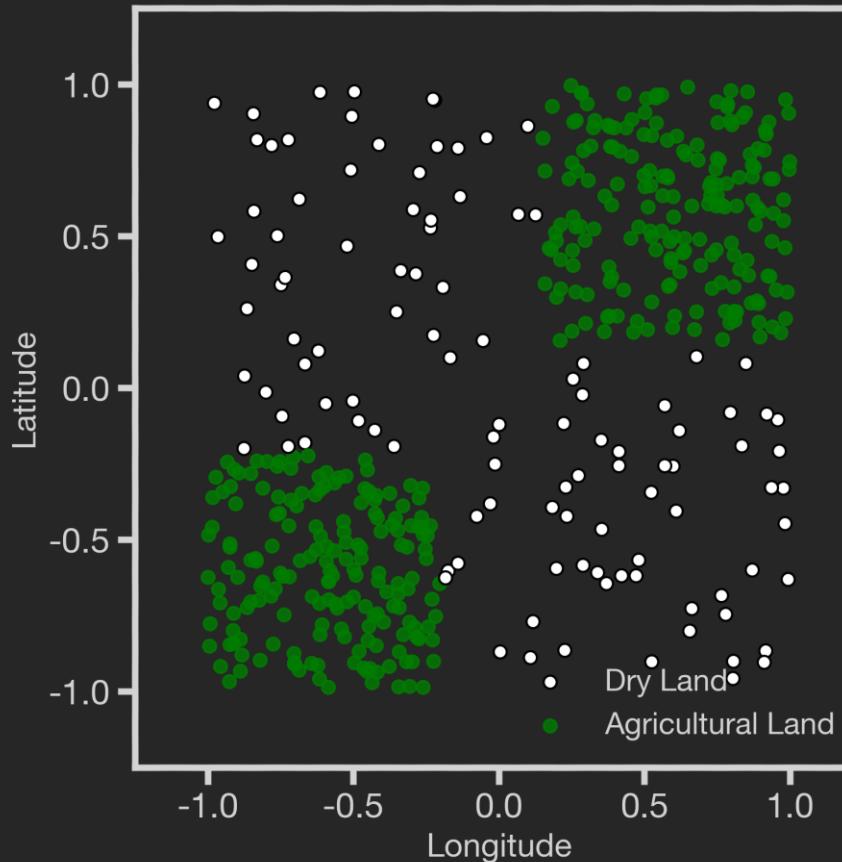
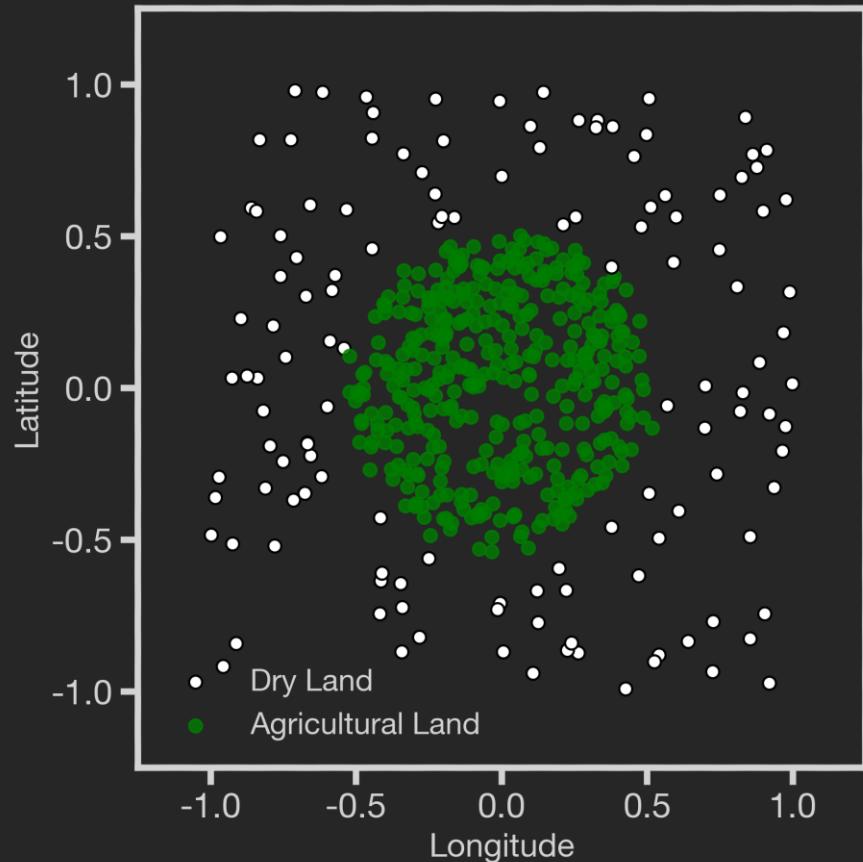
$$\text{Latitude} = 0.8 \text{ Longitude} + 0.1$$

or

$$-0.8\text{Longitude} + \text{Latitude} - 0.1 = 0$$

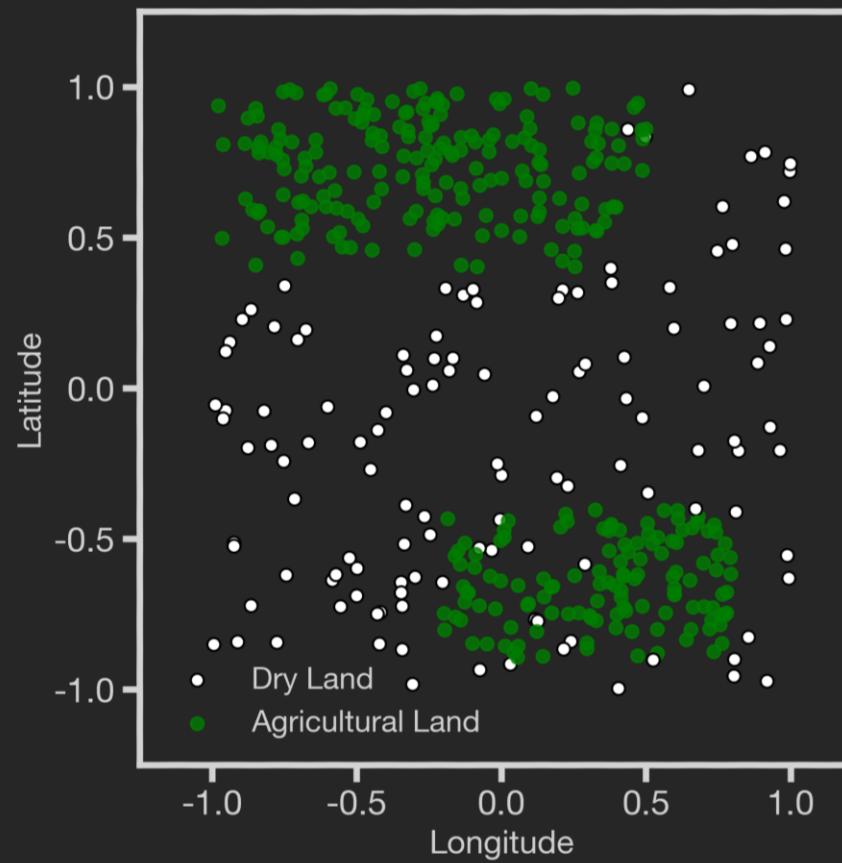
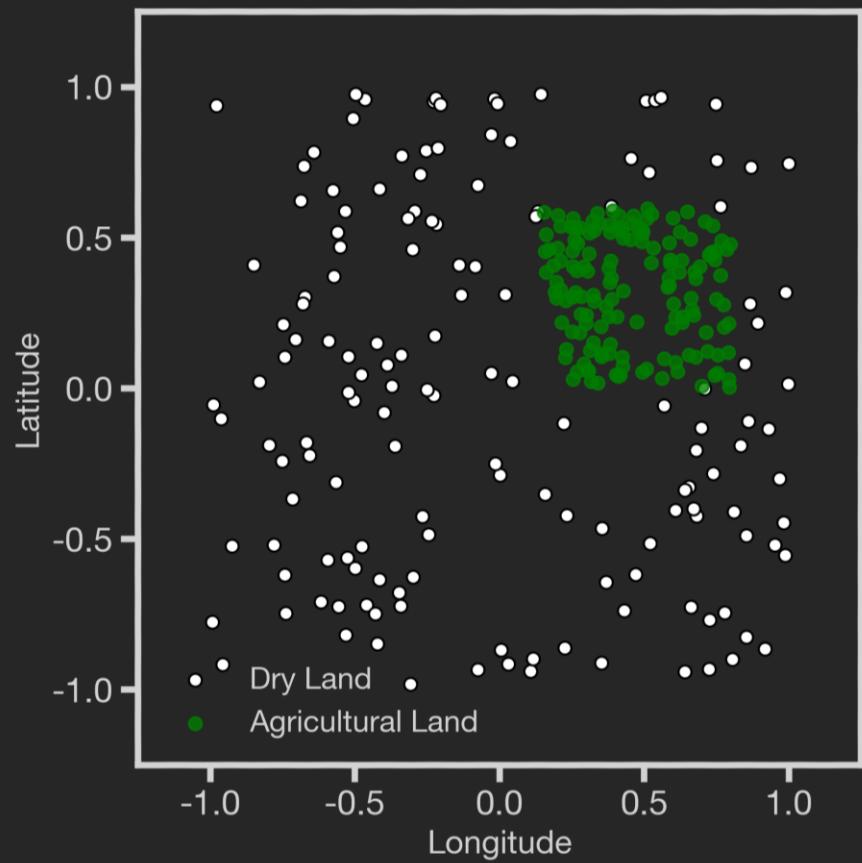
Motivation

This works well when the classification boundary has a nice simple geometry, but what about these?



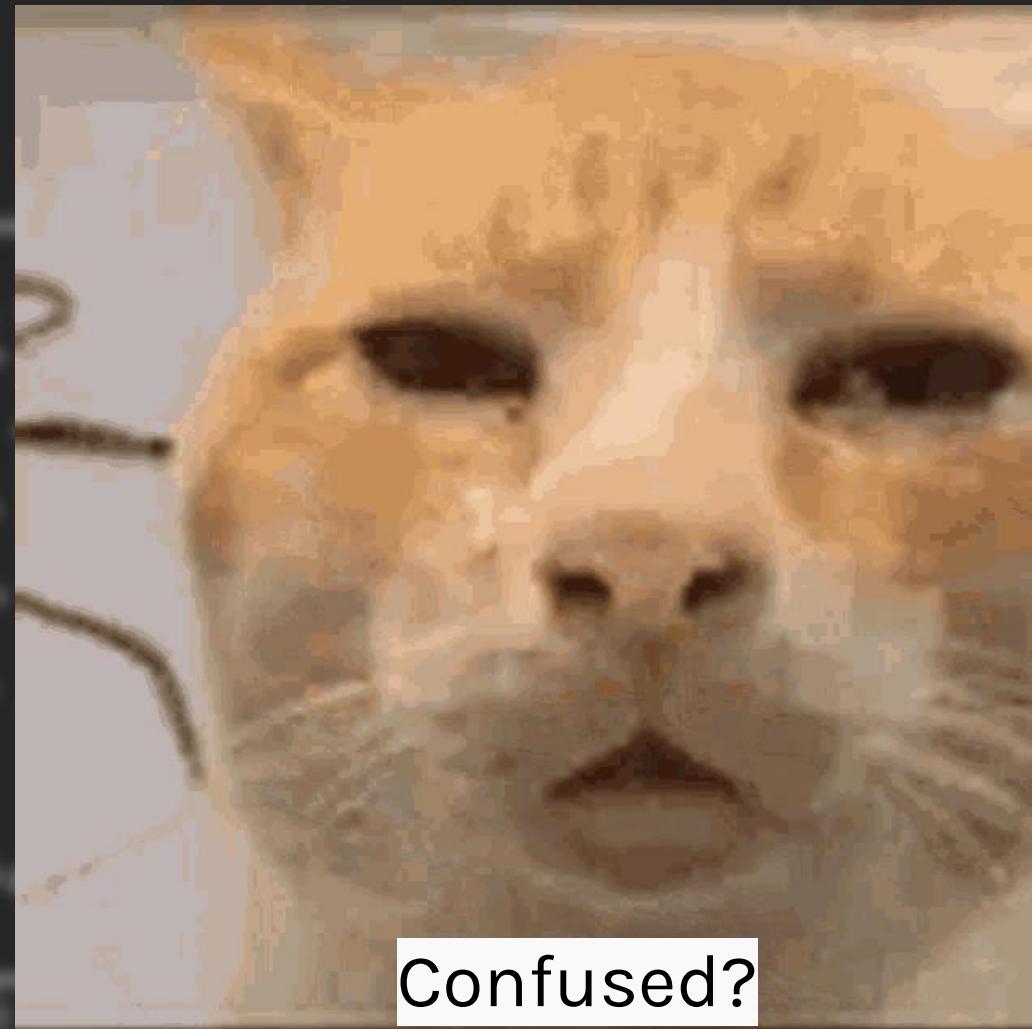
Motivation

Or these?



Motivation

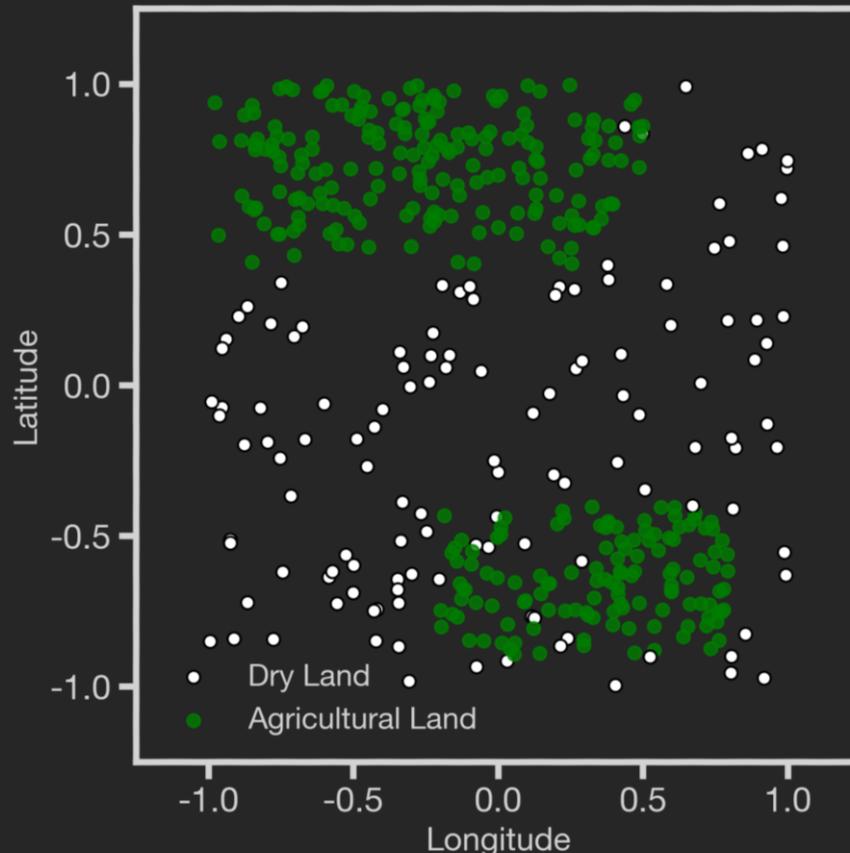
Or these?



Confused?

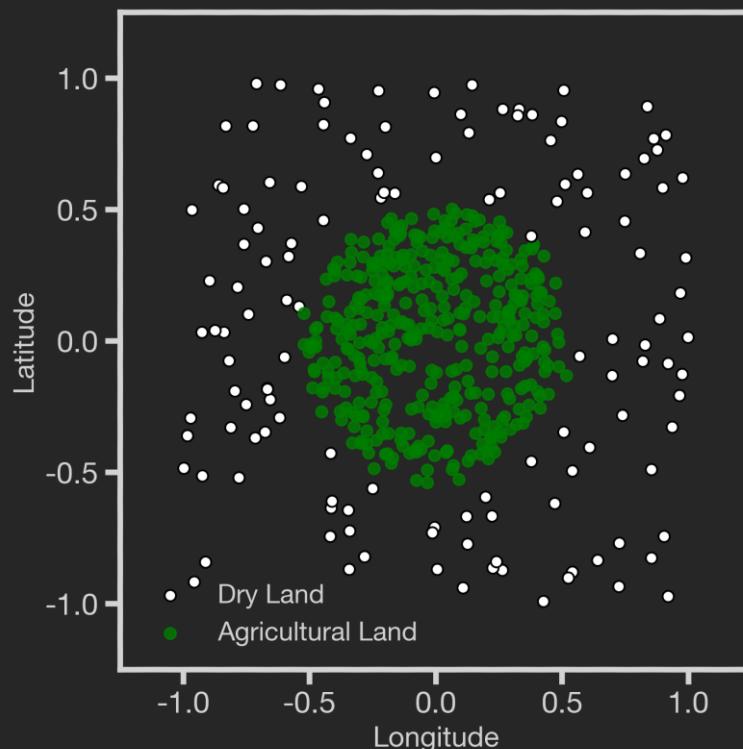
Motivation

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.*



Motivation

Logistic regression models with linear boundaries are intuitive to interpret by examining the impact of each predictor on the log-odds of a positive classification. It is less straightforward to interpret nonlinear decision boundaries in context.



$$x_1^2 + x_2^2 - 0.25 = 0$$

Motivation

The modeling wish list:

We would like to have models
that

1. Allow for complex decision boundaries, and
2. Are also easy to interpret.

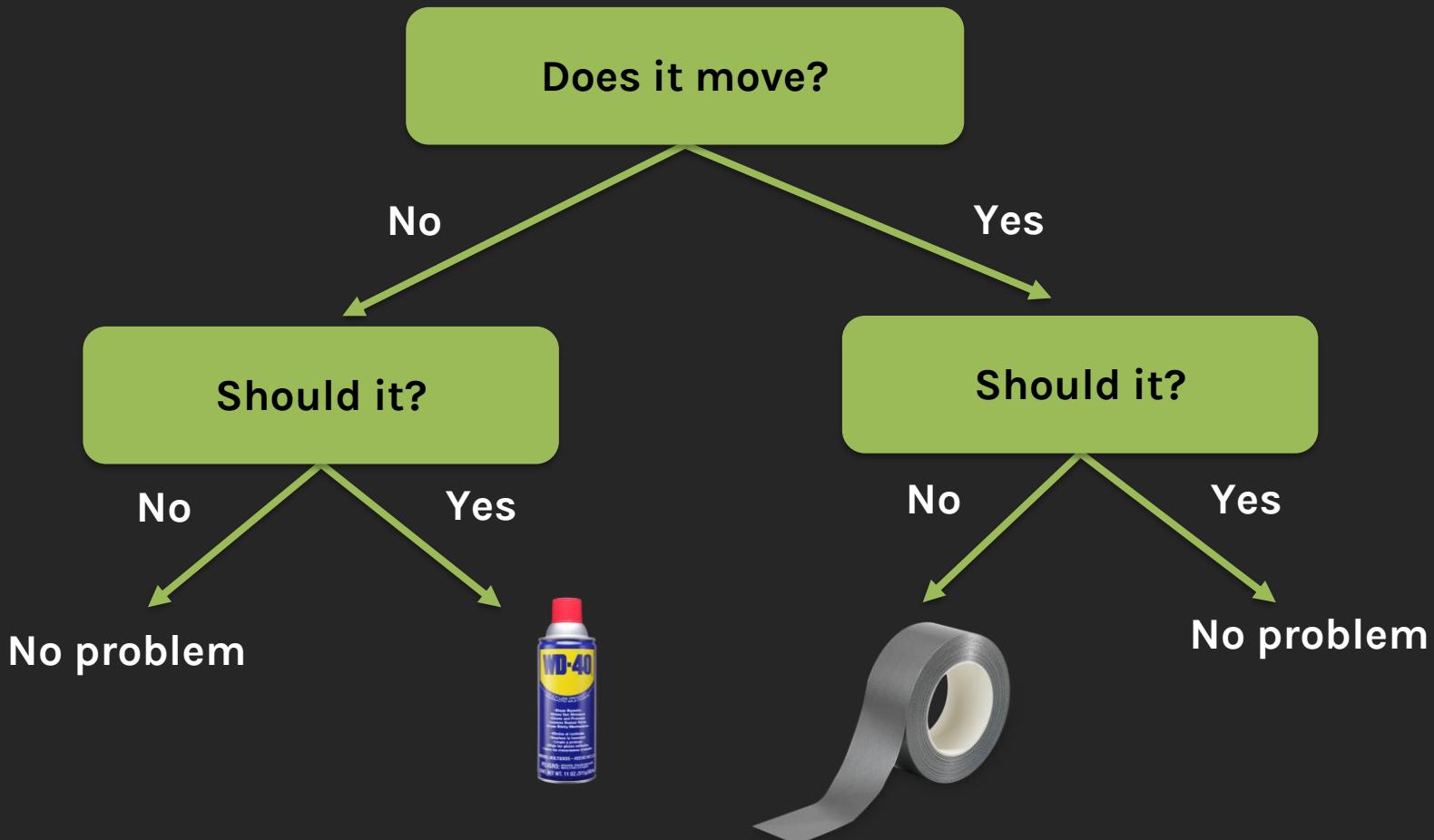


Interpretable Models

People in every walk of life have long been using interpretable models for differentiating between classes of objects and phenomena.

For example, the Engineering Flowchart:

Engineering Flowchart



Interpretable Models

It turns out that the **simple flow chart** in our example can be formulated as a mathematical model for classification, and this model has the properties we desire:

- The model is **interpretable** by humans.
- It has **sufficiently complex** decision boundaries.
- The decision boundaries are **locally linear**, which means each component of the decision boundary is simple to describe mathematically.

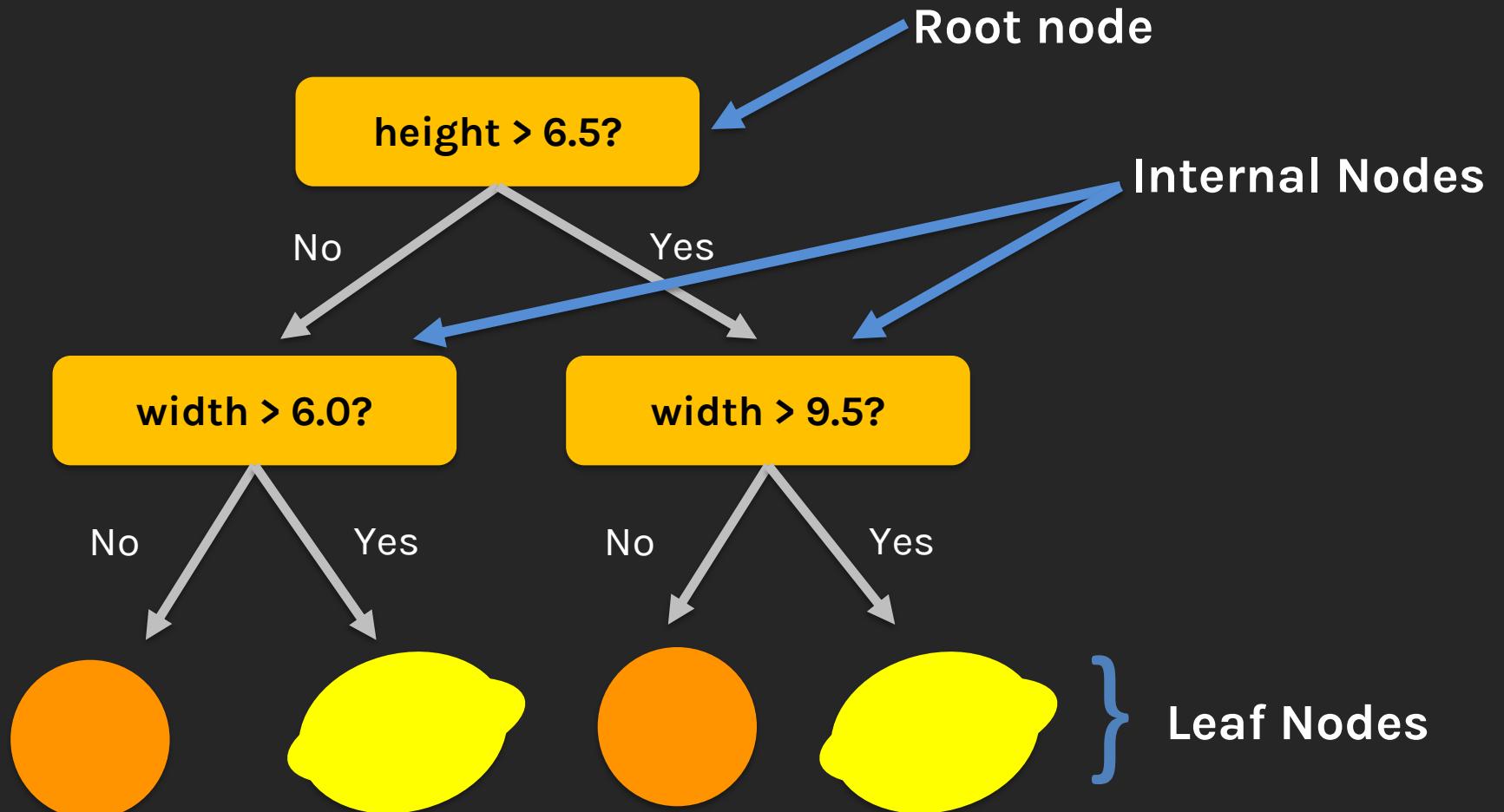


Outline

- Motivation
- **Decision Trees – Classification**
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions
- Decision Trees - Regression

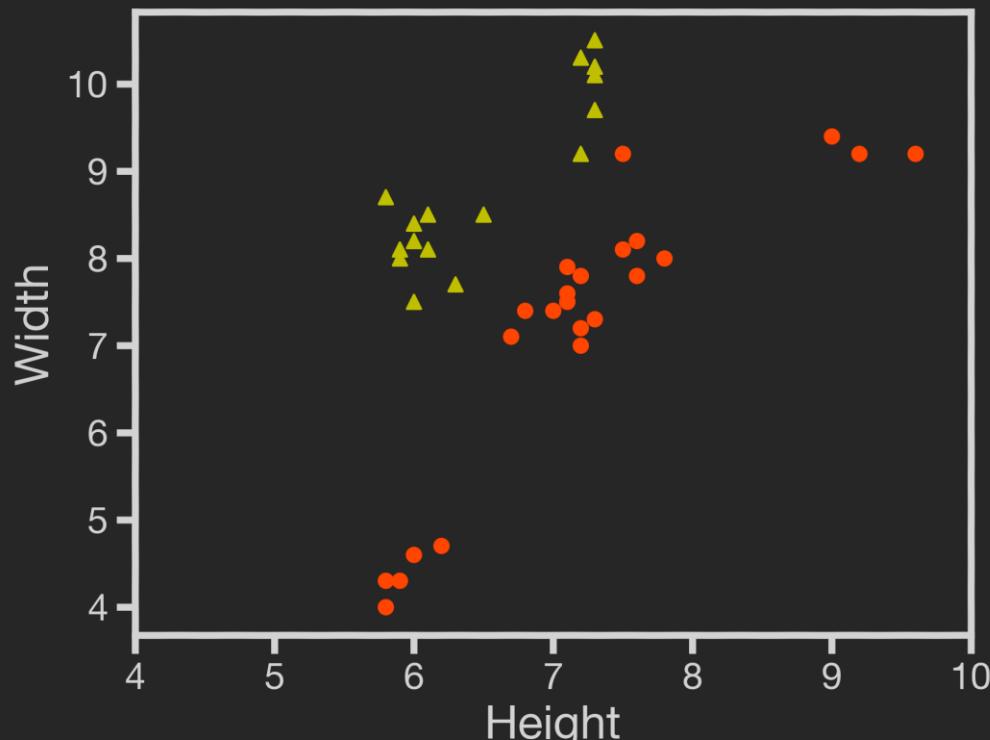
Example

A decision tree to classify fruits into lemons and oranges:



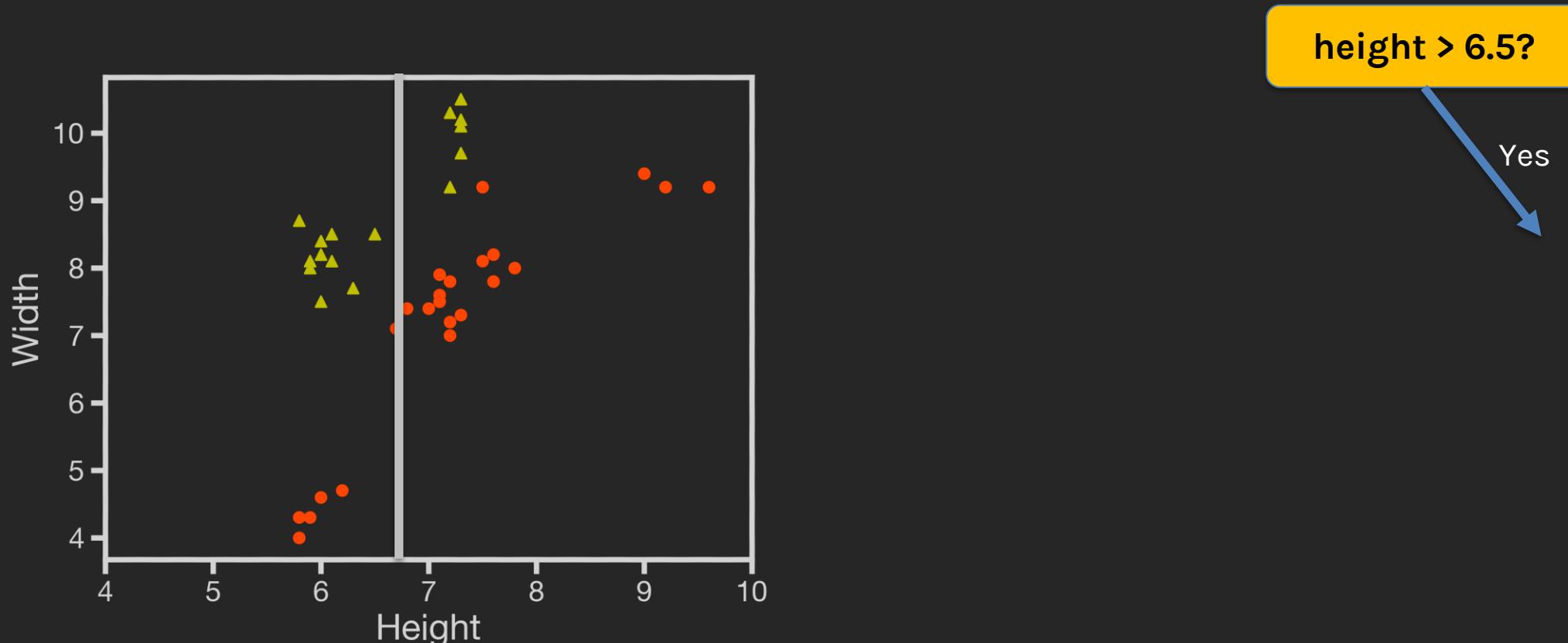
The Geometry of Flow Charts

Every **branch** of the flow chart tree corresponds to a partition of the feature space by **lines parallel to the axis or (hyper) planes**. Conversely, every such partition can be written as a **branch** of the flow chart tree.



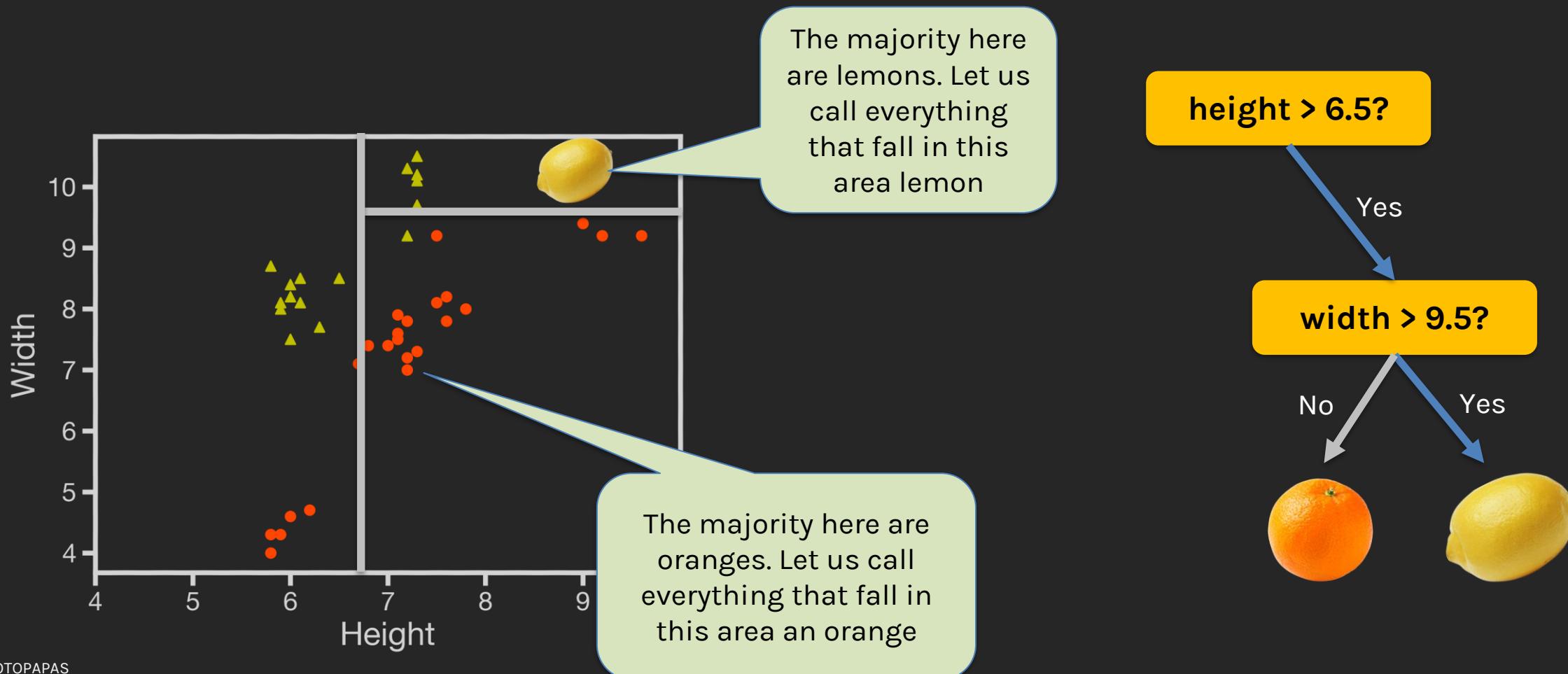
The Geometry of Flow Charts

Every **branch** of the flow chart tree corresponds to a partition of the feature space by **lines parallel to the axis or (hyper) planes**. Conversely, every such partition can be written as a **branch** of the flow chart tree.

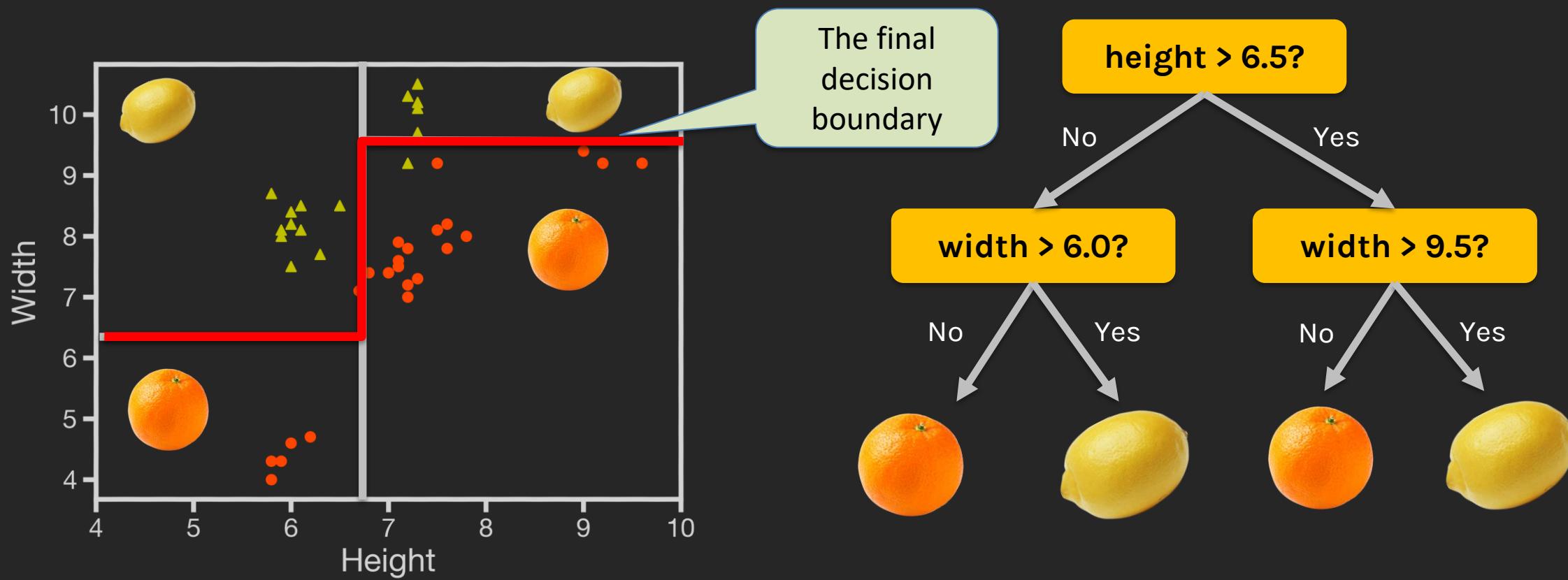


The Geometry of Flow Charts

Every **branch** of the flow chart tree corresponds to a partition of the feature space by **lines parallel to the axis or (hyper) planes**. Conversely, every such partition can be written as a **branch** of the flow chart tree.



The Geometry of Flow Charts

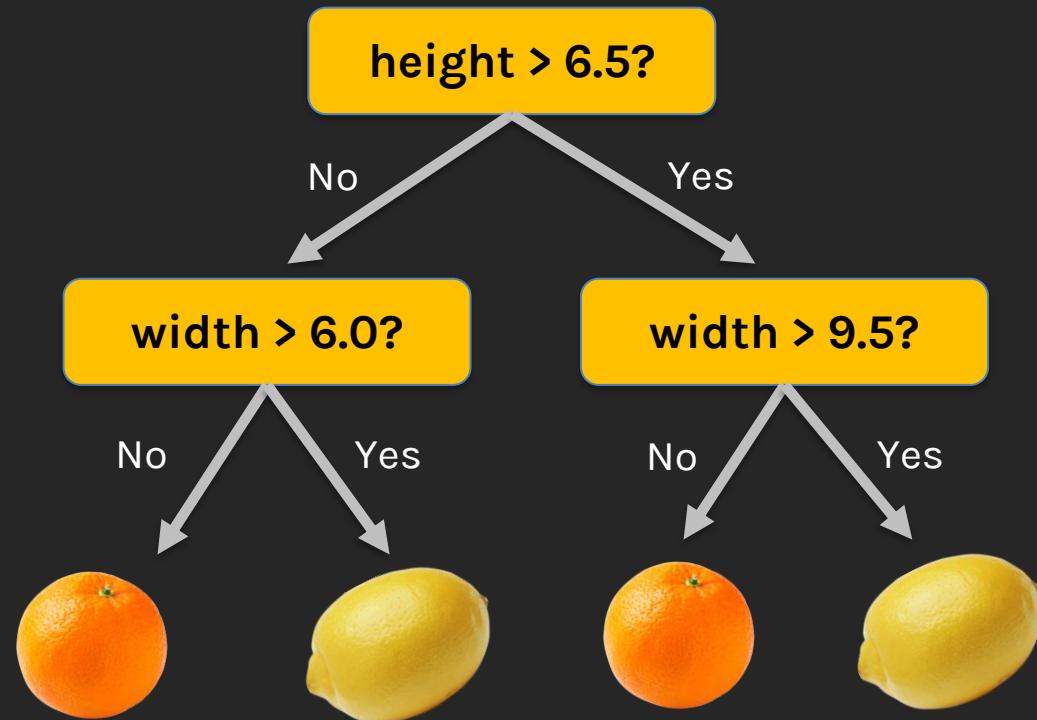


Outline

- Motivation
- **Decision Trees – Classification**
 - Intuition
 - **Predictions**
 - Splitting Criteria
 - Stopping Conditions
- Decision Trees - Regression

Predictions

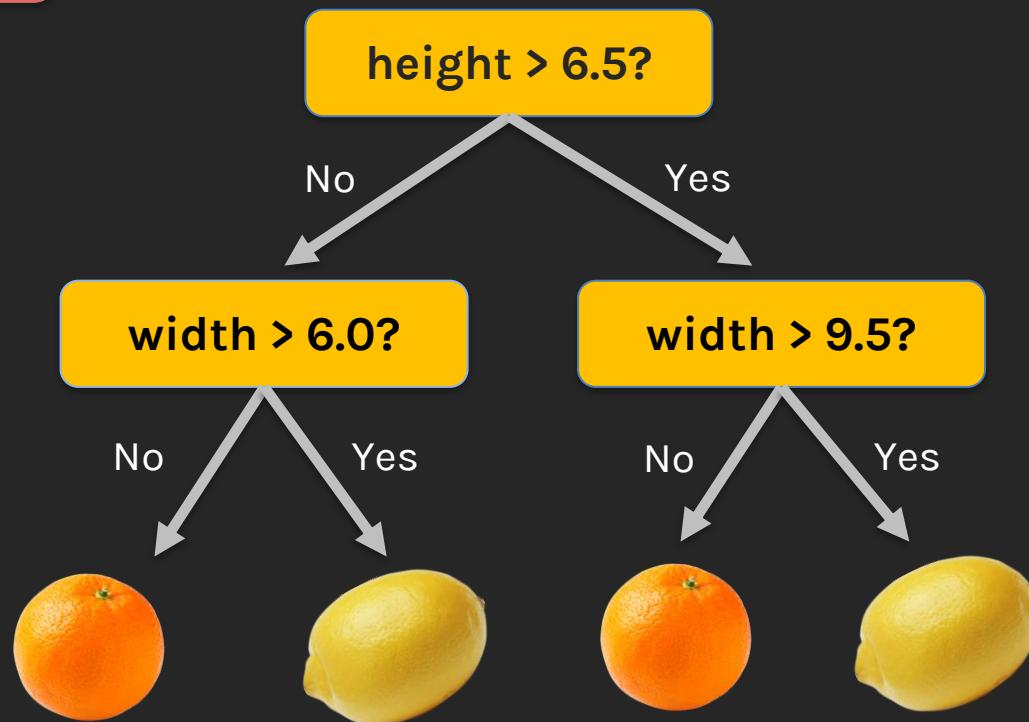
Let's use the decision tree from before to make predictions.



Predictions

?

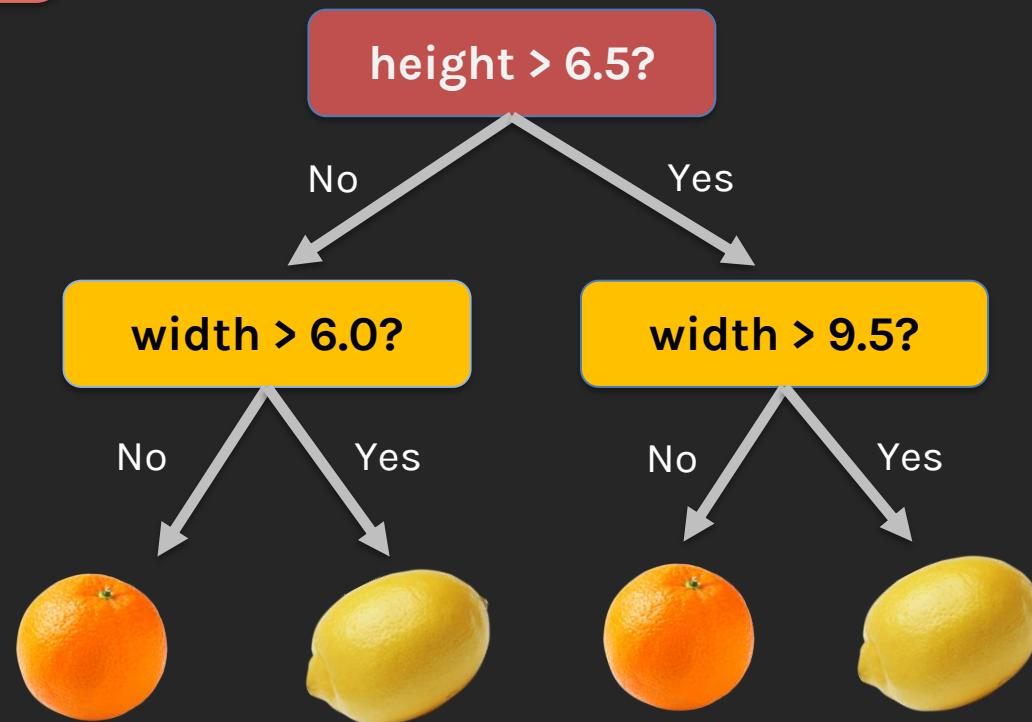
height = 5.9
width = 5.8



Predictions

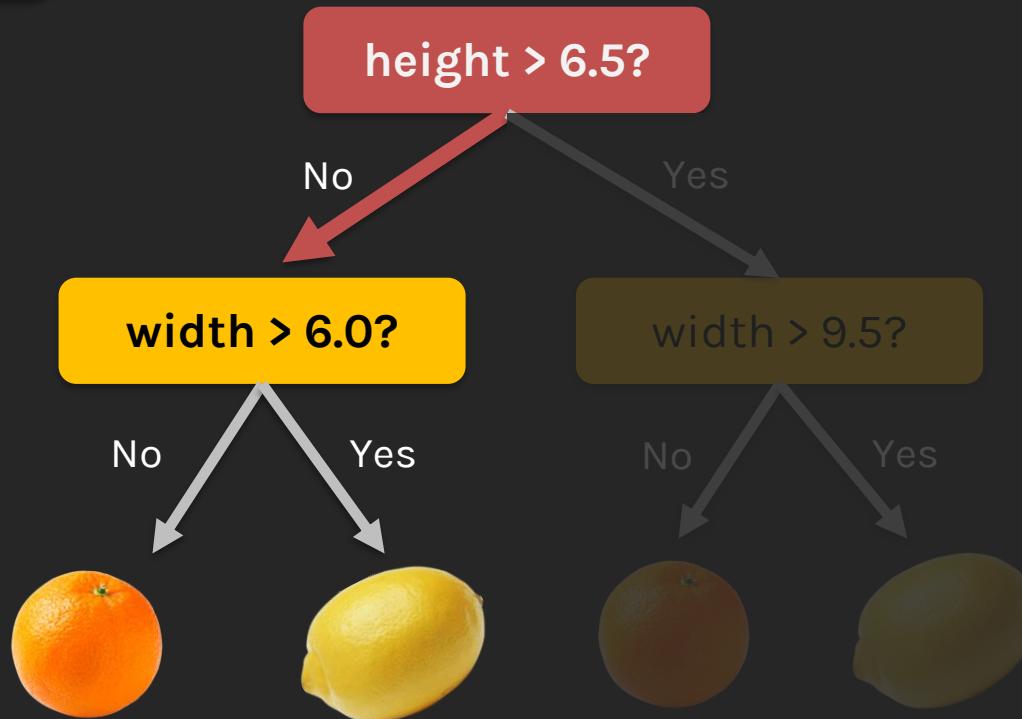
?

height = 5.9
width = 5.8



Predictions

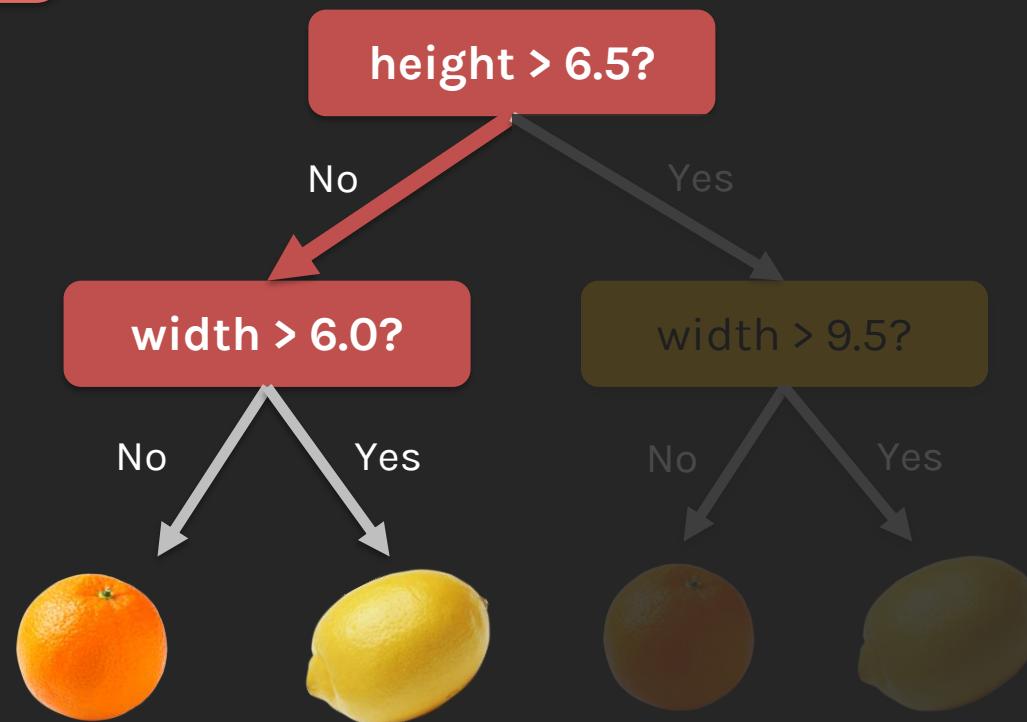
? height = 5.9
 width = 5.8



Predictions

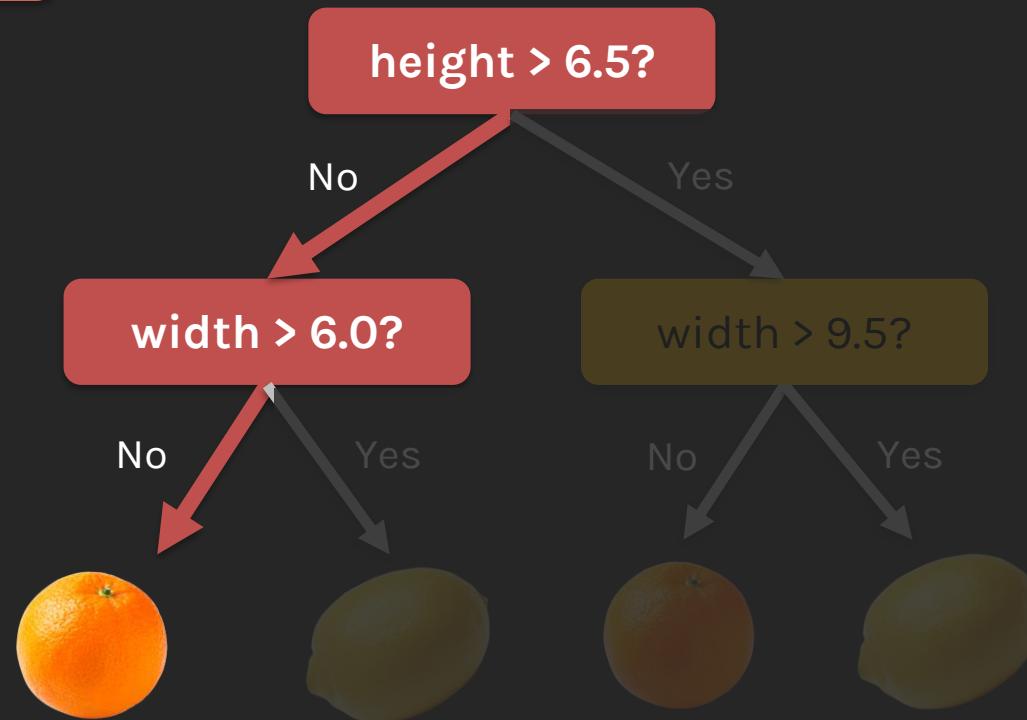
?

height = 5.9
width = 5.8



Predictions

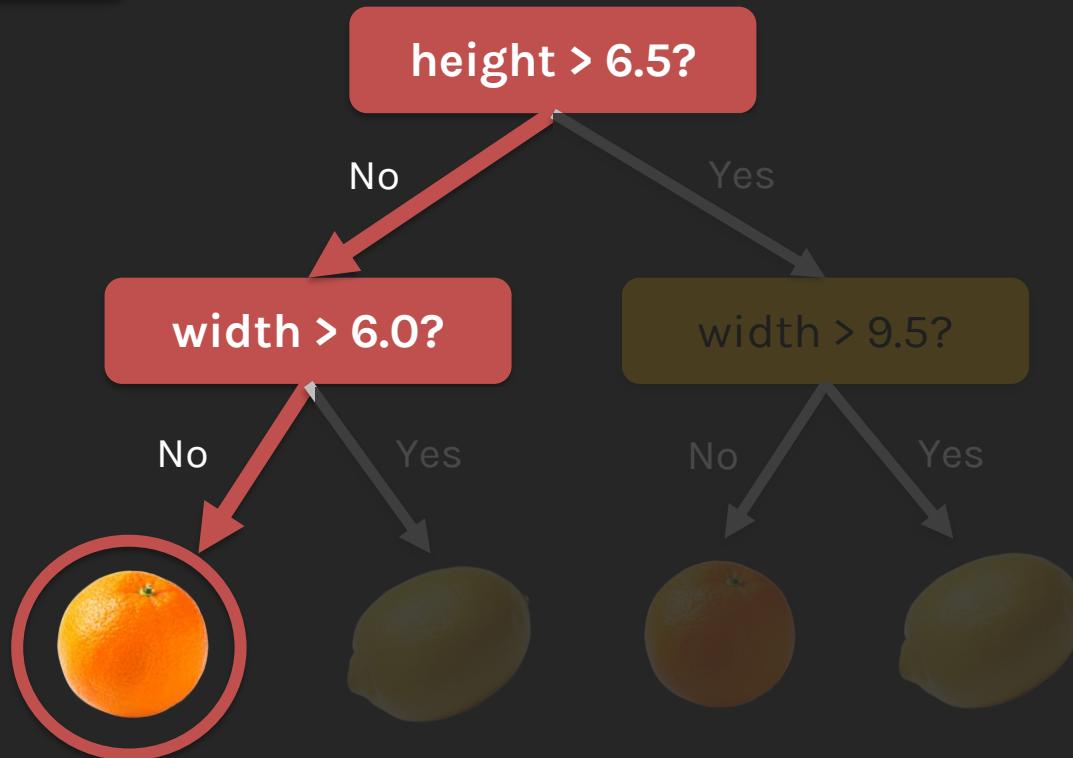
? height = 5.9
 width = 5.8



Predictions



**height = 5.9
width = 5.8**

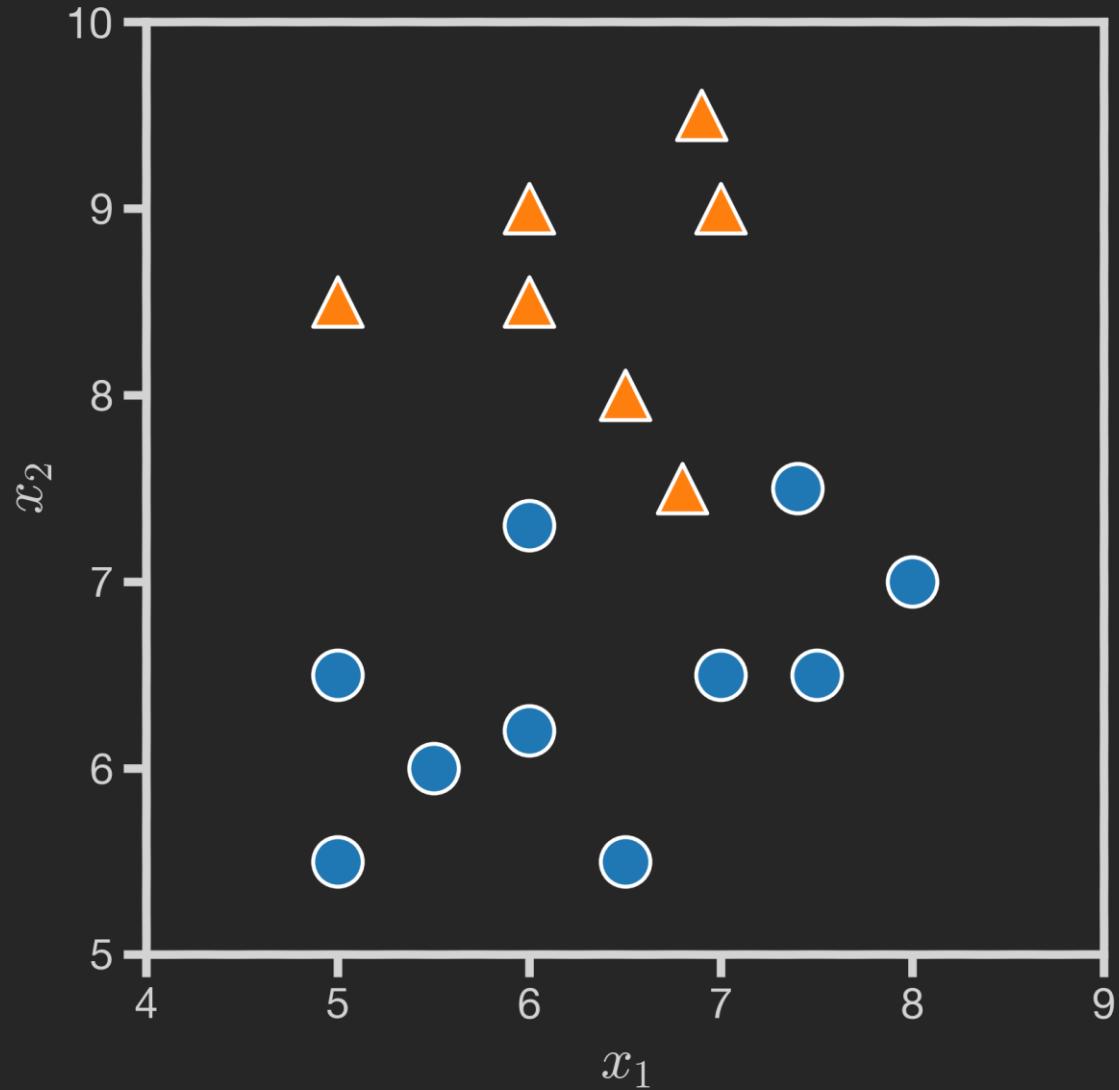


THIS METHOD IS ALSO CALLED TRAVERSING THE TREE.

Outline

- Motivation
- **Decision Trees – Classification**
 - Intuition
 - Predictions
 - **Splitting Criteria**
 - Stopping Conditions
- Decision Trees - Regression

Splitting Criteria

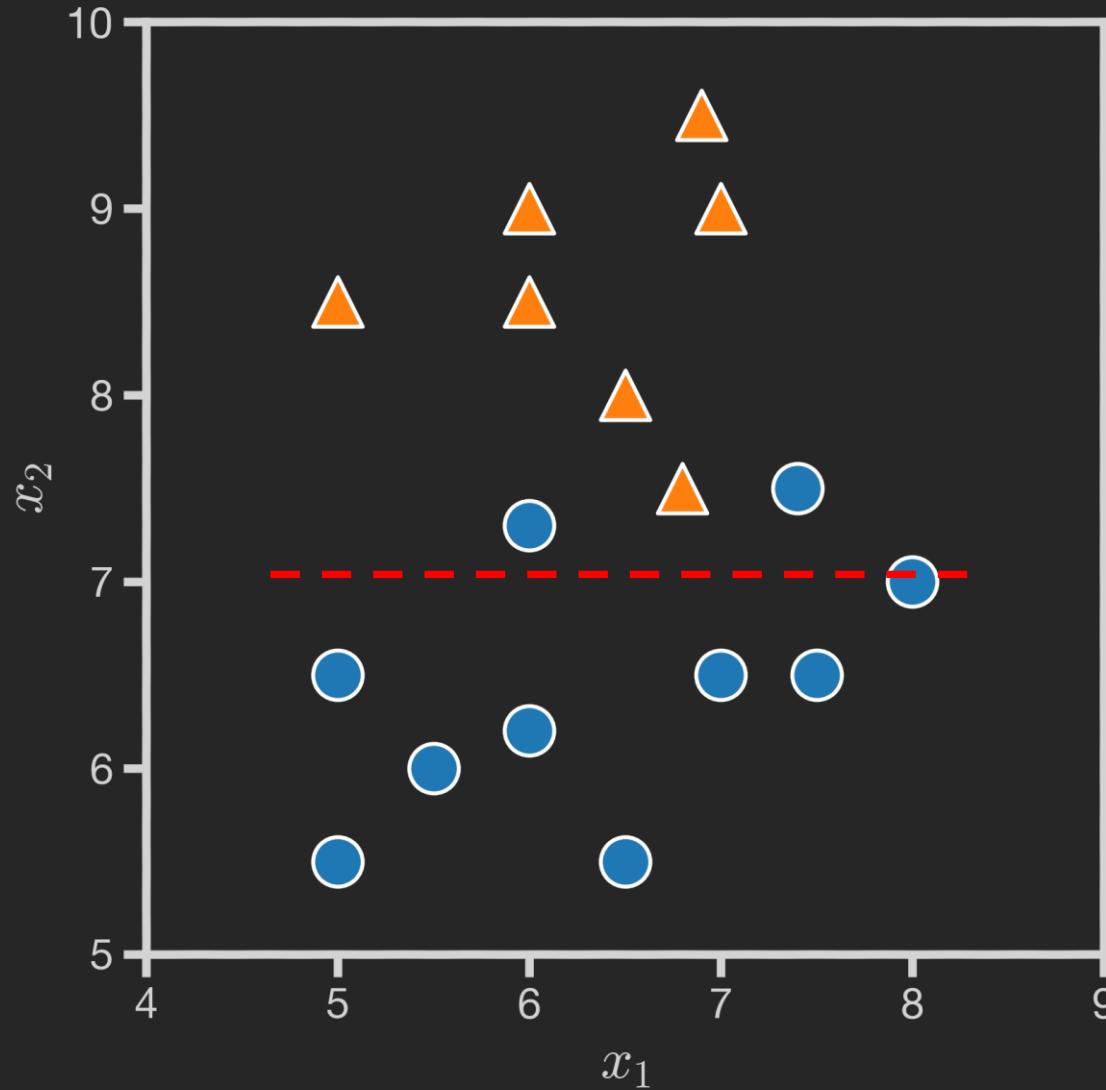


Which gives a better split?

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



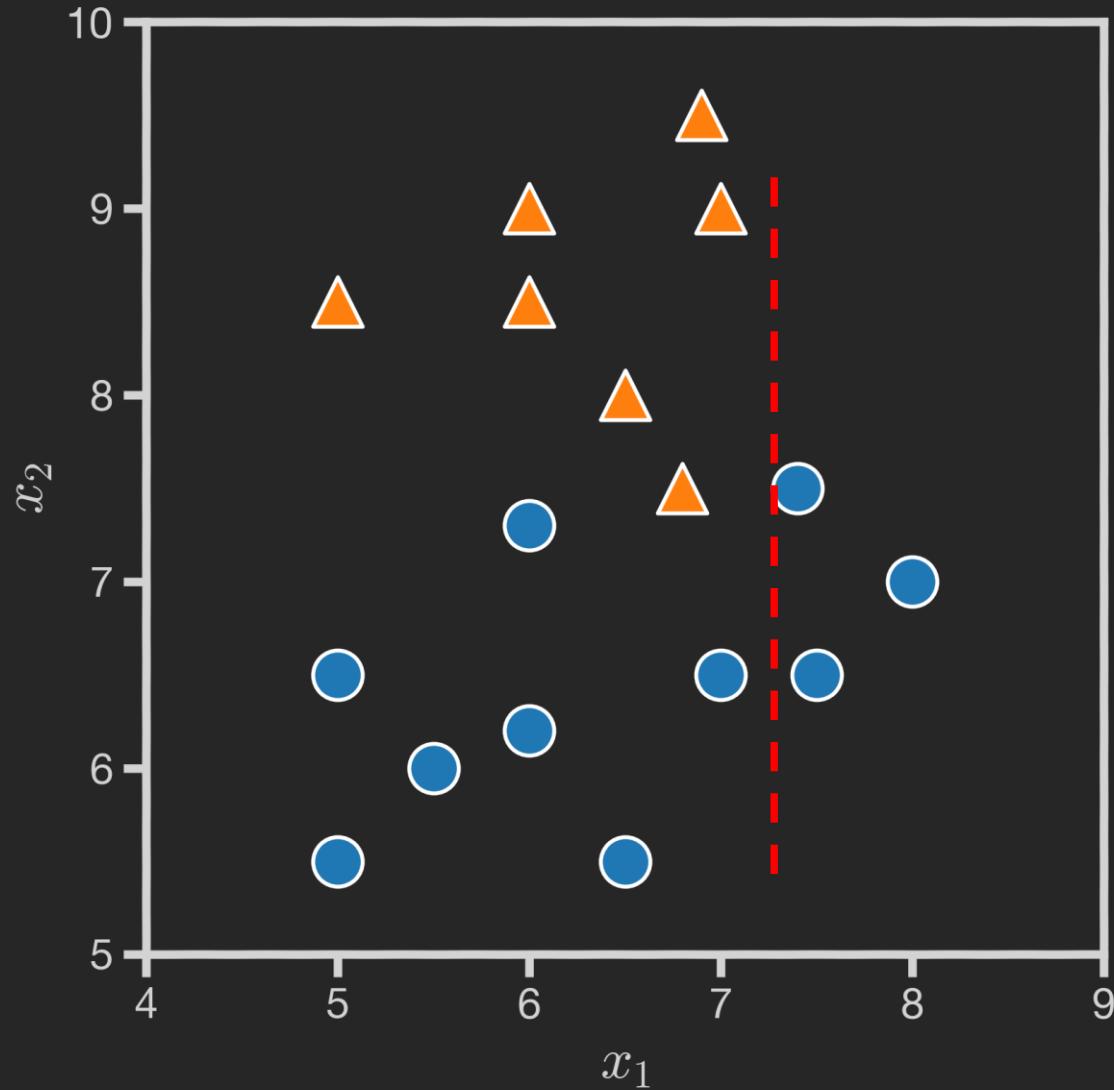
Splitting Criteria



Which gives a better split?

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

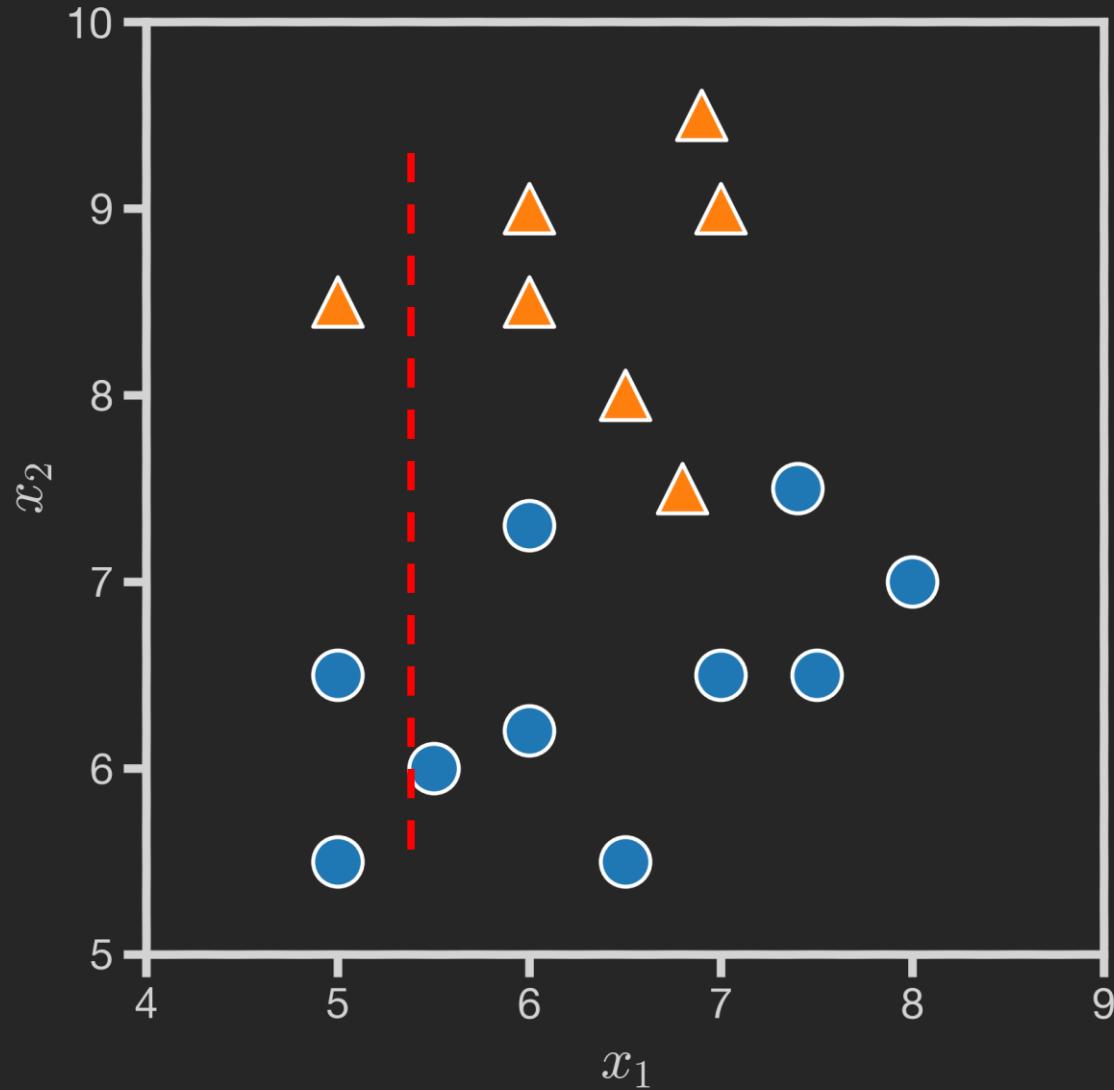
Splitting Criteria



Which gives a better split?

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

Splitting Criteria



Which gives a better split?

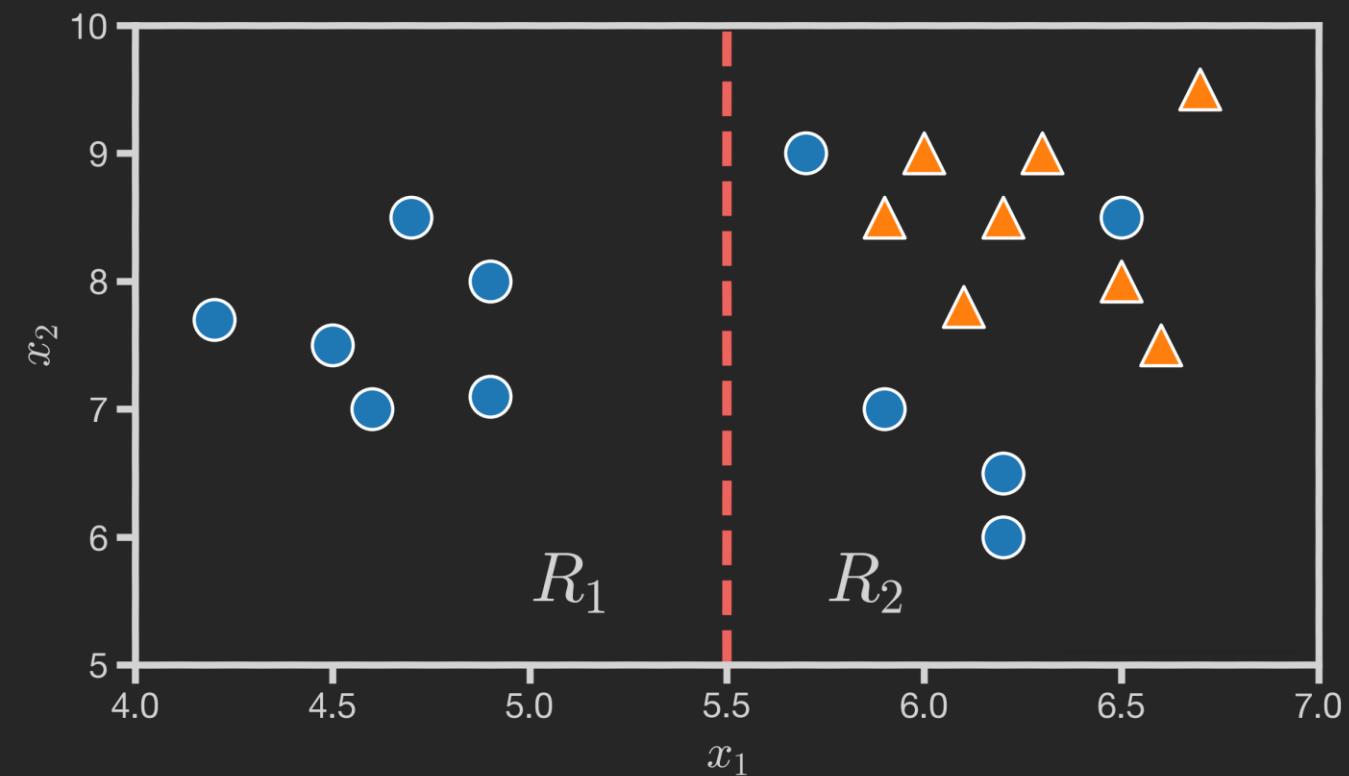
- Should it be split along x_1 or x_2 ?
- Where should we split?

Optimality of Splitting

While there is no ‘correct’ way to define an optimal split, there are some common-sense guidelines for every splitting criterion:

- The regions in the feature space should grow progressively **purer** with the number of splits. That is, we should see that each region ‘specializes’ towards a single class.
- We should end up with **no empty regions** – every region should contain training points.
- The fitness metric of a split should take a **differentiable form** (making optimization possible).

Classification Error



Consider region R_2 :

$$\begin{aligned}\text{Classification error} &= \frac{\text{error}}{\text{total}} \\ &= \frac{\text{Number of minority class data points}}{\text{Total number of data points}} \\ &= \frac{\text{Number of } \bullet}{\text{Total number of data points}} \\ &= 1 - \frac{\text{Number of majority data points}}{\text{Total number of data points}}\end{aligned}$$

Classification Error

Classification error in region R_2 :

$$= 1 - \frac{\text{Number of majority data points}}{\text{Total number of data points}}$$

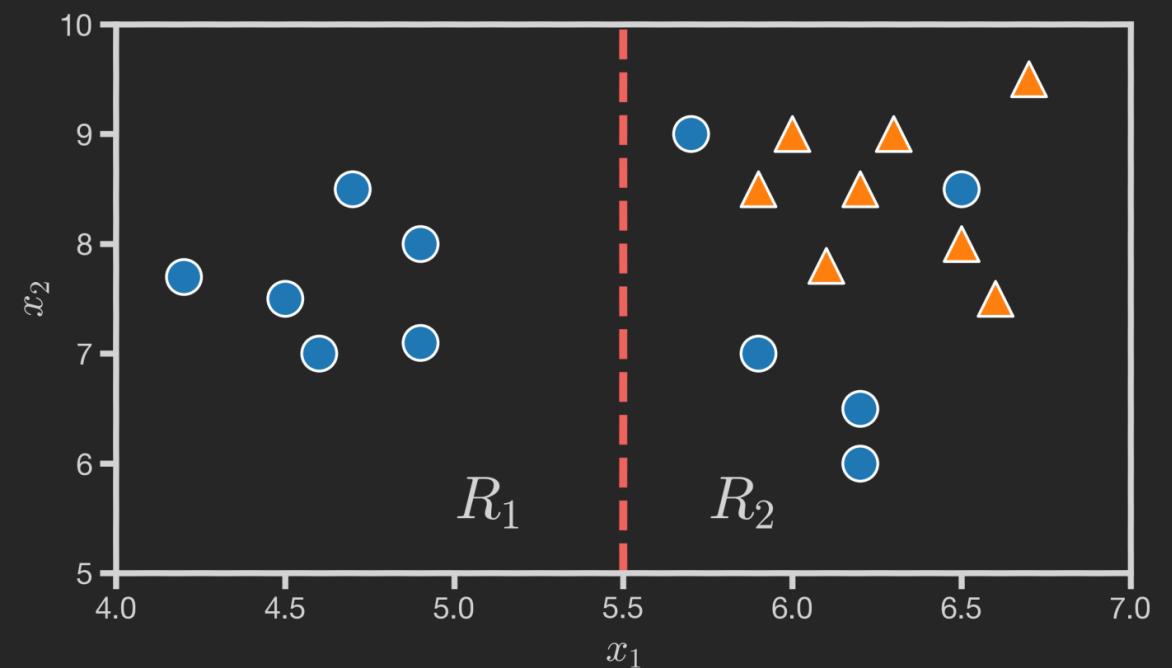
$$= 1 - \Psi(\blacktriangle | R_2)$$

$$= 1 - \max_k(\Psi(k|R_r))$$

$\Psi(k|R_r)$ is the proportion of training points in R_2 that are labeled class k.

We take \max_k here because generally we will not know what the majority class is.

Classification Error



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

$$1 - \max\left(\frac{6}{6}, \frac{0}{6}\right) = 0$$

$$1 - \max\left(\frac{5}{13}, \frac{8}{13}\right) = \frac{5}{13} = 0.38$$

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** .

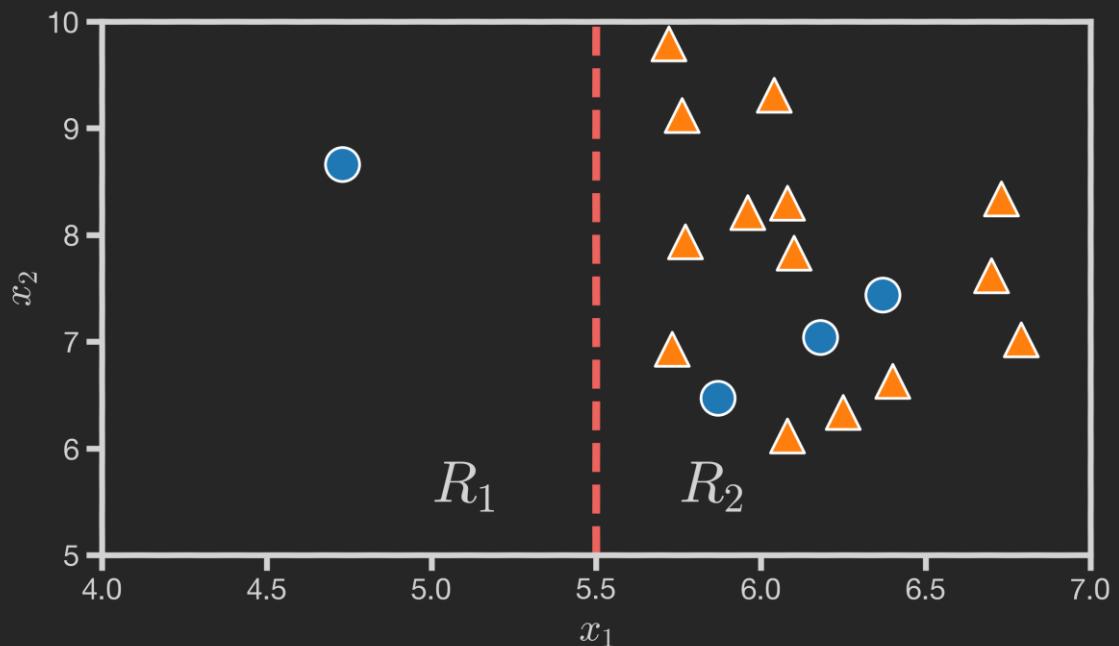
We can assess the quality of this split by calculating the **classification error** made by each newly created region:

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

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

Classification Error

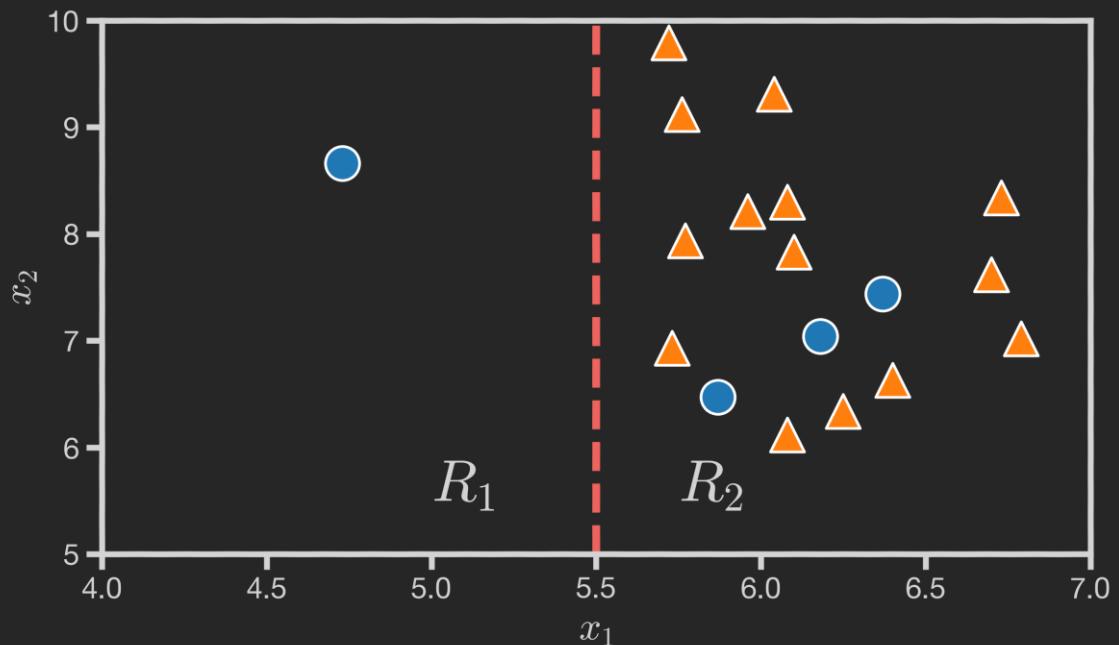
Now calculate the error for the following split:



			$Error = 1 - \max_k(\Psi(k R_r))$
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 than 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:

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

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 ratio of points in the k -th class and the total number of points in the region r (PSI)

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

Question: What is the effect of squaring the proportions of each class?



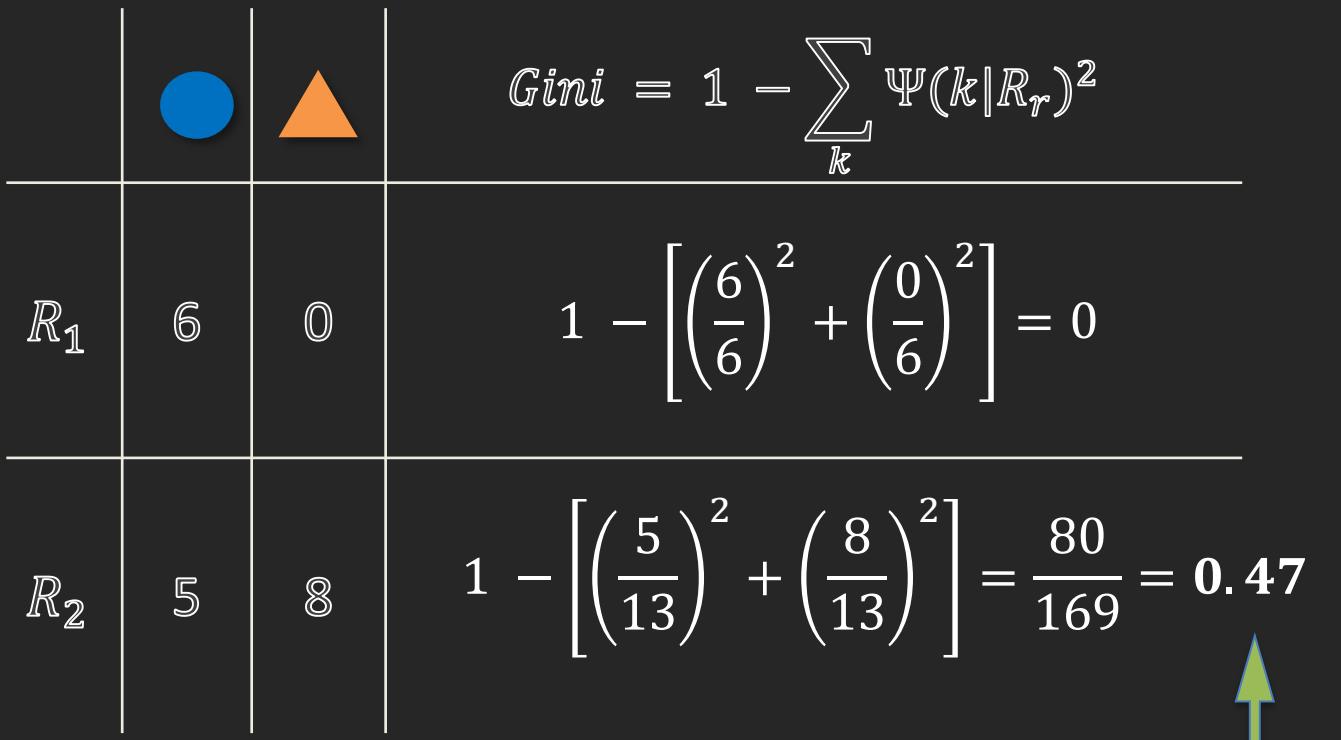
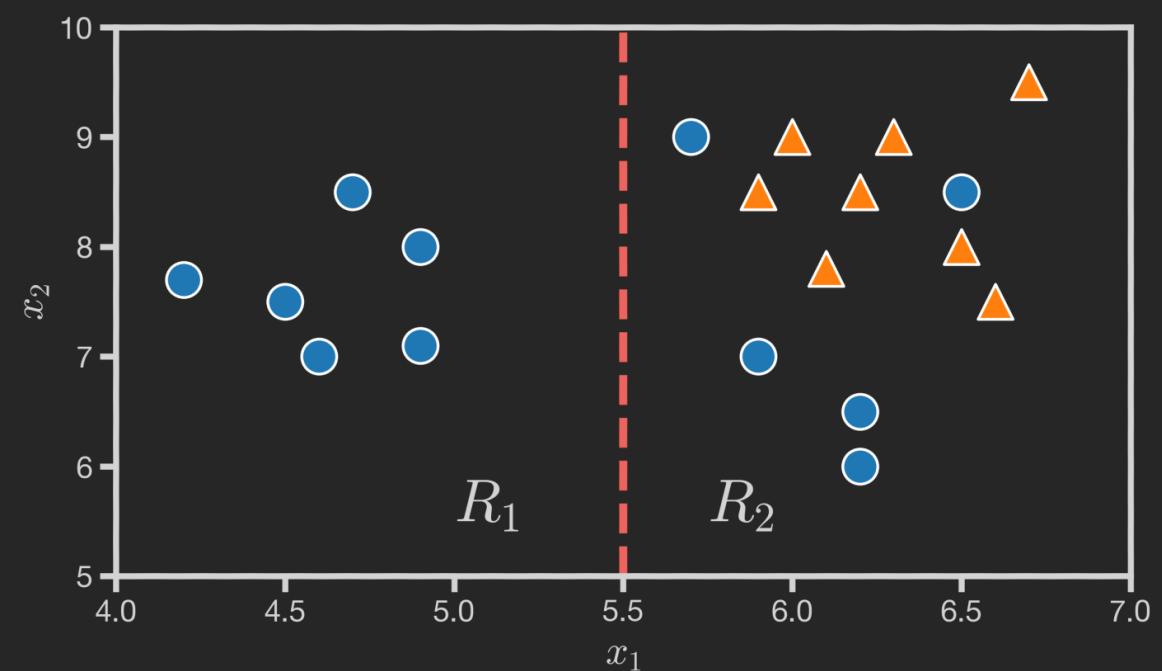
Quiz time

Which of the following statements best describes the effect of squaring the proportions of each class when calculating the Gini Index?

Options:

- A) Squaring the proportions magnifies the influence of majority classes in a region, thereby overstating their significance.
- B) The squaring process accentuates the contrast between pure regions and mixed ones, enabling the index to better differentiate between quality splits.
- C) The act of squaring diminishes the index's sensitivity to minor fluctuations in class distributions, making it more stable.
- D) Squaring the proportions directly correlates the Gini Index with the count of predictors P , biasing the selection of split predictors.

Gini Index



Compared to 0.38 when we used **classification error**

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:

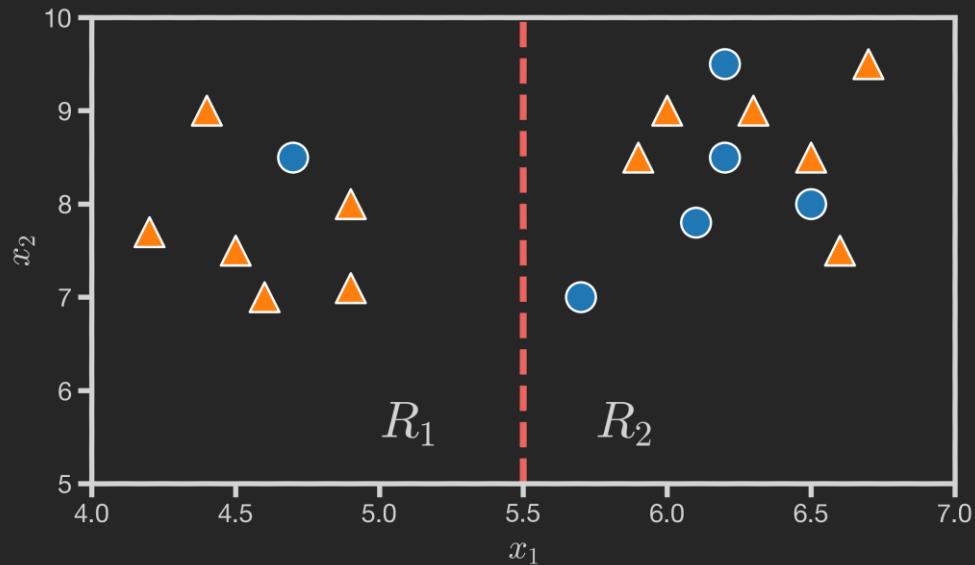
$$\min_{p,t_p} \left[\frac{N_1}{N} Gini(R_1|p, t_p) + \frac{N_2}{N} Gini(R_2|p, t_p) \right]$$

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

Information Theory

The last metric for evaluating the quality of a split is motivated by metrics of uncertainty in information theory.

Question: In the below plot, which region is ‘purer’?



While both regions are impure, R_1 clearly sends a **stronger ‘signal’** for class 2 than R_2 .

Information Theory

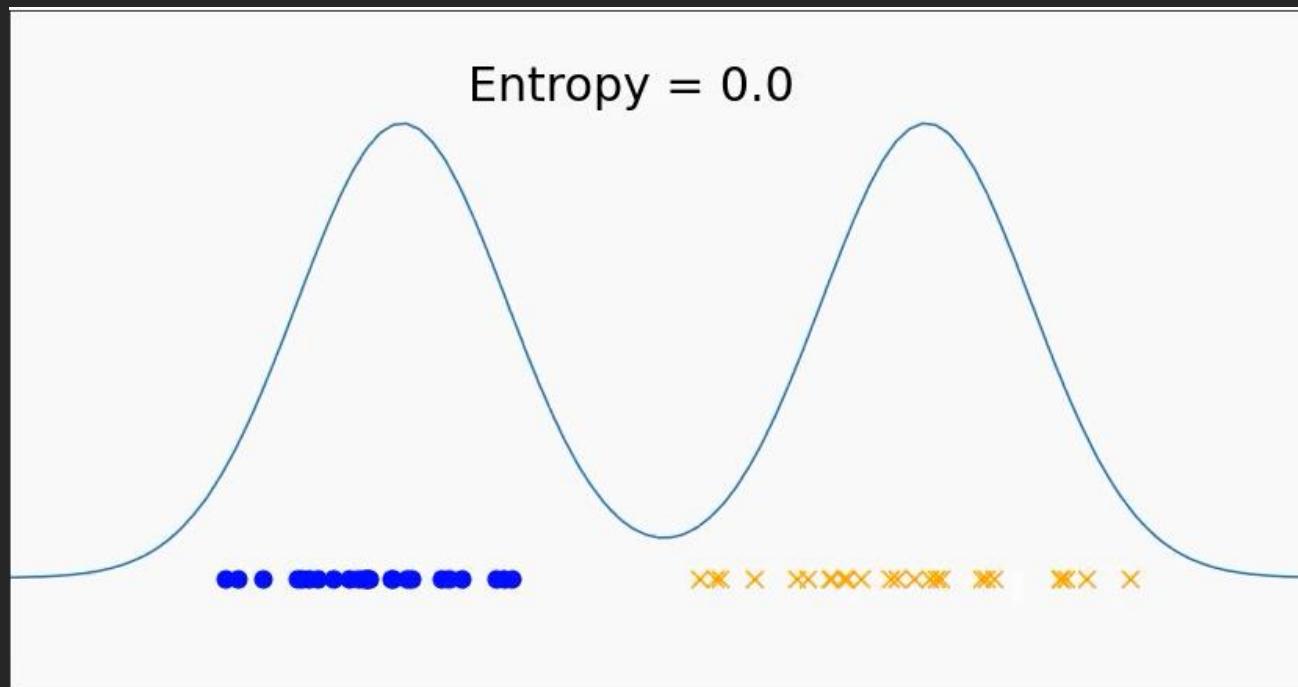
One way to quantify the strength of a signal in a particular region is to analyze the **distribution of classes** within the **region**. We compute the **entropy** of this distribution.

For a random variable with a discrete distribution, the entropy is computed by:

$$H(x) = - \sum_{x \in X} \Psi(x) \log_2 \Psi(x)$$

Information Theory

$$H(x) = - \sum_{x \in X} \Psi(x) \log_2 \Psi(x)$$



Entropy

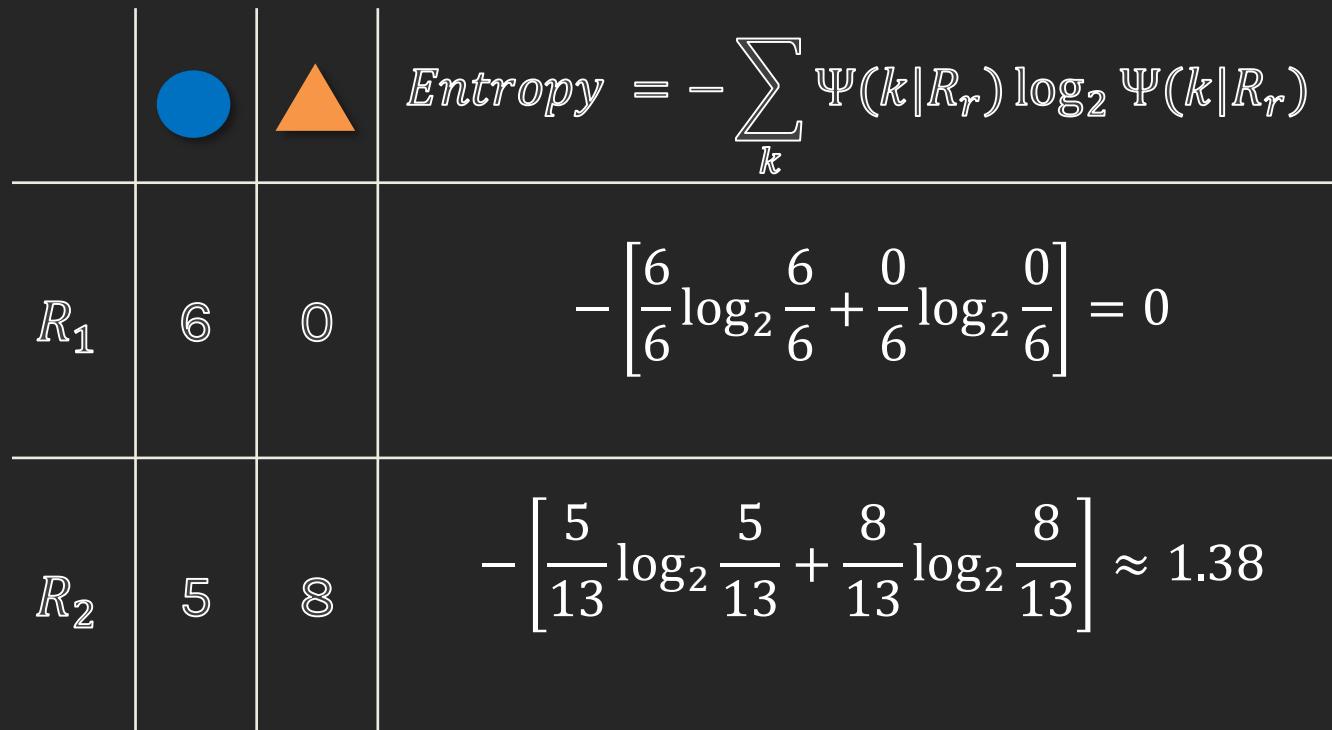
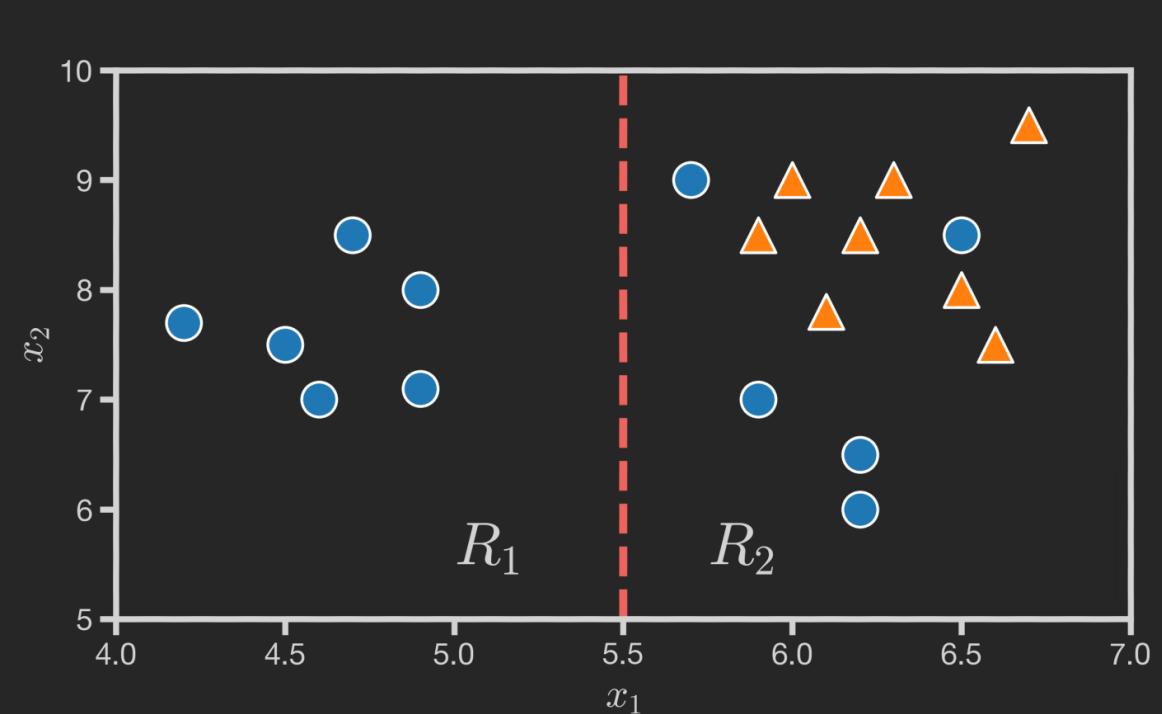
Assume we have **P predictors** and **K classes**. Suppose we select the p^{th} predictor and split a region along the **threshold t_p** .

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

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

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

Error Measurement	Value
Classification Error	0.38
Gini Index	0.47
Entropy	1.38

Entropy

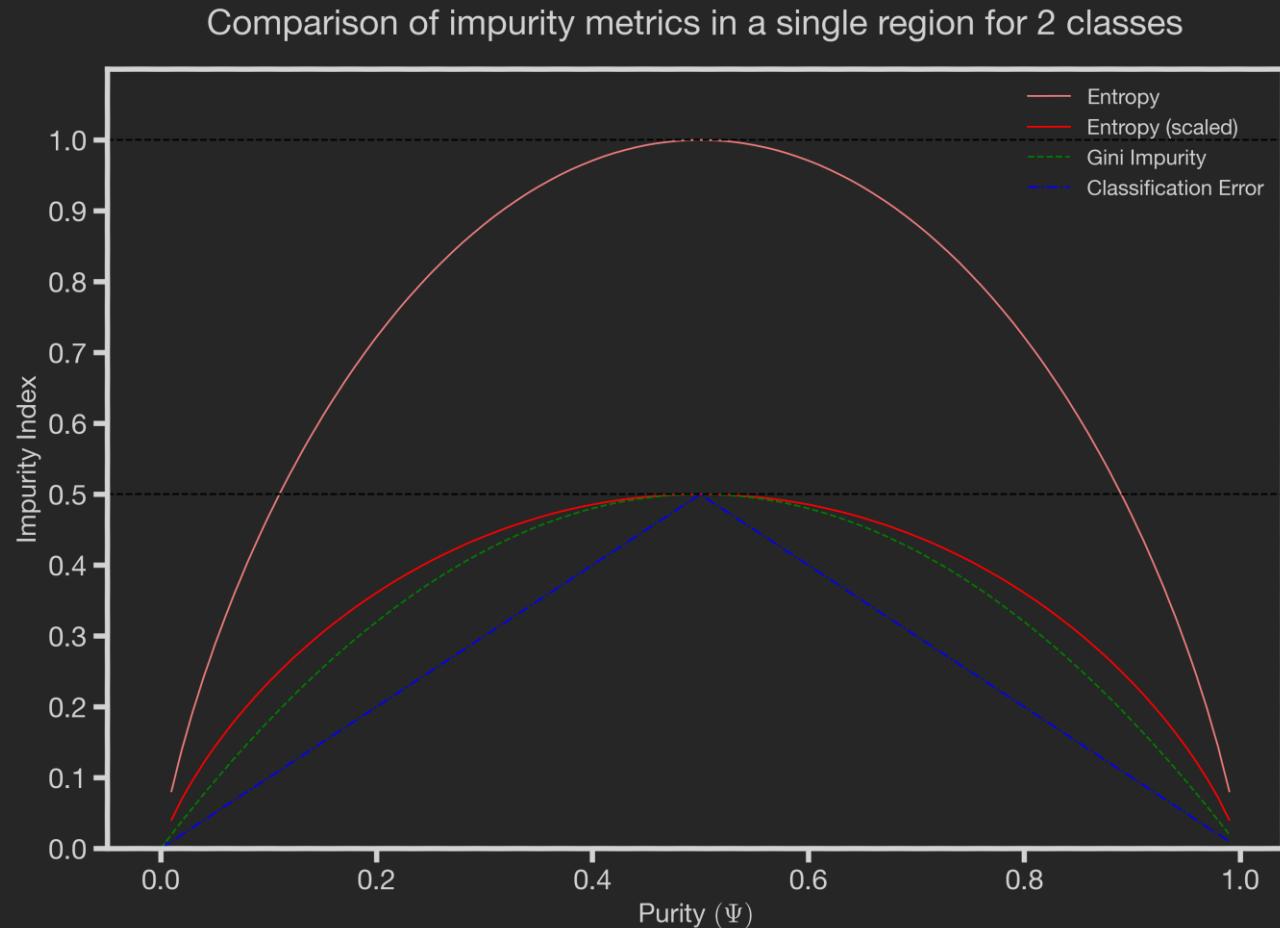
We can now try to find the predictor p and the threshold t_p that minimize the **weighted average entropy** over the two regions:

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

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

Comparison of Criteria

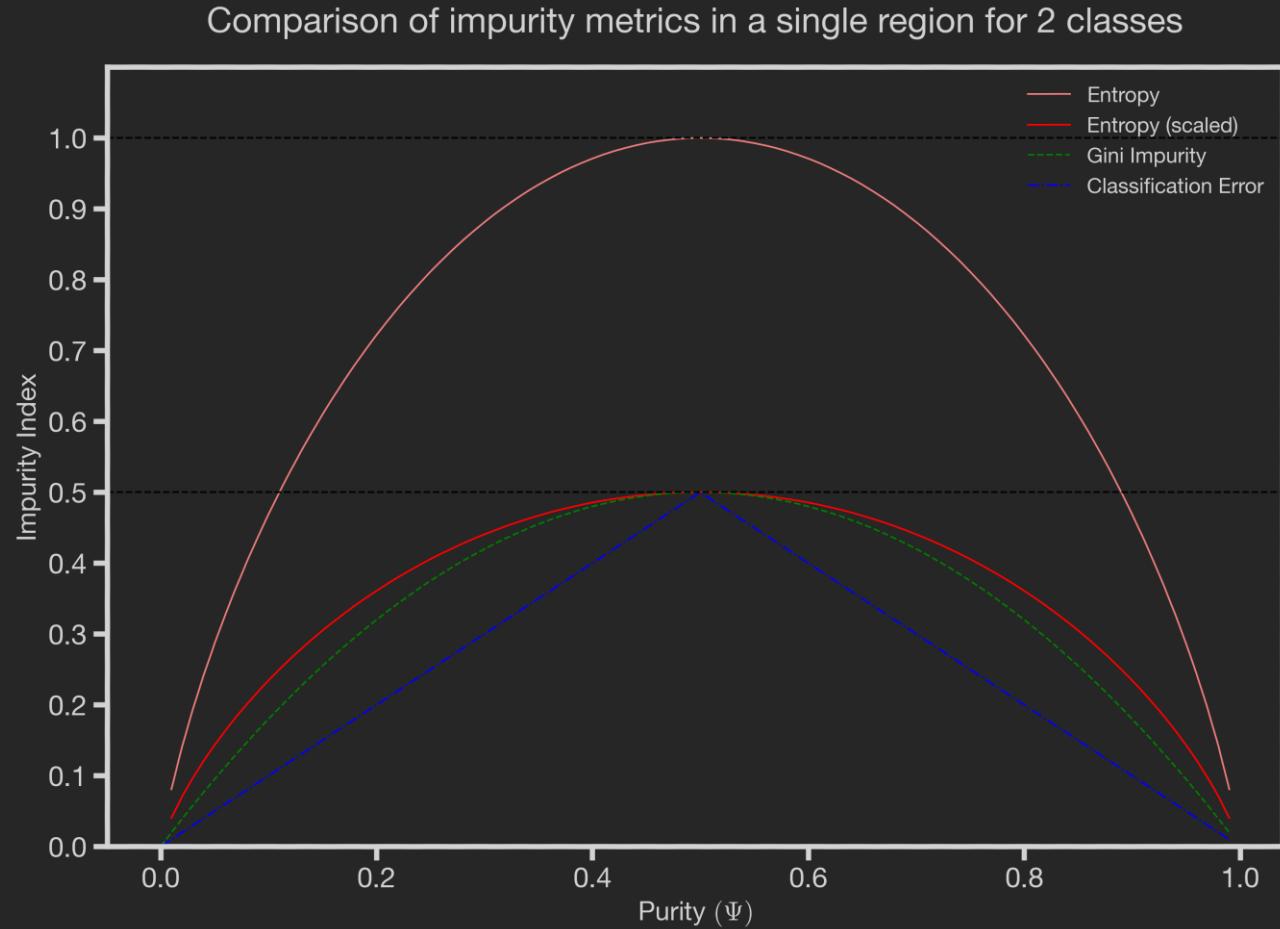
Question: Which of the three criteria fits our guideline the best?



Note: Sklearn uses the unscaled entropy as a splitting criterion

Comparison of Criteria

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



Entropy penalizes
impurity the most



Learning the Model

Learning the ‘optimal’ decision tree for any given set of data is NP complete (intractable) for numerous simple definitions of ‘optimal’. Instead, we will use a **greedy algorithm**.

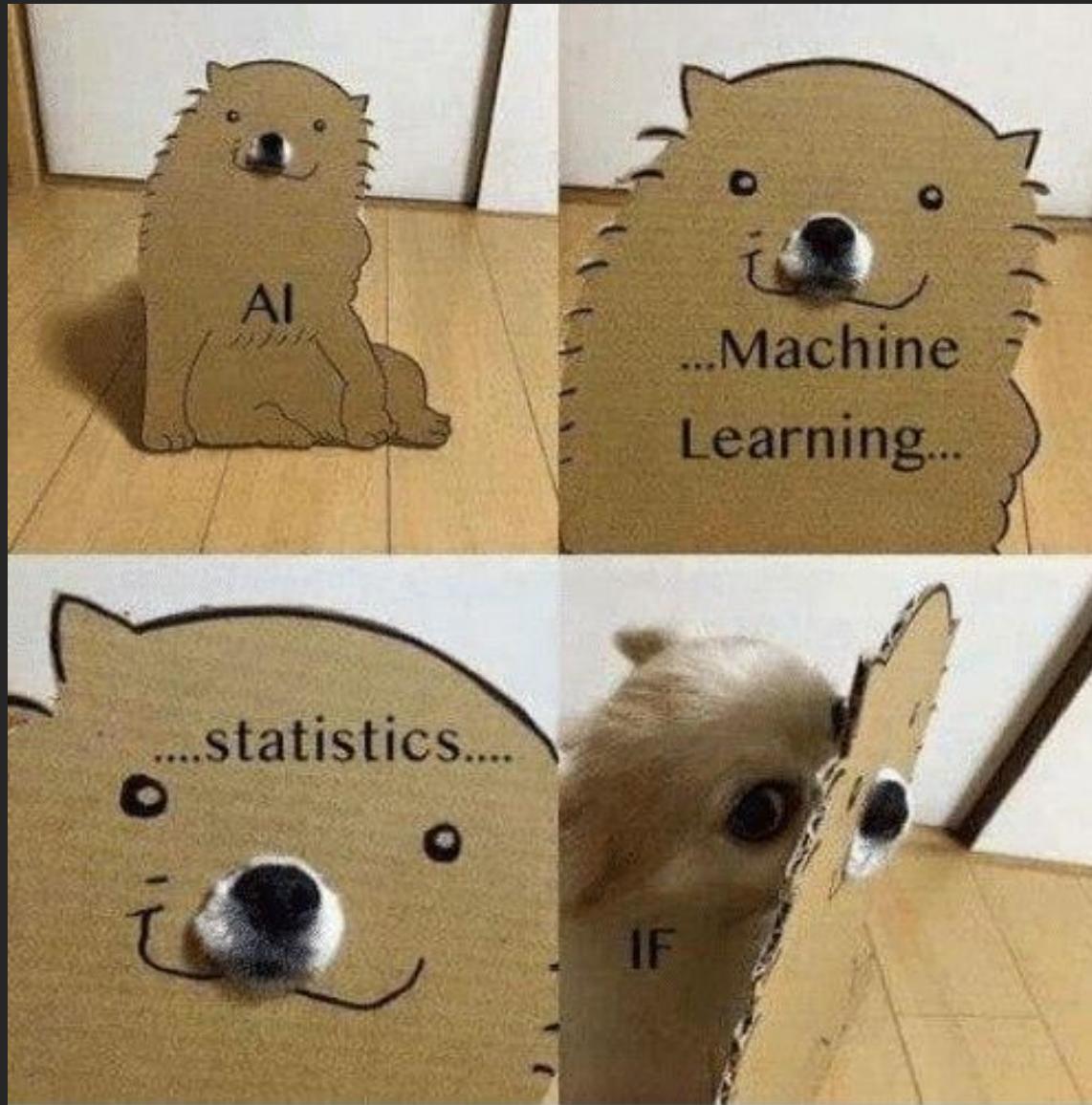
1. Start with an empty decision tree (undivided feature space)
2. Choose the ‘optimal’ predictor on which to split and choose the ‘optimal’ threshold value for splitting.
3. Recurse on each new node until the **stopping condition** is met.
4. For the case of classification, predict each region to have a class label based on the largest class of the training points in that region (majority class).

Learning the Model

Recurse on each new node until the **stopping condition** is met.

Outline

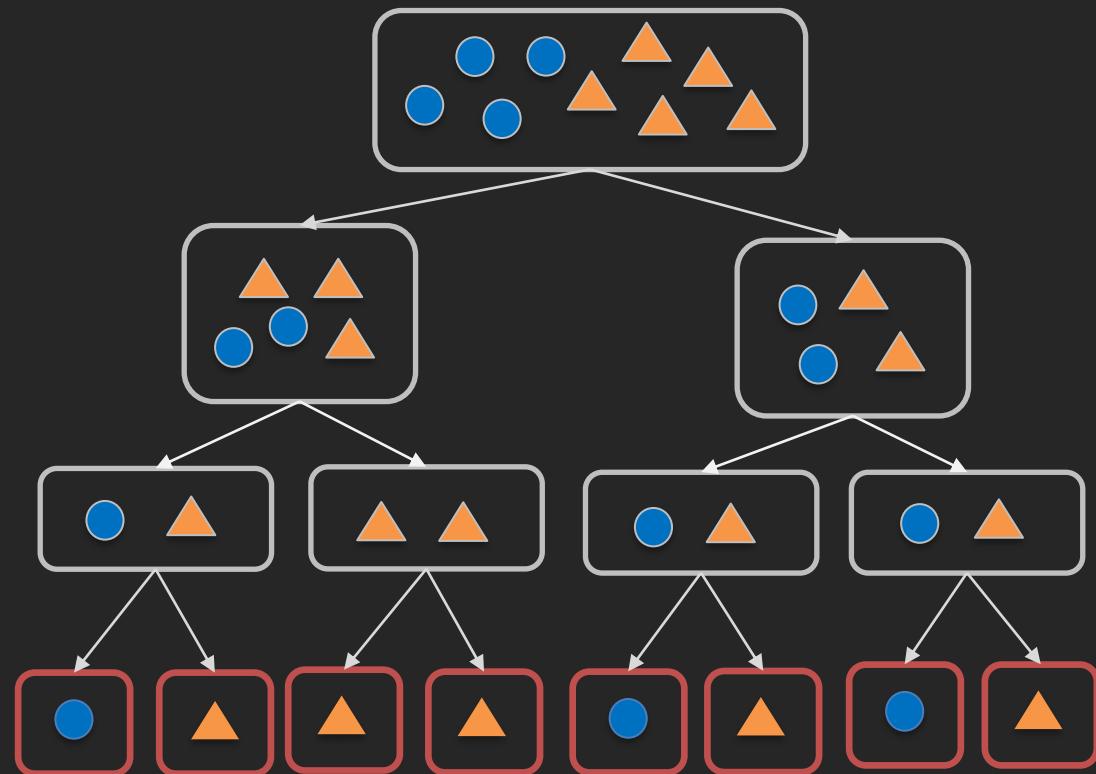
- Motivation
- **Decision Trees – Classification**
 - Intuition
 - Predictions
 - Splitting Criteria
 - **Stopping Conditions**
- Decision Trees - Regression



IF IF IF IF IF IF IF WE!

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

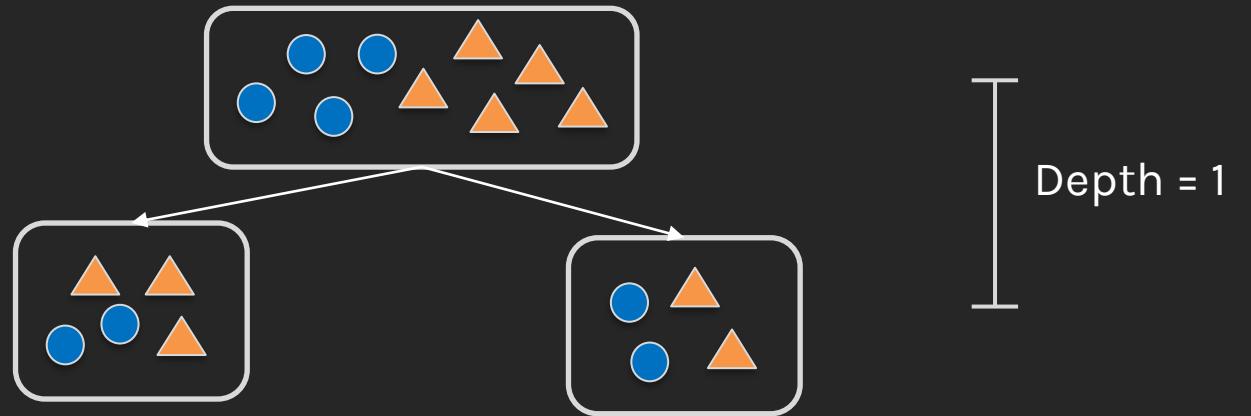


Question: How can we prevent this from happening?

Stopping Conditions

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

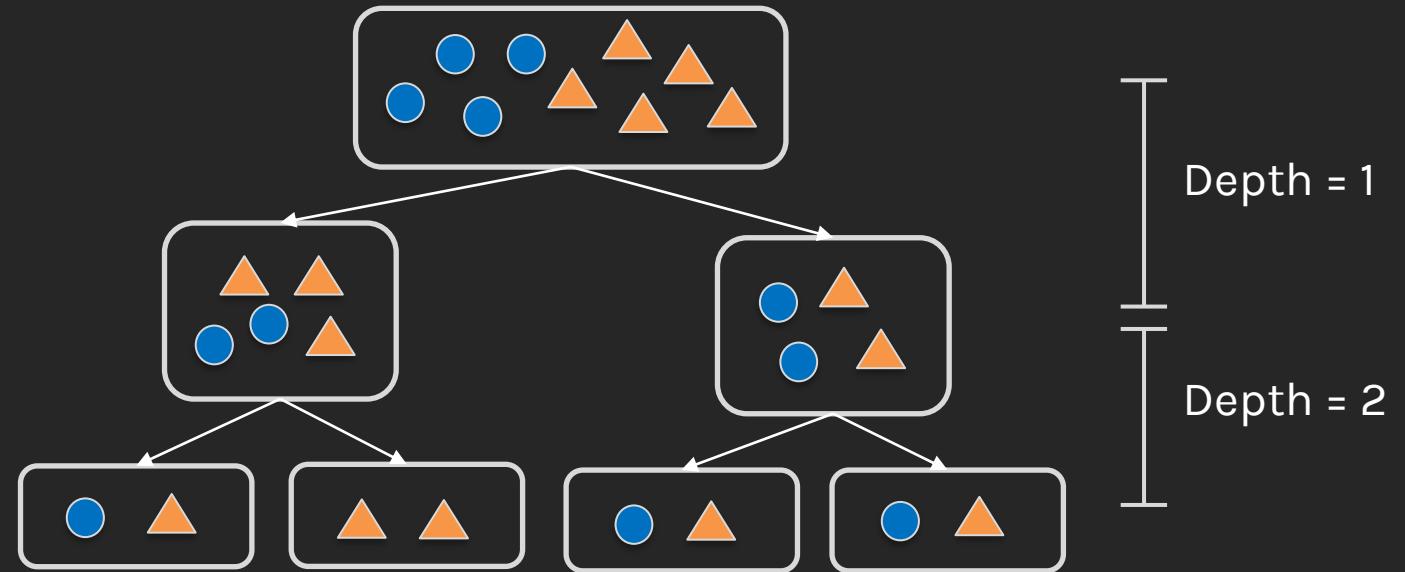
$$\text{max_depth} = 1$$



Stopping Conditions

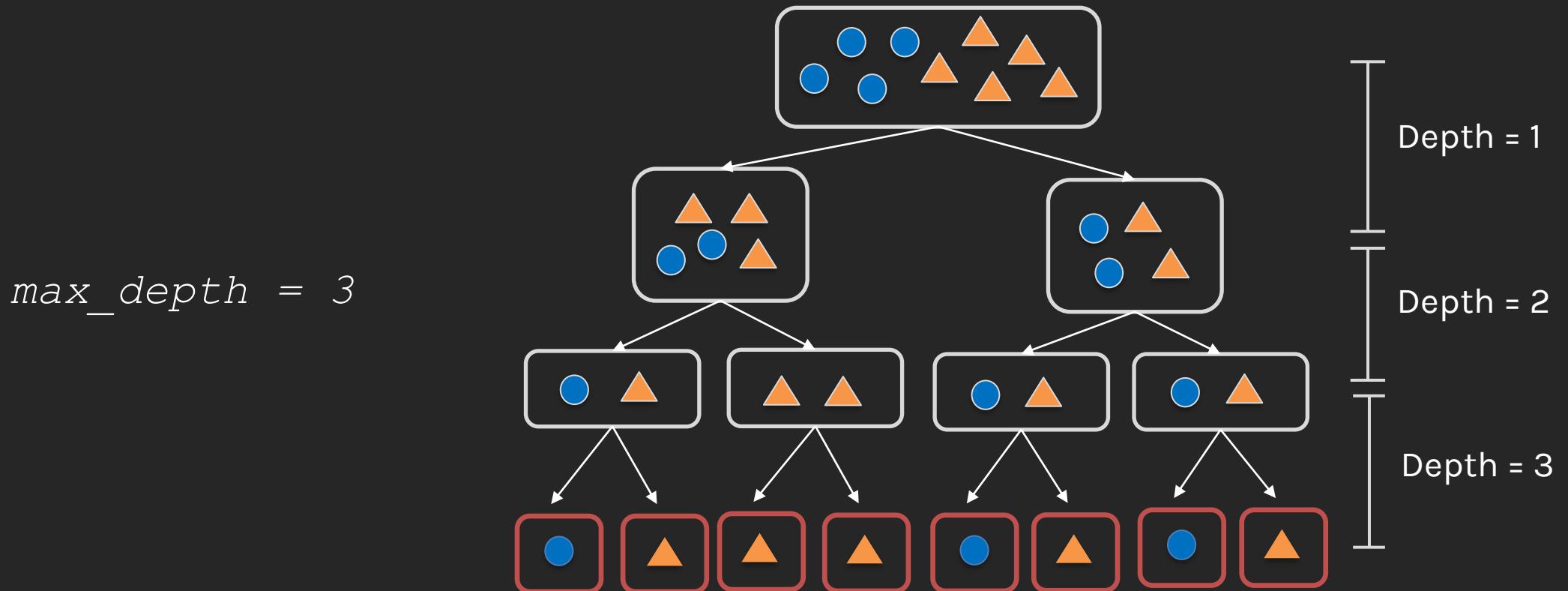
The most common stopping condition is to limit the maximum depth (*max_depth*) of the tree.

max_depth = 2



Stopping Conditions

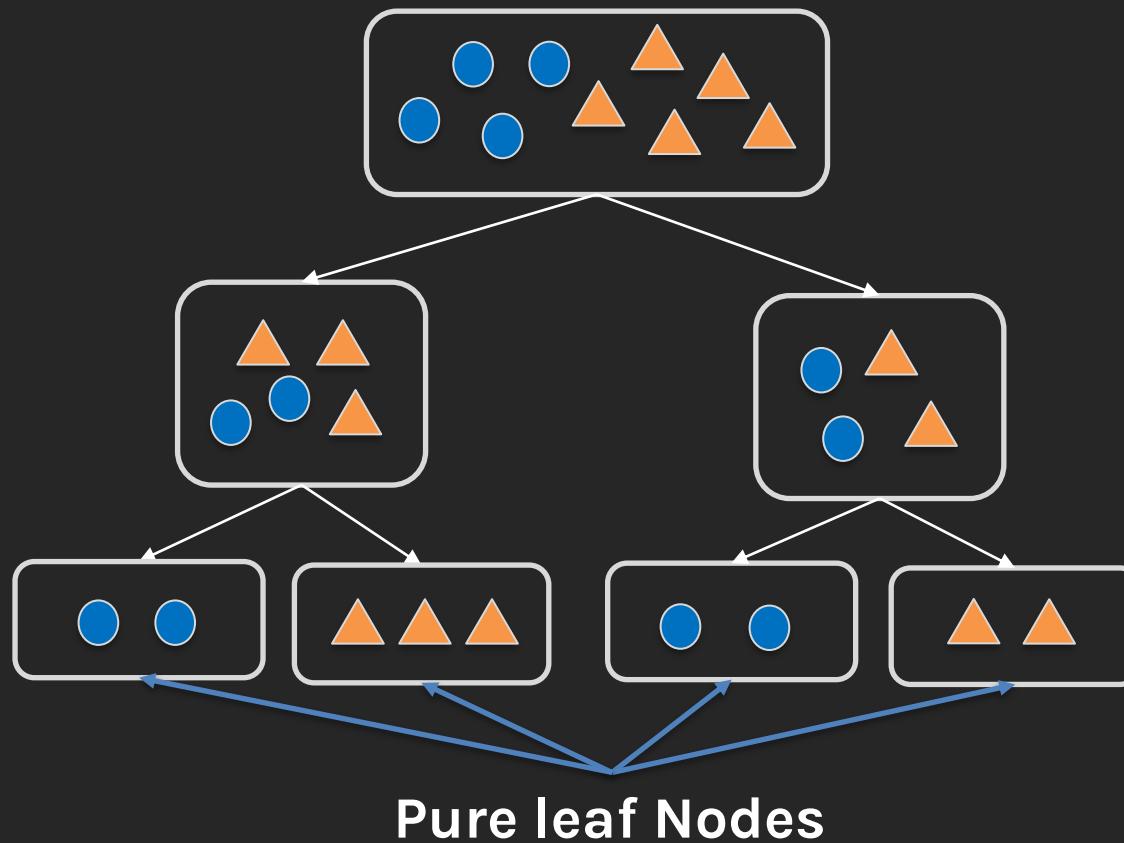
The most common stopping condition is to limit the maximum depth (*max_depth*) of the tree.



Stopping Conditions

Other common simple stopping conditions are:

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

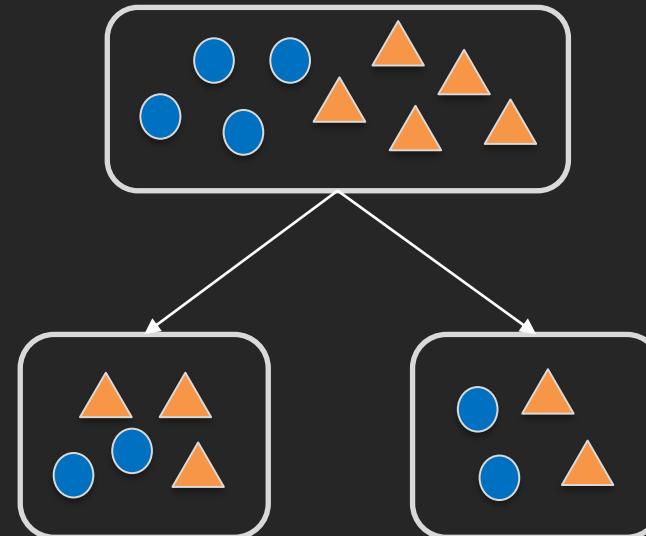


Stopping Conditions

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 ($min_samples_leaf$).

$min_samples_leaf = 4$



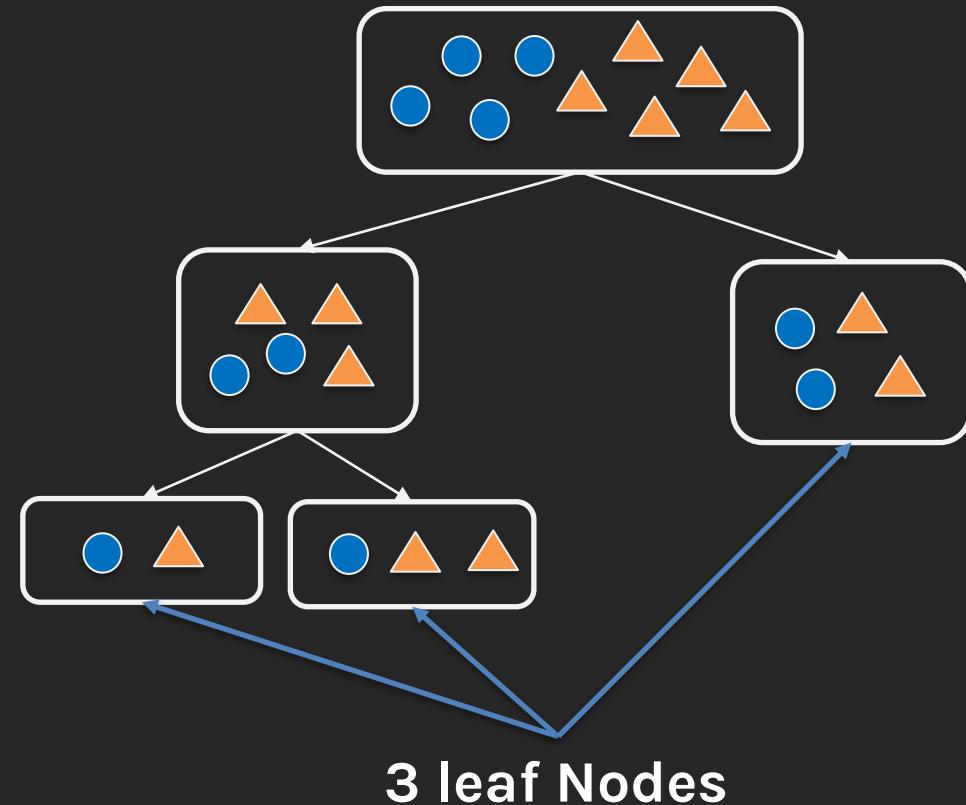
Stopping Conditions

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 does 'level-order' 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.



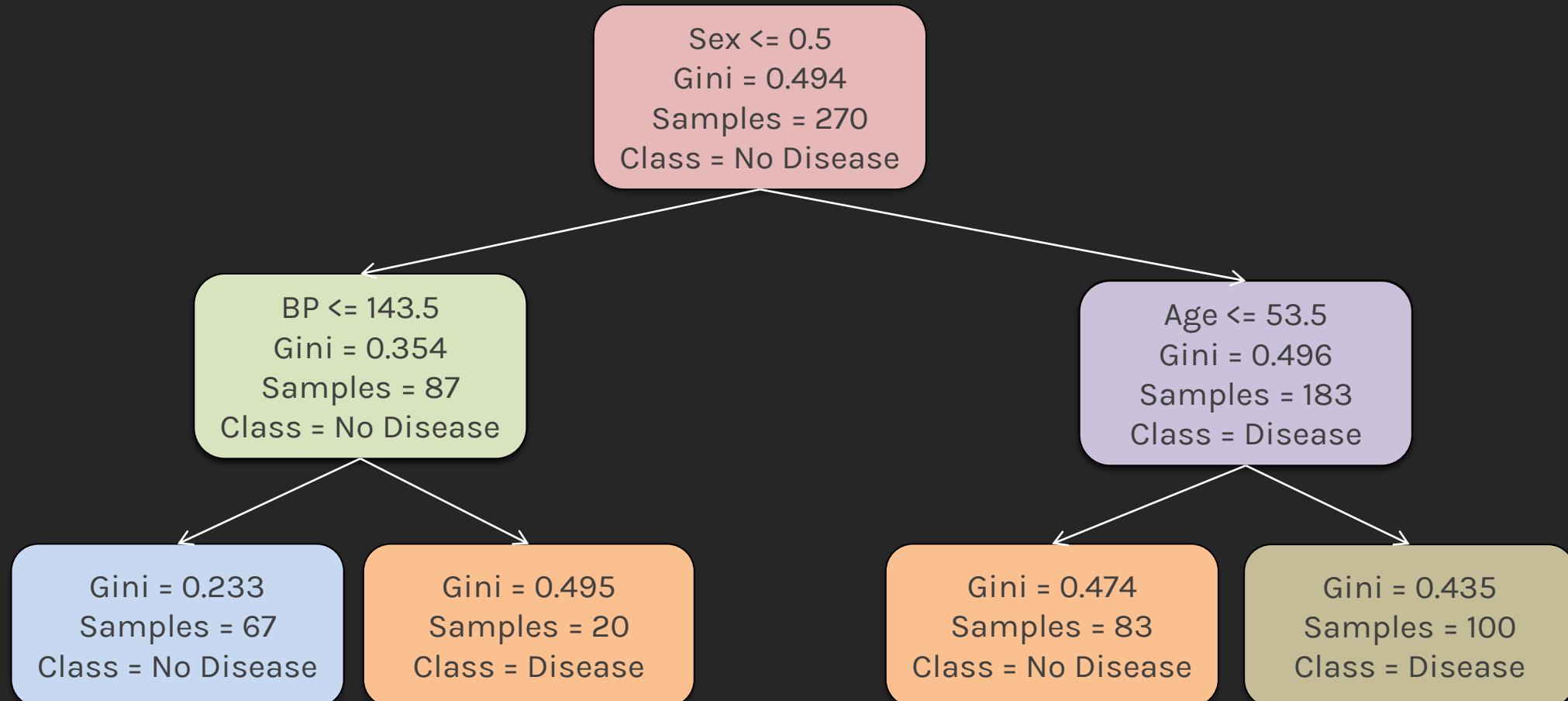
Example 1: Level-Order

Consider the following decision tree with max_depth=2 that predicts if a person has heart disease based on age, sex, BP and cholesterol:

Gini = 0.494
Samples = 270
Class = No Disease

Example 1: Level-Order

Consider the following decision tree with `max_depth=2` that predicts if a person has heart disease based on age, sex, BP and cholesterol:



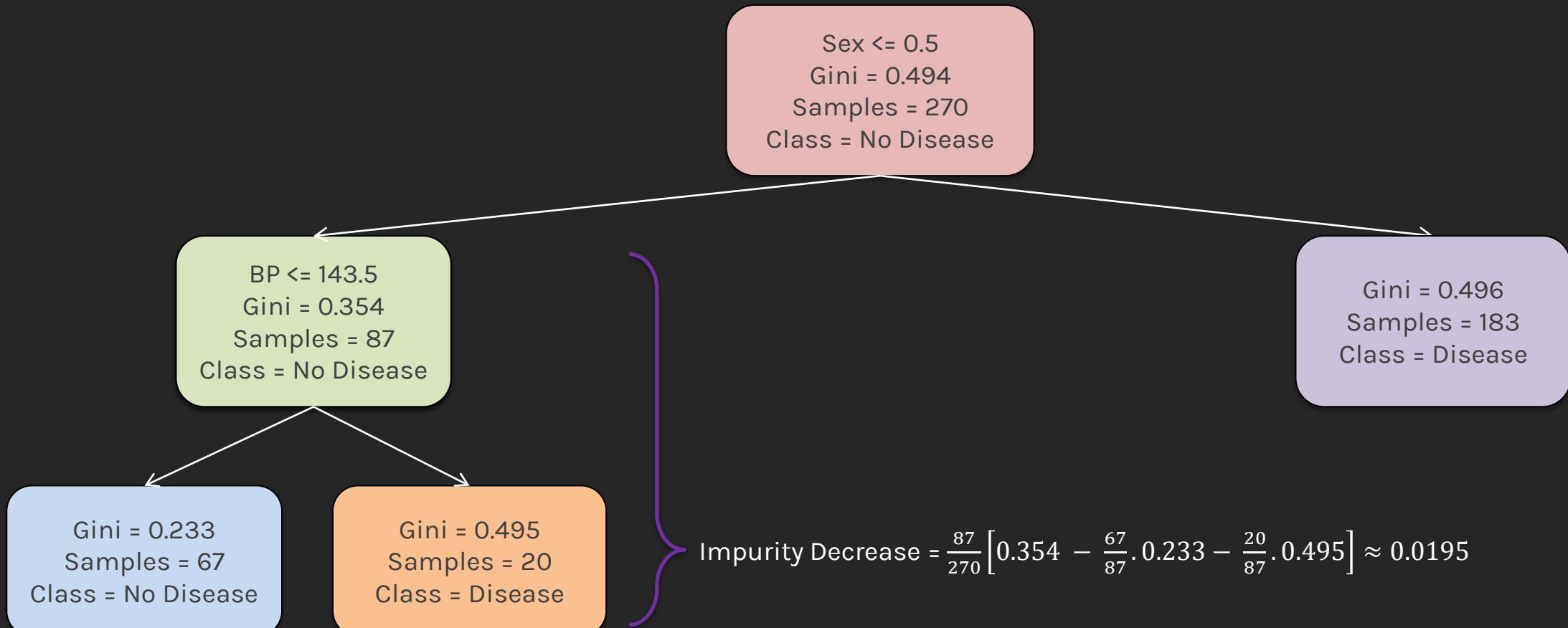
Example 1: Best-first growth

Sklearn determines the best split based on **impurity decrease**. The resulting tree will be the same when fully grown, just the order in which it is built is different.

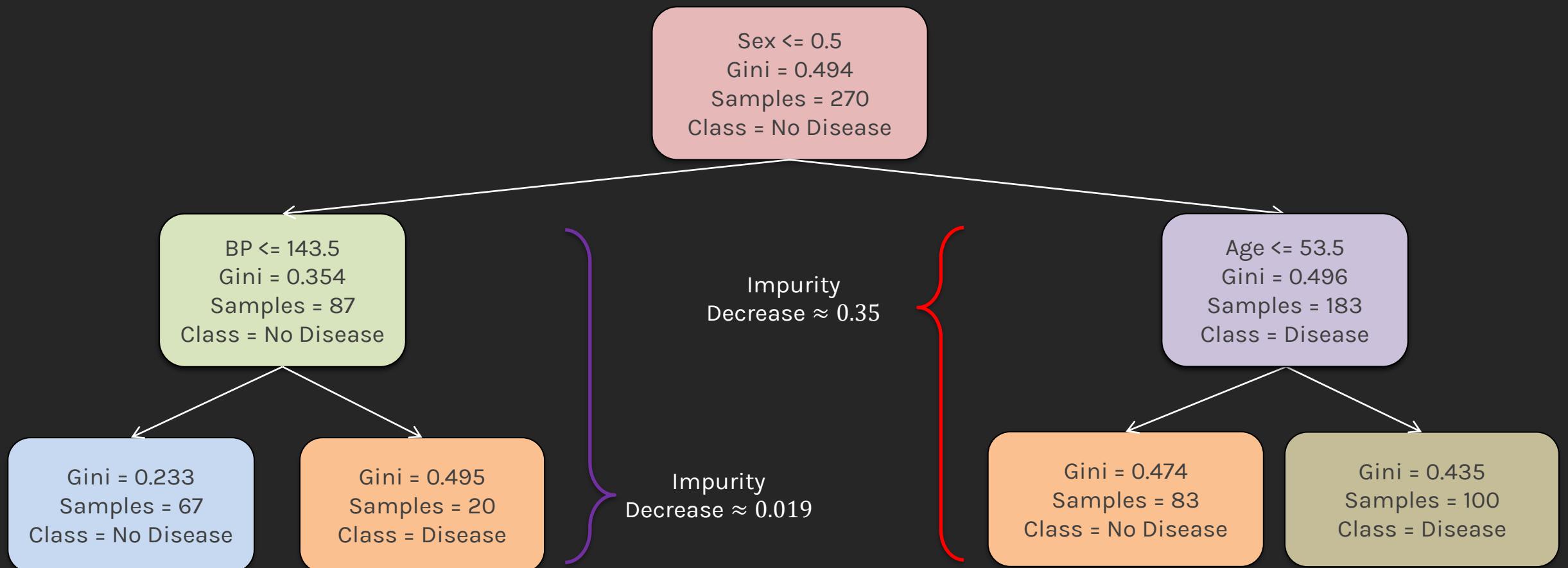
Gini = 0.494
Samples = 270
Class = No Disease

Example 1: Best-first growth

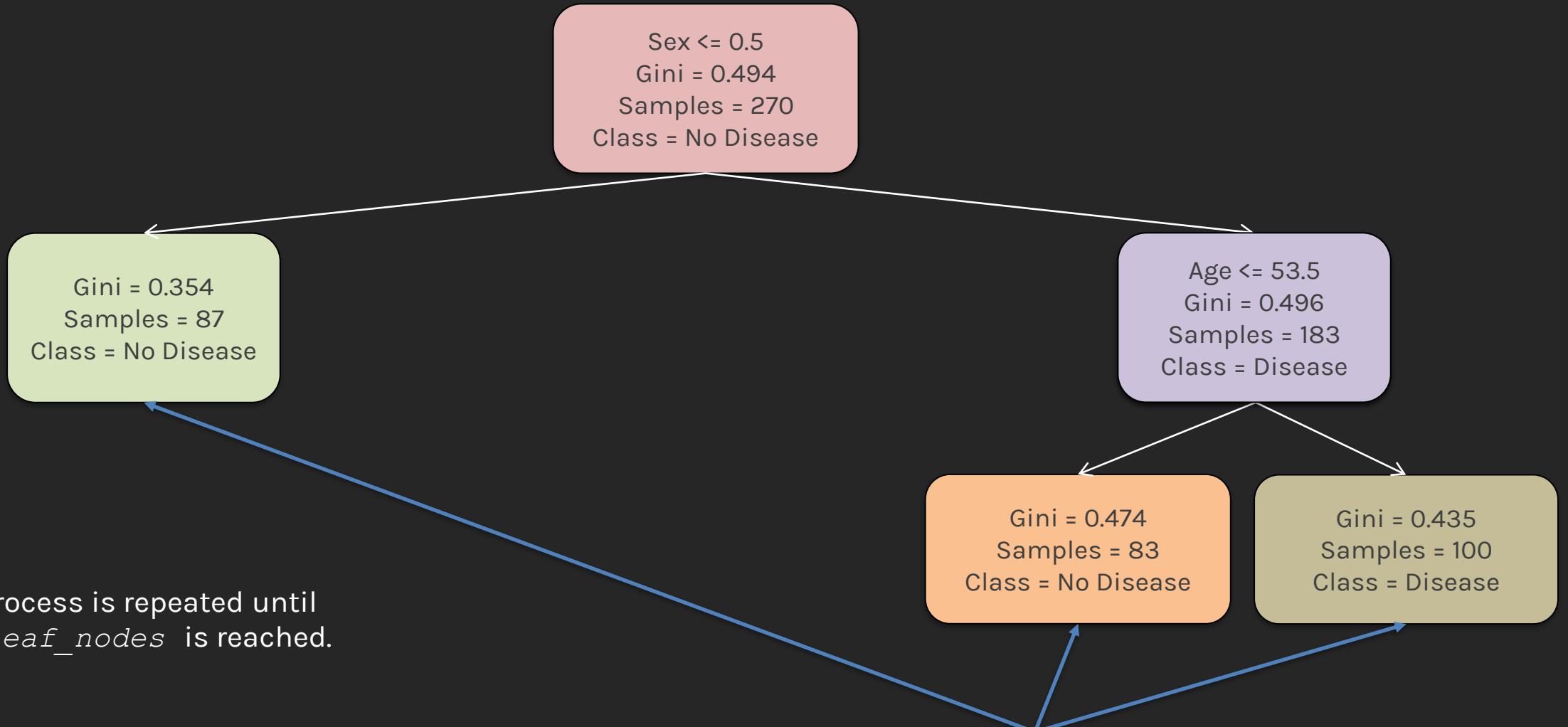
Sklearn determines the best split based on **impurity decrease**. The resulting tree will be the same when fully grown, just the order in which it is built is different.



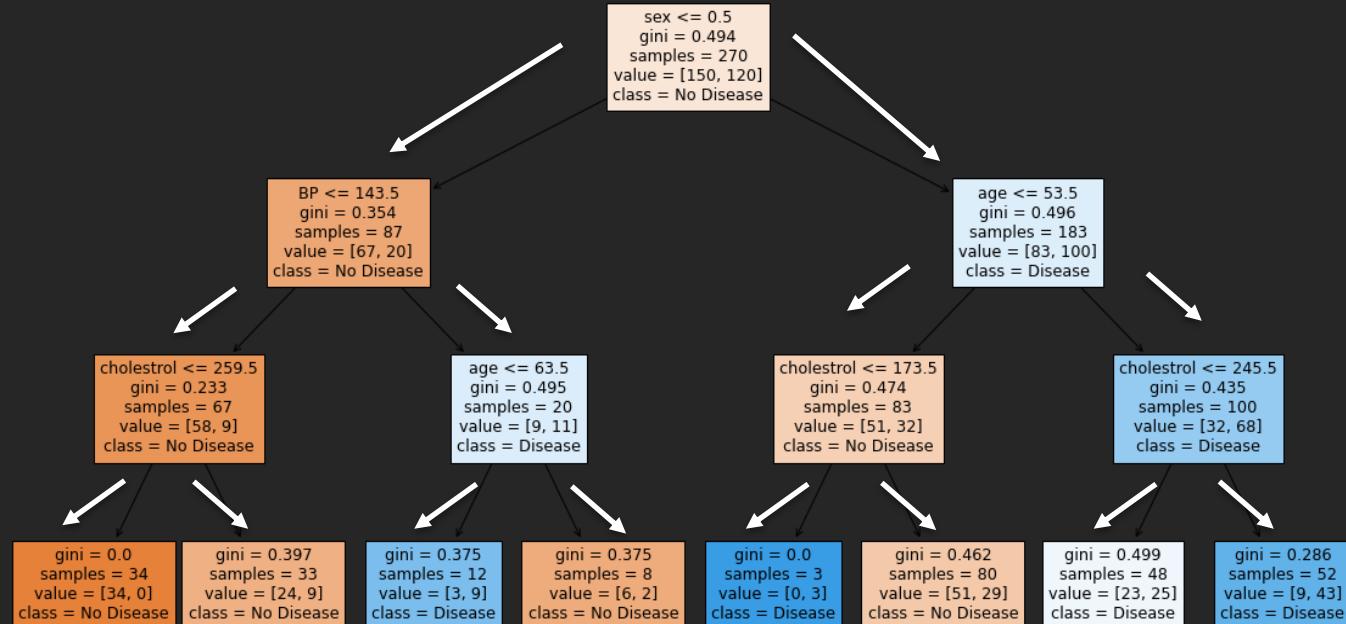
Example 1: Best-first growth



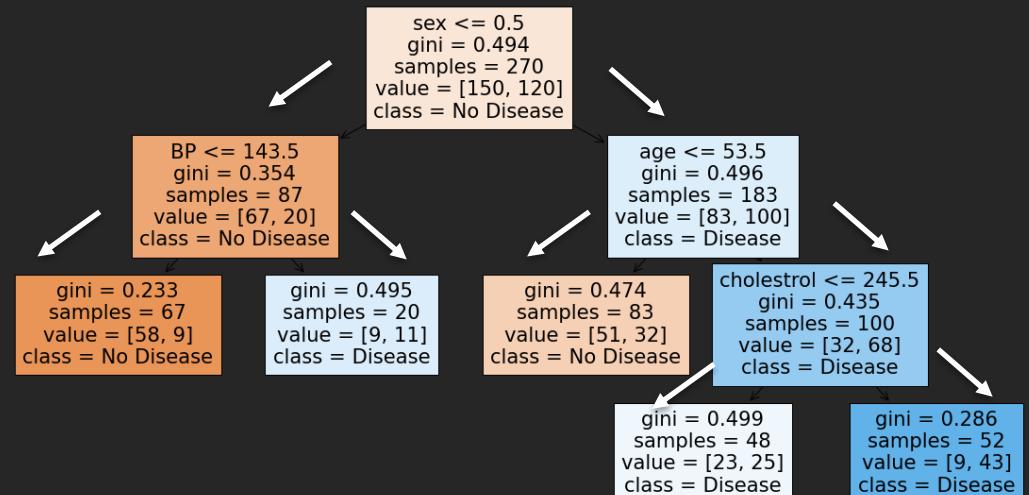
Example 1: Best-first growth



Example 2: Level-order vs Best-first growth



max_depth = 3



max_leaf_nodes = 5

Stopping Conditions

A more restrictive stopping condition is:

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

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

\uparrow

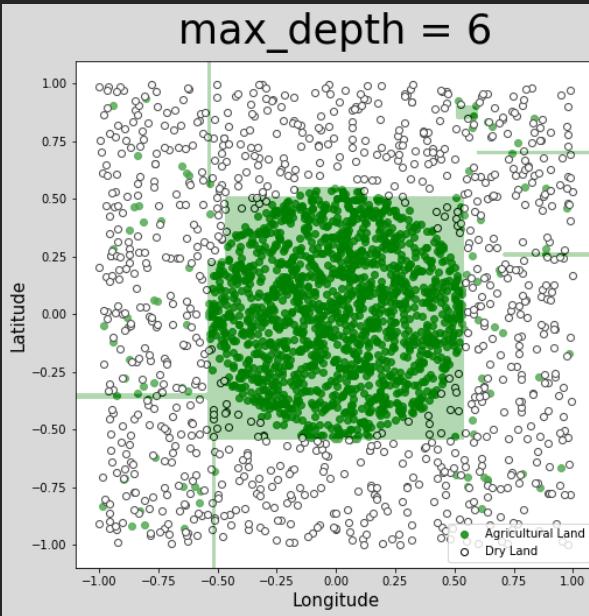
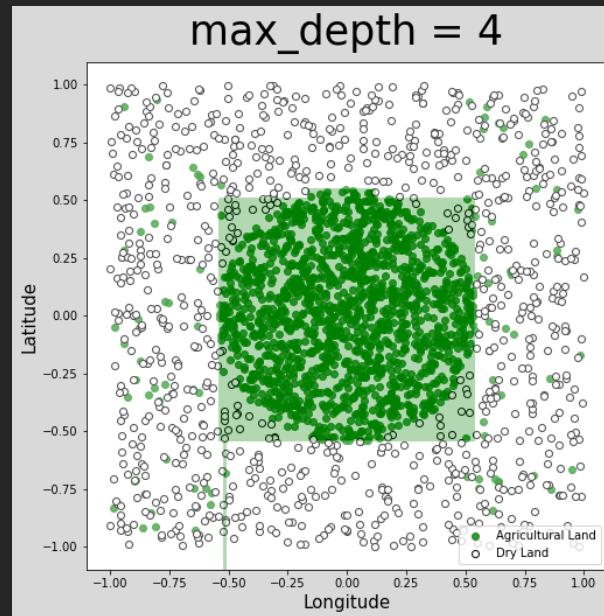
Classification Error/Gini Index/Entropy

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

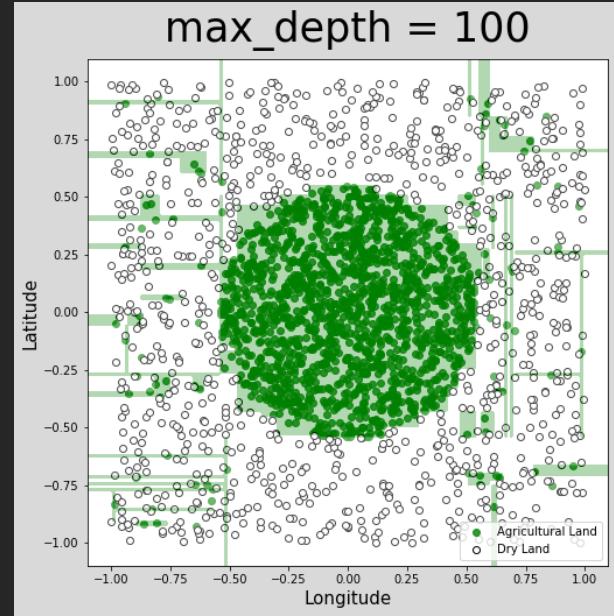
How do we decide what is the appropriate stopping condition or stopping method?



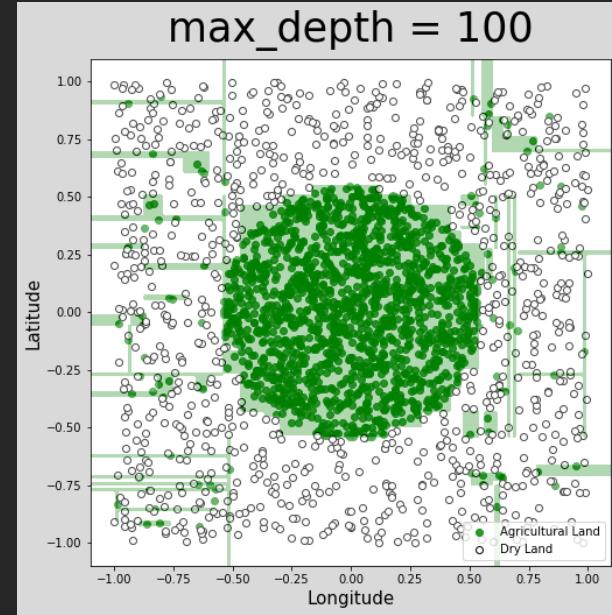
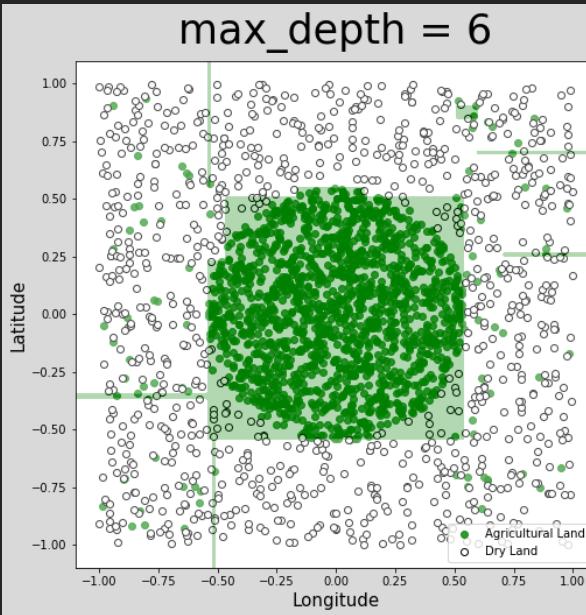
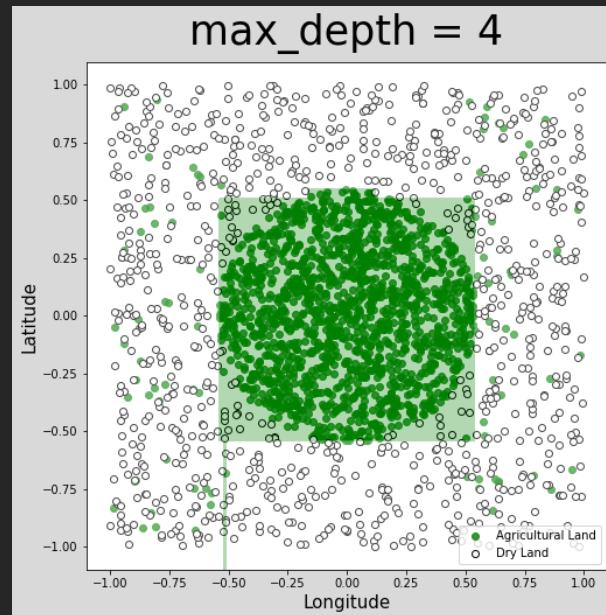
Variance vs Bias



• • • •



Variance vs Bias

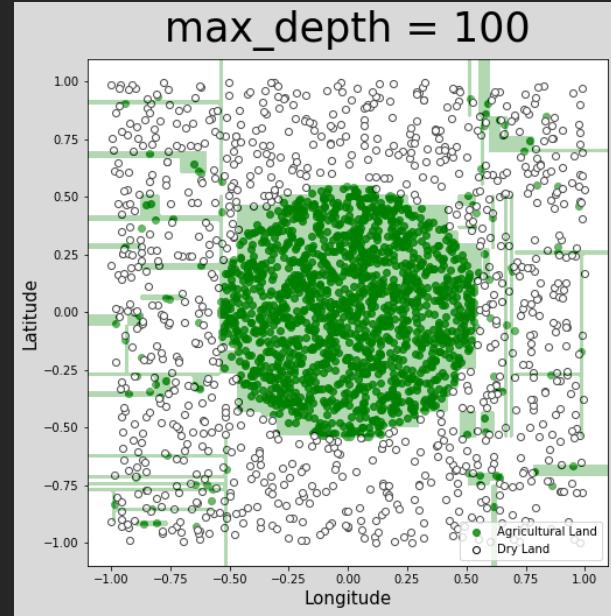
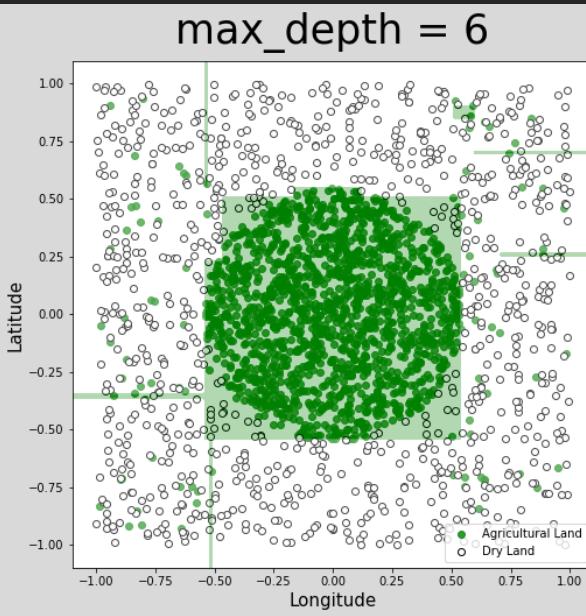
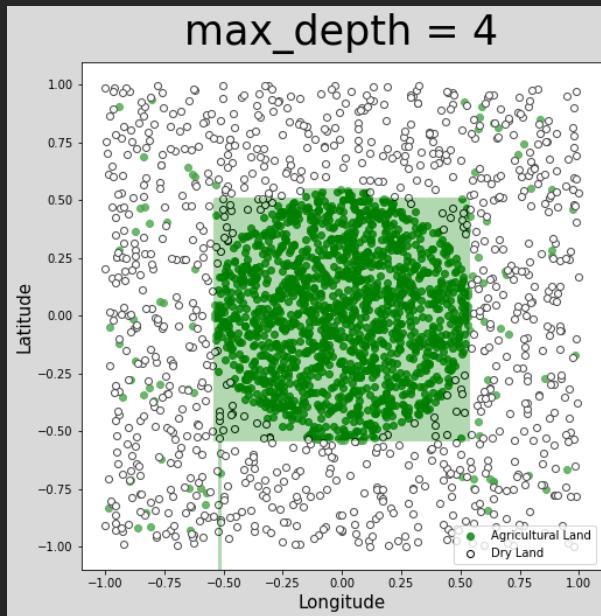


• • • •



Bias decreases (can overfit)

Variance vs Bias



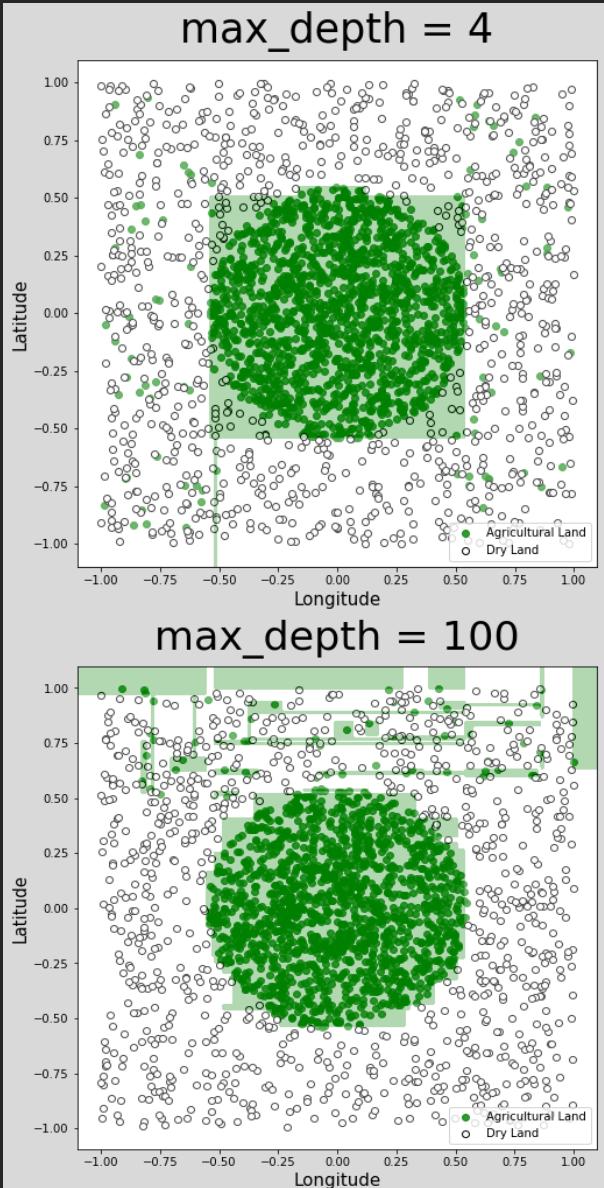
• • • •

Bias decreases (can overfit)

Variance decreases (can underfit)

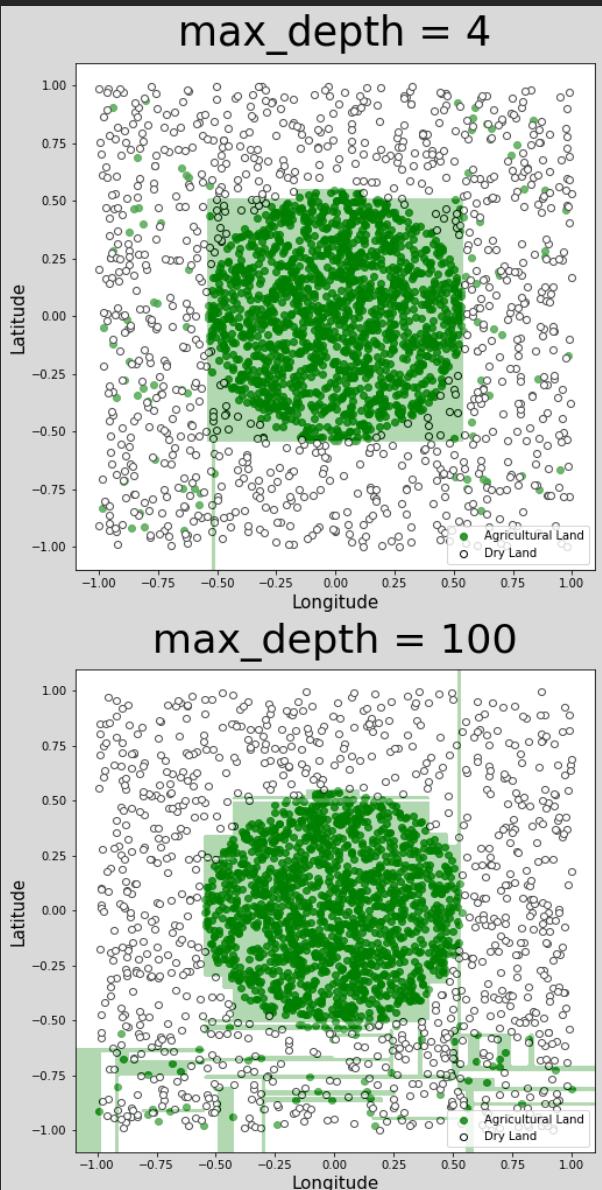
Complex trees are also harder to interpret and more computationally expensive to train.

Variance vs Bias



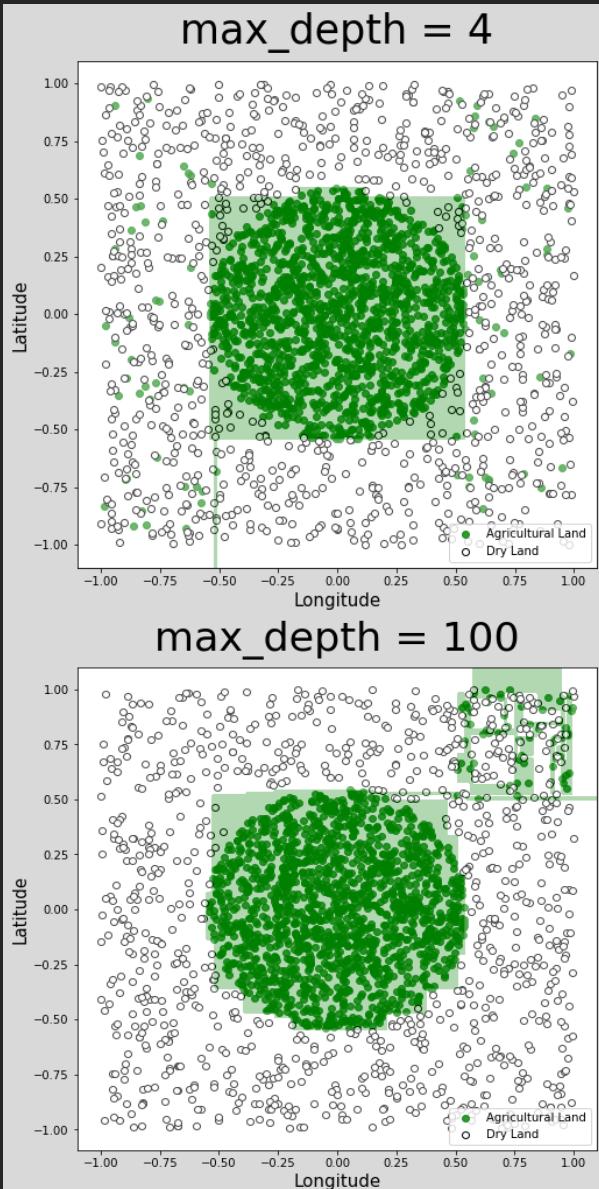
- **High Bias:** Trees of low depth are not a good fit for the training data - it's unable to capture the nonlinear boundary separating the two classes.
- **Low Variance:** Trees of low depth are robust to slight perturbations in the training data - the square carved out by the model is stable if you 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).

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).

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.

Stopping Conditions

`max_depth`

`min_samples_leaf`

`max_leaf_nodes`

`min_impurity_decrease`

How can we determine the appropriate hyperparameters?



cross-validation

Outline

- Motivation
- Decision Trees - Classification
 - Intuition
 - Predictions
 - Splitting Criteria
 - Stopping Conditions
- **Decision Trees - Regression**

Regression Trees

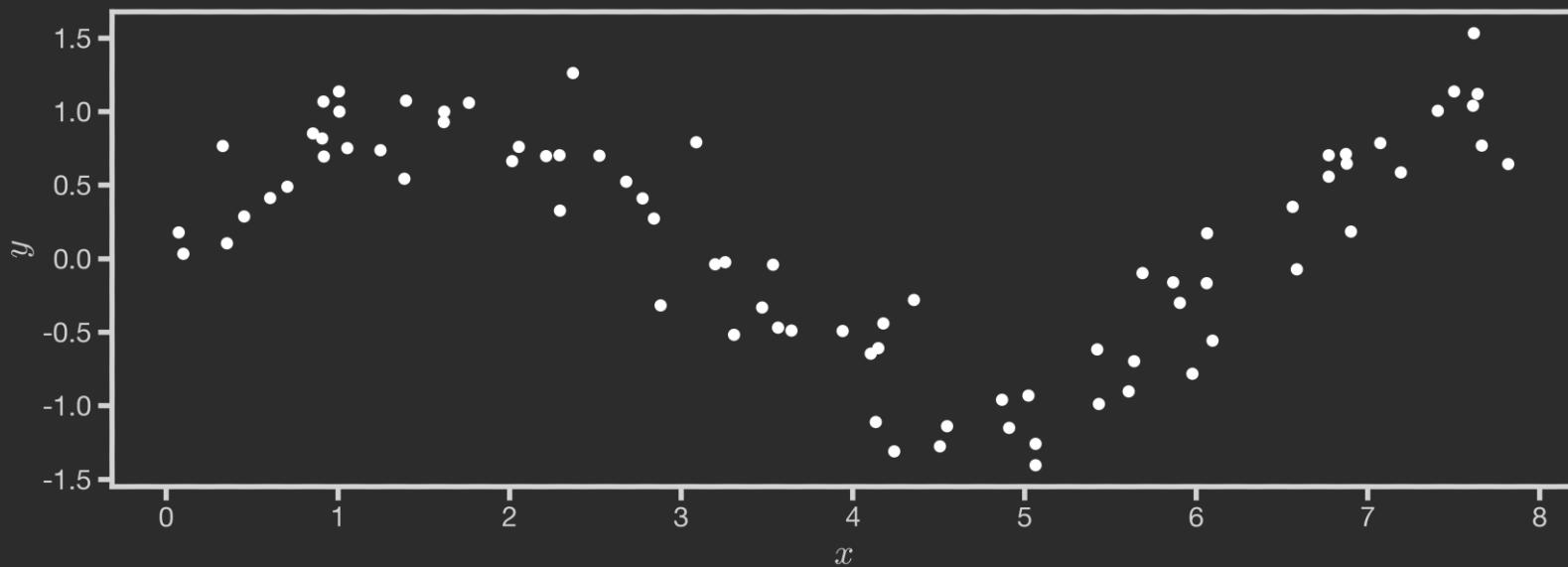
How can this decision tree approach apply to a *regression problem* (quantitative outcome)?

Questions to consider:

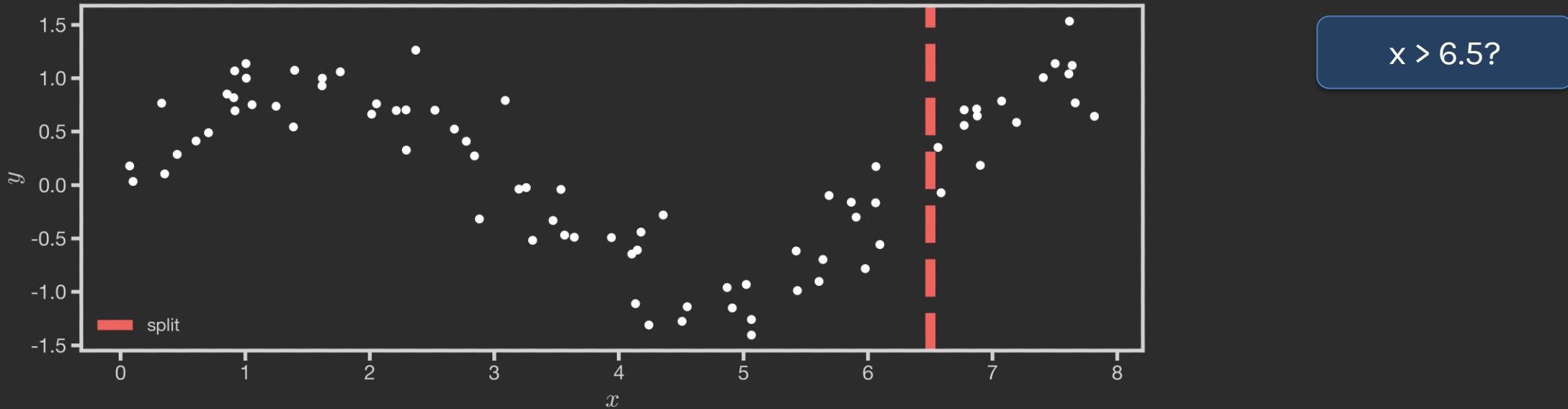
- How would you determine any **splitting criteria**?
- What would be a reasonable **objective function**?
- How would you perform **prediction in each leaf**?

Splitting Criteria

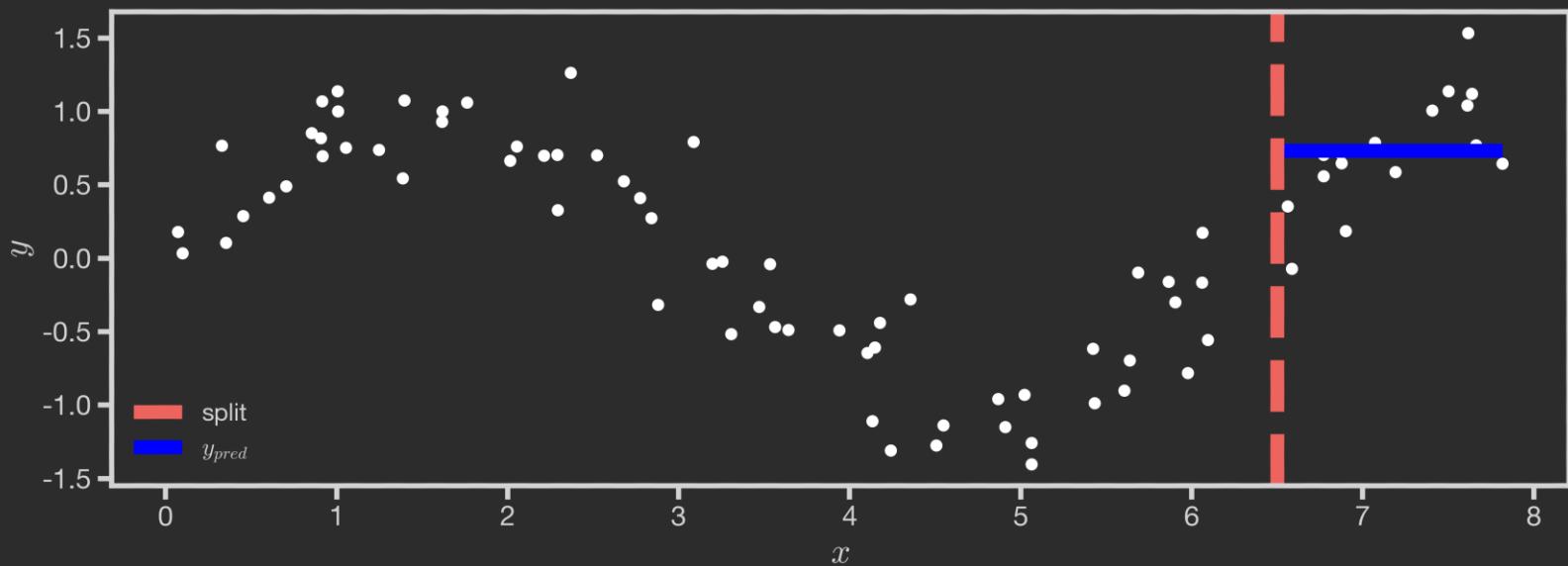
The plot visualizes data points of $\{x, y\}$. By observing patterns, we can identify how and where to make optimal splits. Ideally, each split should lead to groups of data points with more homogeneous values, allowing for better predictions.



Splitting Criteria



Splitting Criteria

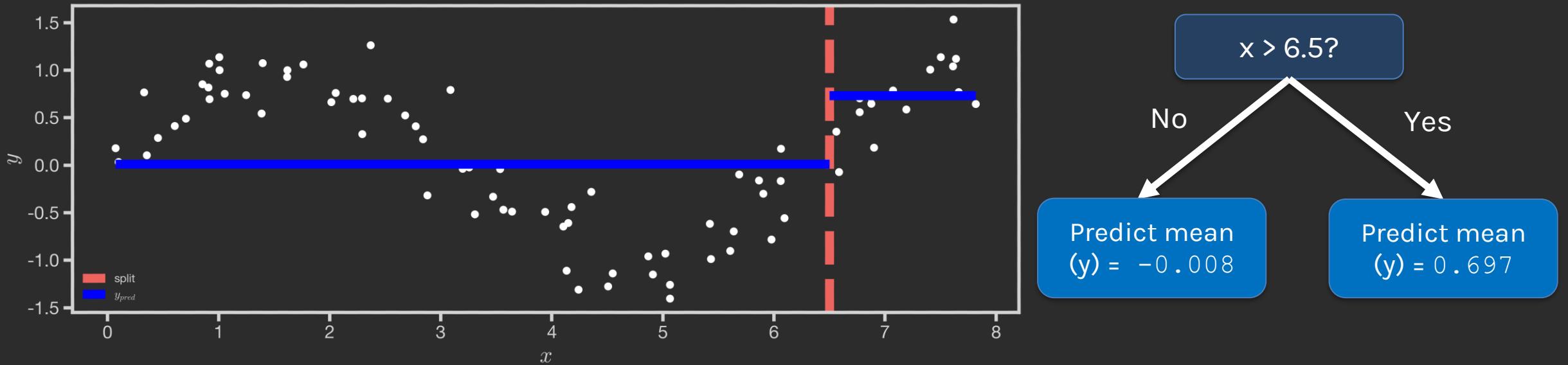


$x > 6.5?$

Yes

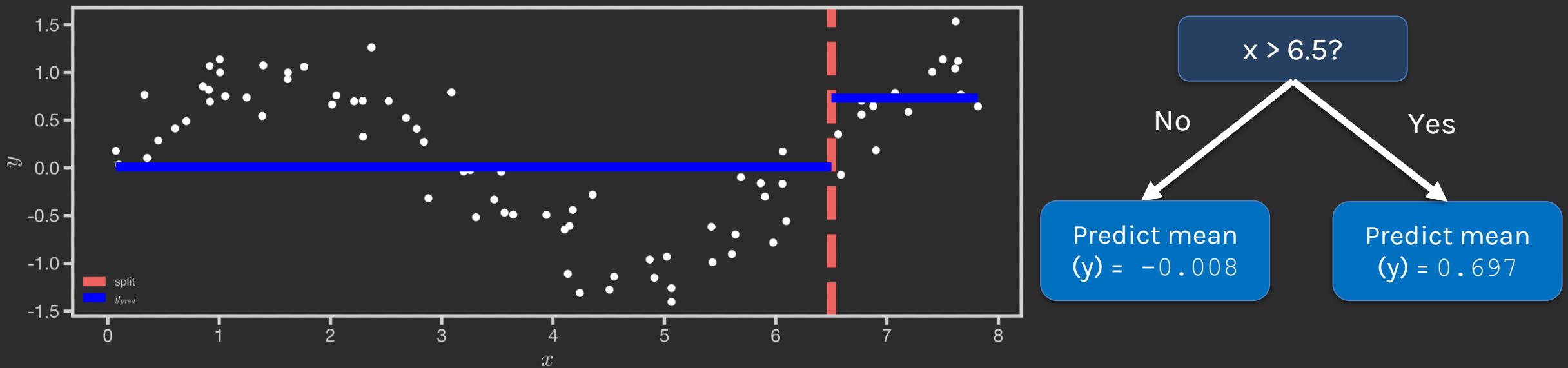
Predict mean
(y) = 0.697

Splitting Criteria



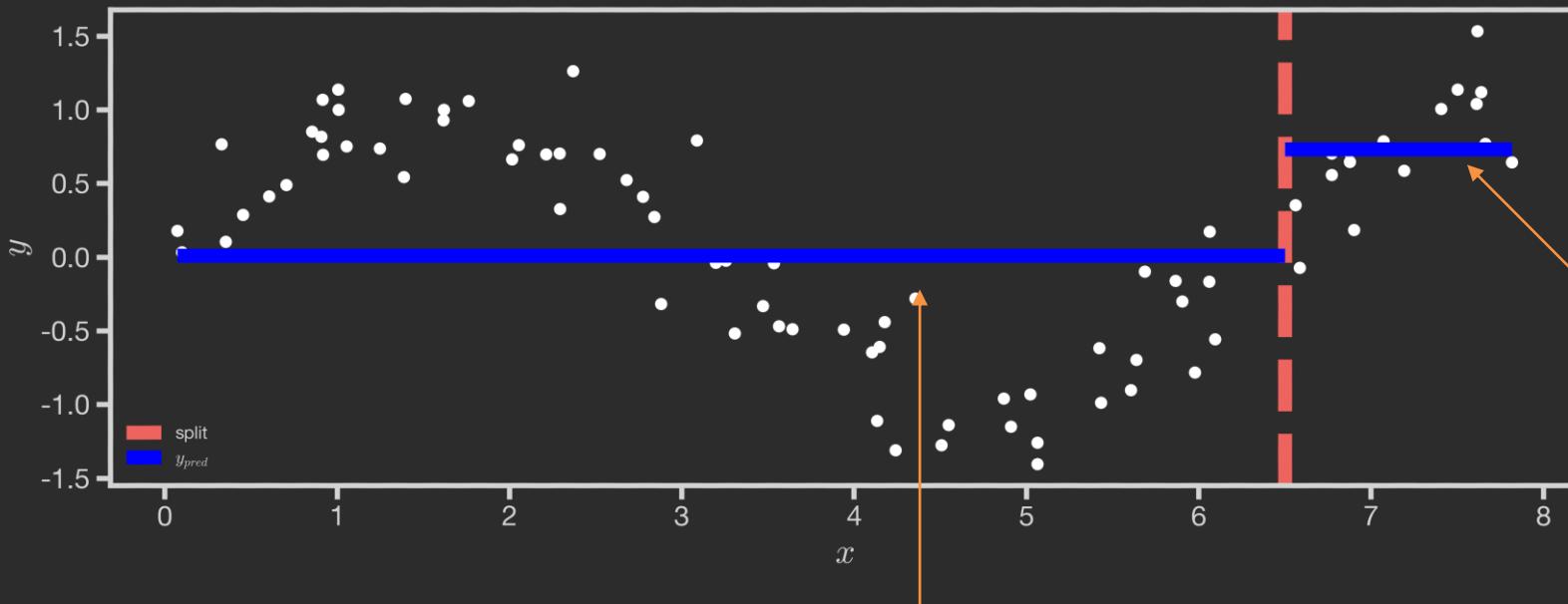
Splitting Criteria

Question: How did we choose the splitting criteria for this regression tree?



Splitting Criteria

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

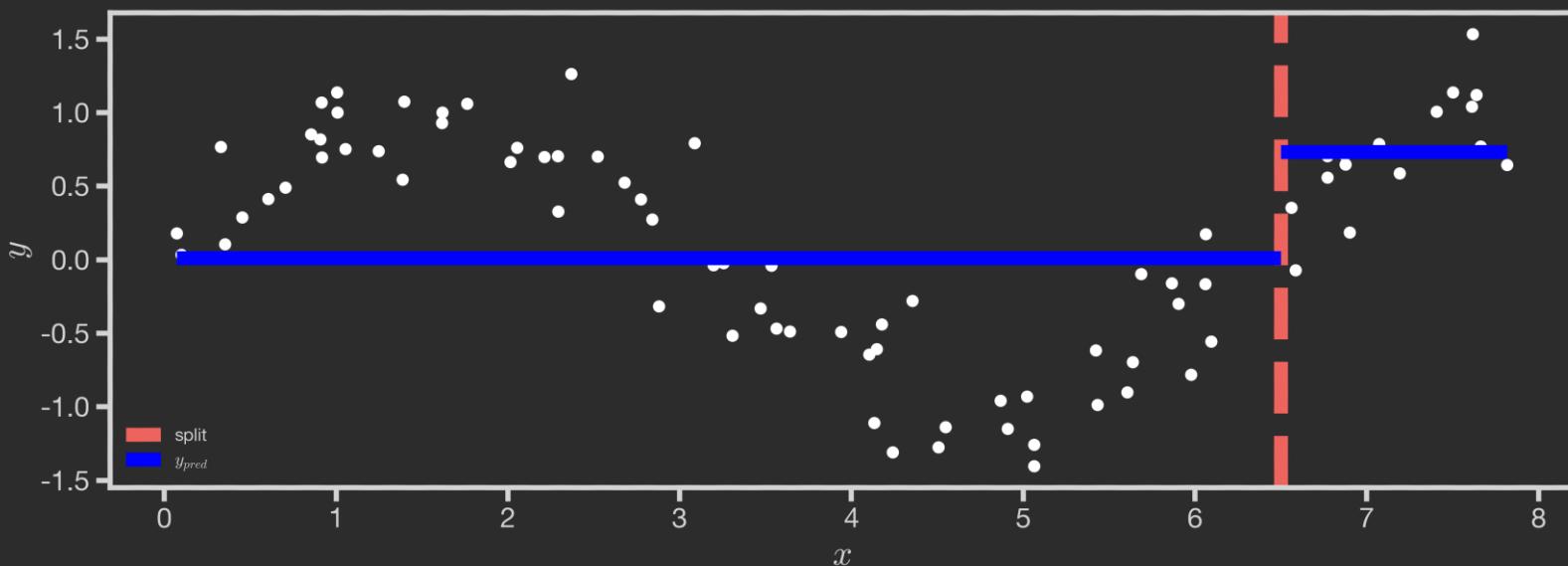


$$MSE(R_1) = \frac{1}{n_1} \sum_{i \in R_1} (y_i - \bar{y}_{R_1})^2$$

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

Splitting Criteria

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



This is the prediction

$$MSE(R_r) = \frac{1}{n_r} \sum_{i \in R_{wr}} (y_i - \bar{y}_{R_r})^2$$

Note: This is the same as the variance!

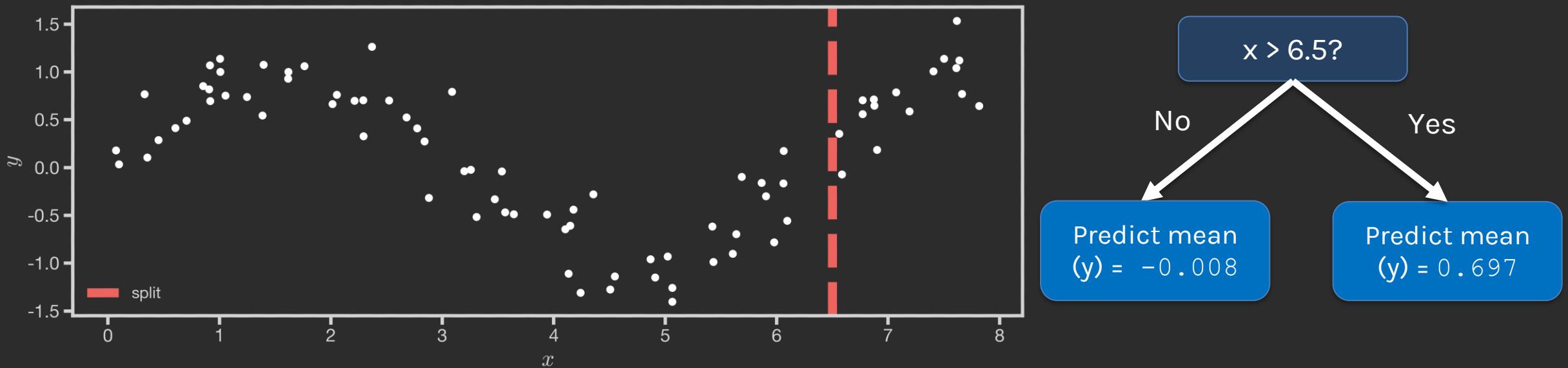
Splitting Criteria

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

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

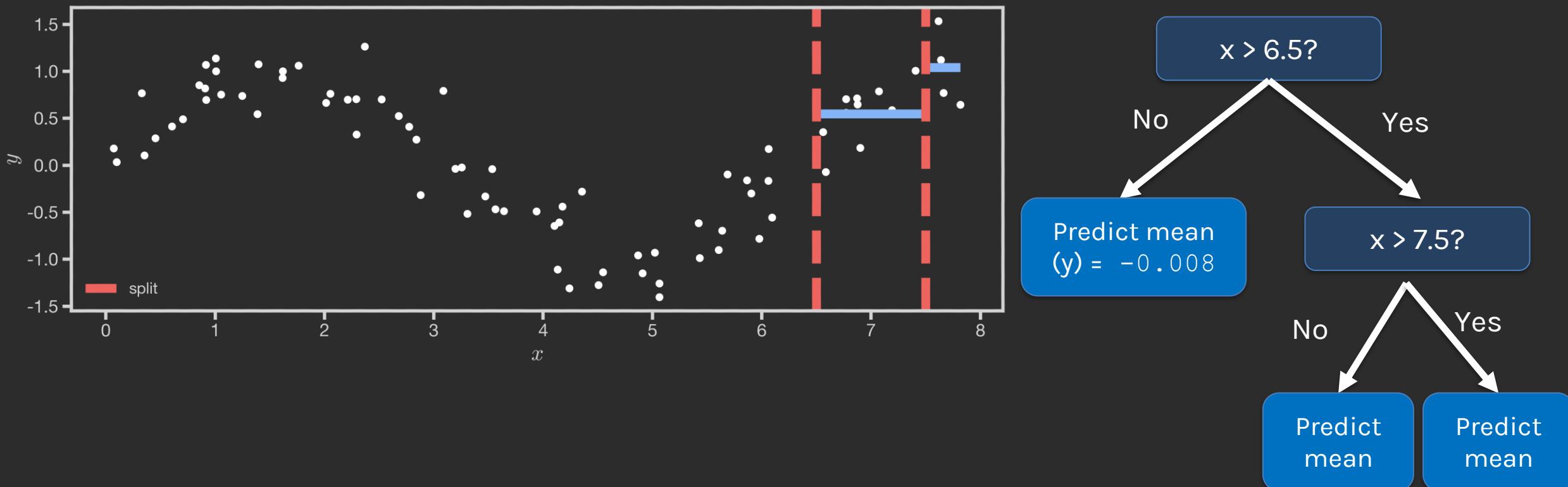
Splitting Criteria

Question: How did we choose the splitting criteria for this regression tree?

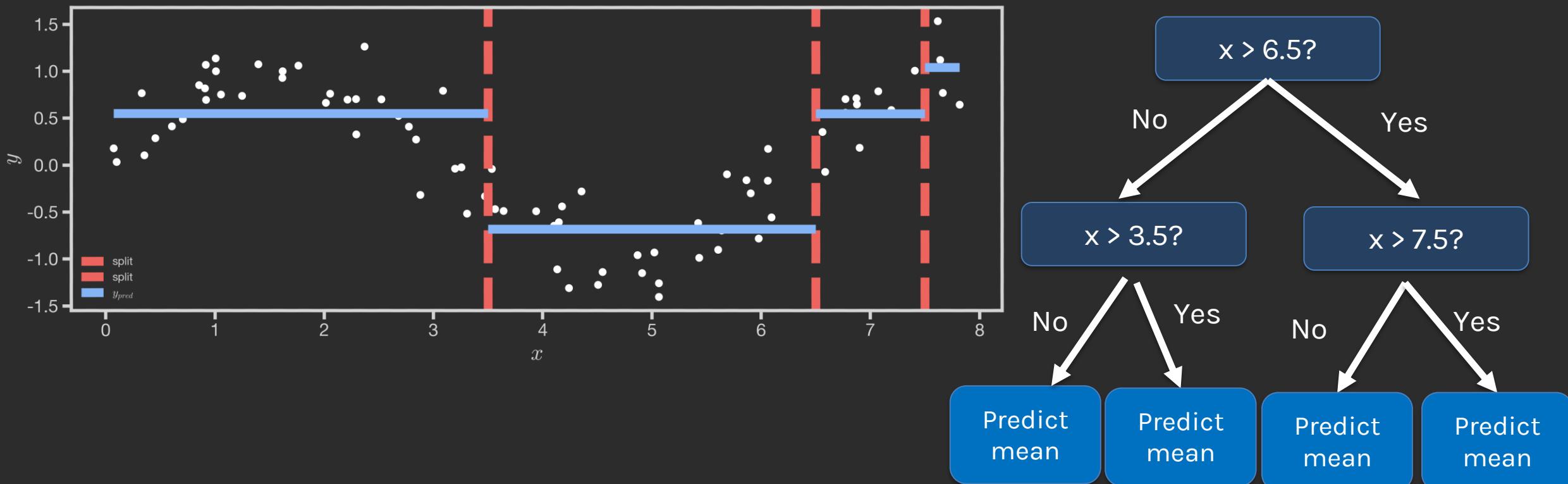


Splitting Criteria

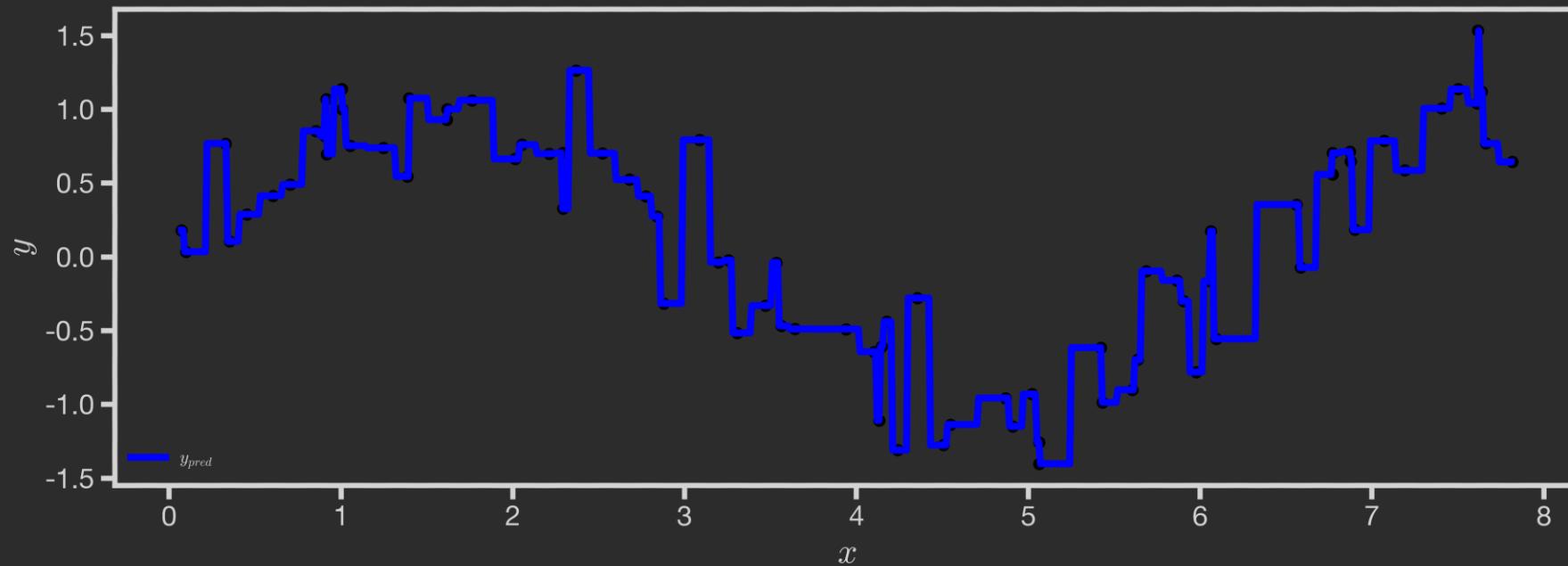
Question: How did we choose the splitting criteria for this regression tree?



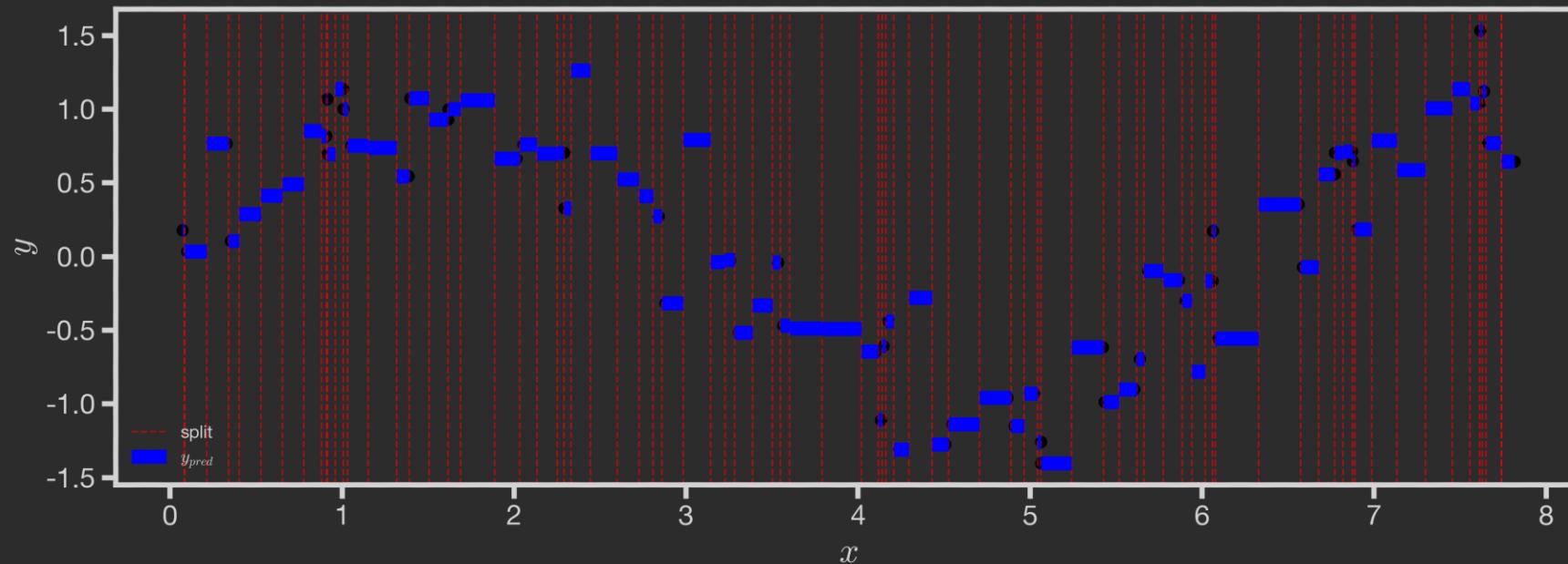
Regression Tree (max_depth = 2)



Regression Tree (max_depth = 5)



Regression Tree (max_depth = 10)



Stopping Conditions

Most of the stopping conditions we saw for classification trees can also be applied to regression.

In the place of purity gain, we can instead compute **accuracy gain** (in this case **MSE reduction**) for splitting a region R .



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

MSE after split

Stopping Conditions

Most of the stopping conditions we saw for classification trees can also be applied to regression.

Can you name them?

Are you even paying attention?



Thank you