Machine Learning and AI with Python - Notes

HarvardX CS109xa – Alessio Santoro

October 27, 2024

Chapter 1

Decision Trees

1.1 Decision trees for classification

1.1.1 Decision Trees Part 1

Logistic regression is a fundamental statistical method used in machine learning for binary classification tasks. It predicts the probability of an instance belonging to a particular class.

Part A: Classification using trees You may have learned in previous courses that **logistic regression** is most effective for constructing classification boundaries when:

- The classes are well-separated in the feature space
- The classification boundary possesses a simple geometry

The **decision boundary** is determined at the point where the probability of belonging to class 1 is equal to that of class 0.

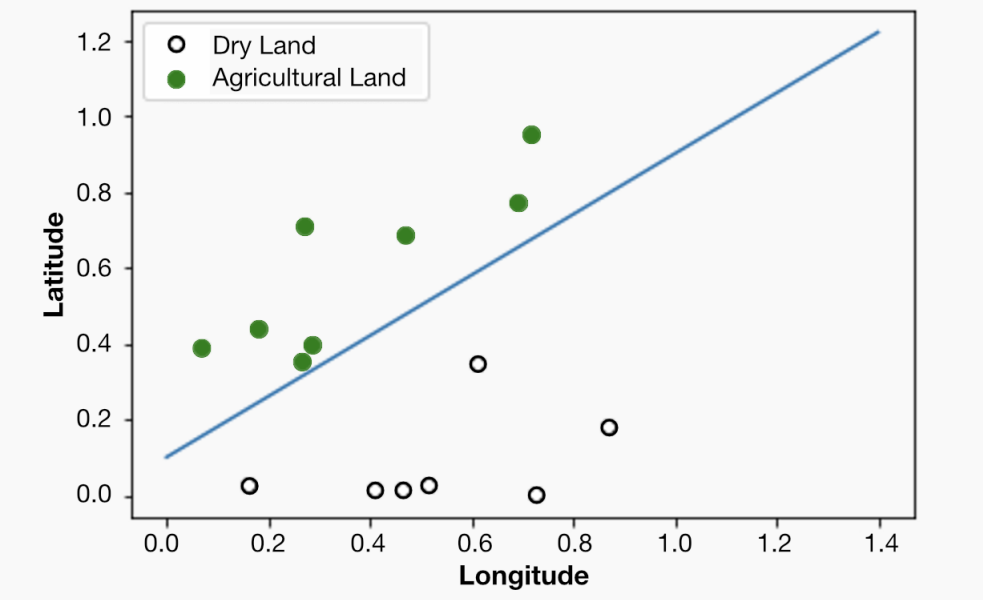
$$P(Y = 1) = 1 - P(Y = 0)$$

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

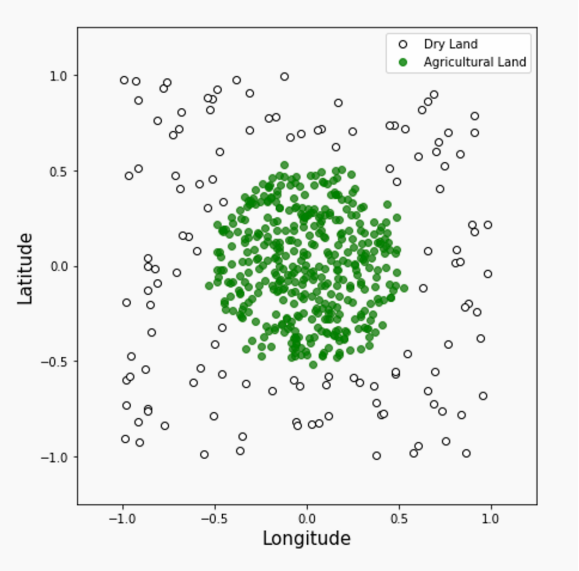
This is equivalent to the scenario where the log-odds are zero. The log-odds are defined as:

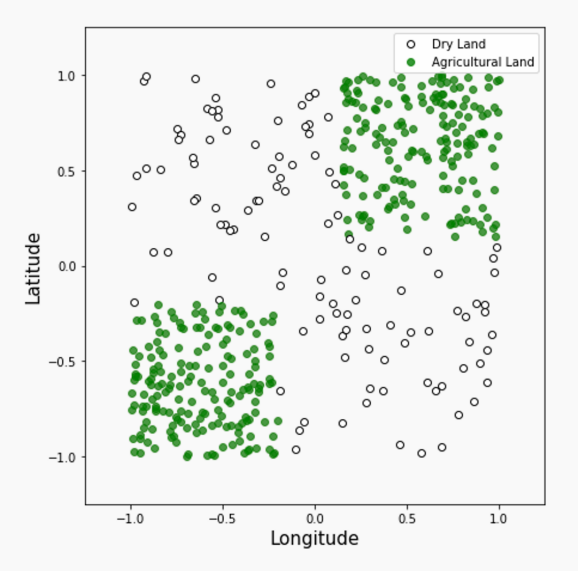
$$\log \left[\frac{P(Y = 1)}{1 - P(Y = 1)} \right] = X\beta = 0$$

The equation $X\beta = 0$ deine an hyperplane, but can be **generalized** using higher order polynomial terms to specify a non-linear boundary hyperplane.



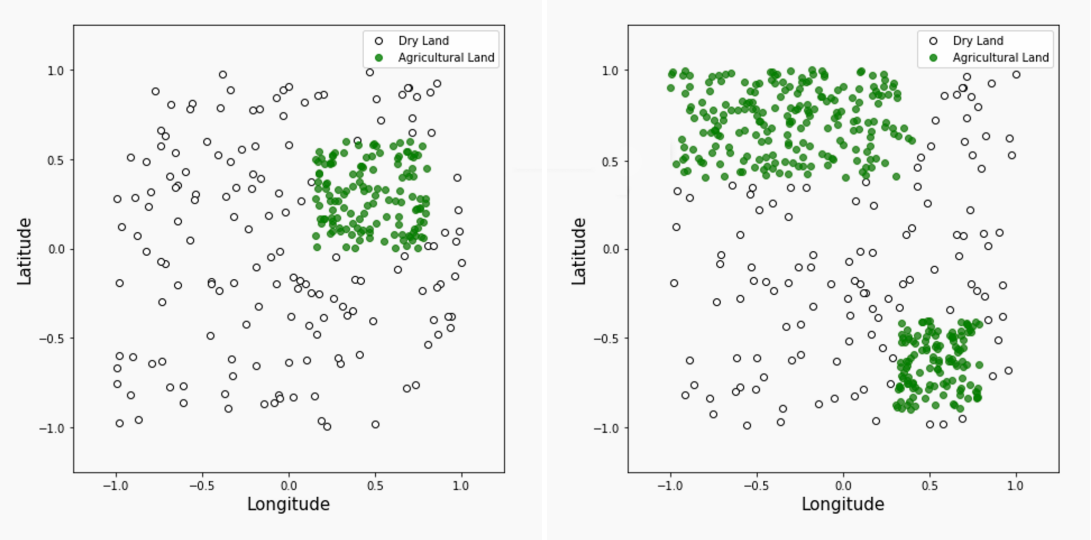
In this case the blue line can be easily described as $(y = 0.8x + 0.1)$ or $-0.8x + y = 0.1$ (assuming that, given their position as coordinates we call *longitude* as x and *latitude* as y).





In the first figure, we see that we can make a circular bounding box whereas in the second figure it is likely that we can make two square bounding boxes. However, as the geometric shapes in the figures become more complicated, determining the appropriate bounding boxes becomes increas-

ingly less straightforward and more complex.



Observe that in all the datasets, the classes remain well-separated in the feature space, **yet the decision boundaries cannot be easily described using a single equation**.

While 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 the same context.

1.1.2 Interpretable Models

People in every walk of life have long been using interpretable models for differentiating between classes of objects and phenomena.
Simple flow charts can be formulated as mathematical models for classification and these models have the properties we desire. Those properties are:

1. They possess **sufficiently complex** decision boundaries, and
2. They remain **interpretable** by humans.

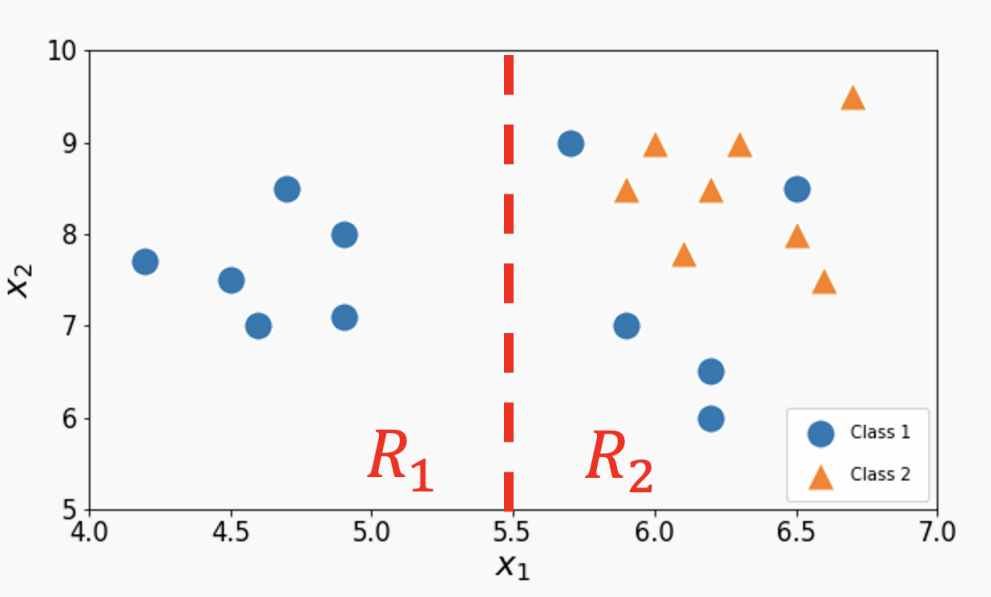
In addition, an important characteristic of these models is that their decision boundaries are locally linear. This means each segment of the decision boundary can be simply described in mathematical terms.

Geometry of Flow Charts Each flow chart node corresponds to a partitioning of the feature space — specifically, the height and width of the fruit — by axis-aligned lines or (hyper)planes.

Prediction Now, when presented with a data point, we can utilize its values to navigate or traverse the flow chart. This process will guide us to the model's predicted classification outcome - in other words, it leads us to land on a leaf node.

1.1.3 Classification Error

To understand the concept of classification error as a splitting criterion, consider the following figure.



Consider the region R_2.
We define Classification error as $\frac{error}{total}$,

$$= \frac{\text{number of minority data points}}{\text{total number of data points}}$$

$$= \frac{\text{number of majority data points}}{\text{total number of data points}}$$

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

In addition, when considering the region R_2,

$$1 - \frac{\text{number of majority data points}}{\text{total number of data points}}$$

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

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

The table below shows how the classification error for each region is calculated.

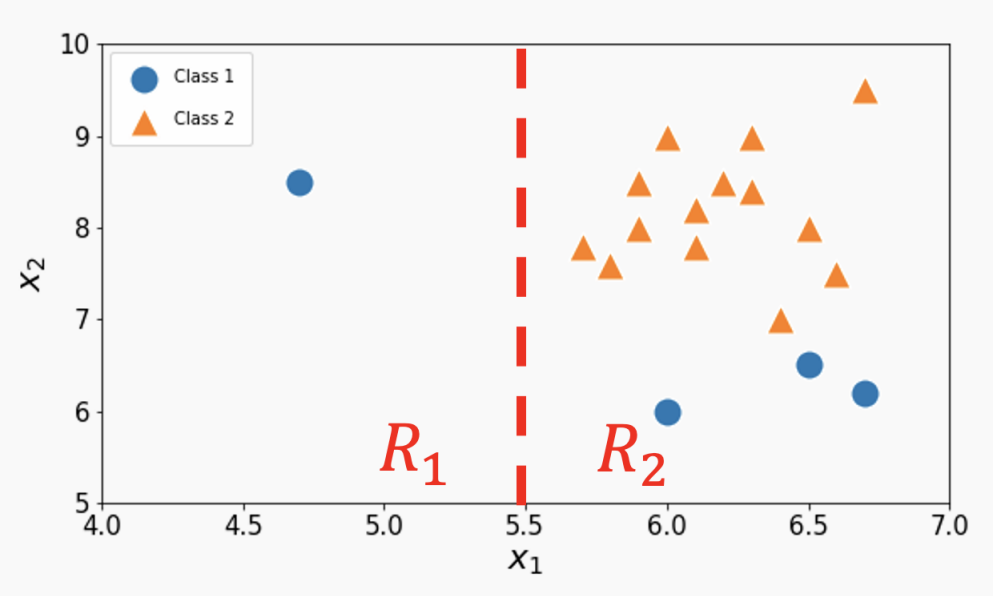
	◯	△	Error=$1 - max_k(\Psi(k\|R_r))$
R_1	6	0	$1 - max(\frac{6}{6}, \frac{0}{6}) = 1 - 1 = 0$
R_2	5	8	$1 - max(\frac{5}{13}, \frac{8}{13}) = \frac{5}{13} \approx 0.38$

In general: Assume we have P predictors and K classes. Suppose we select the pth predictor and split a region along the threshold t_p.

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

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

Where $\Psi(k|R_r)$ is the proportion of training points in R_r that are labeled class k. Here's another example:



	◯	△	Error $= 1 - max_k(\Psi(k\|R_r))$
R_1	1	0	$1 - max(\frac{1}{1}, \frac{0}{1}) = 1 - 1 = 0$
R_2	3	14	$1 - max(\frac{3}{17}, \frac{14}{17}) = \frac{3}{17} \approx 0.18$

We need to calculate the **weighted average** over both regions, taking into consideration the number of points in each region, and then minimize this weighted average over the parameters p and t_p:

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

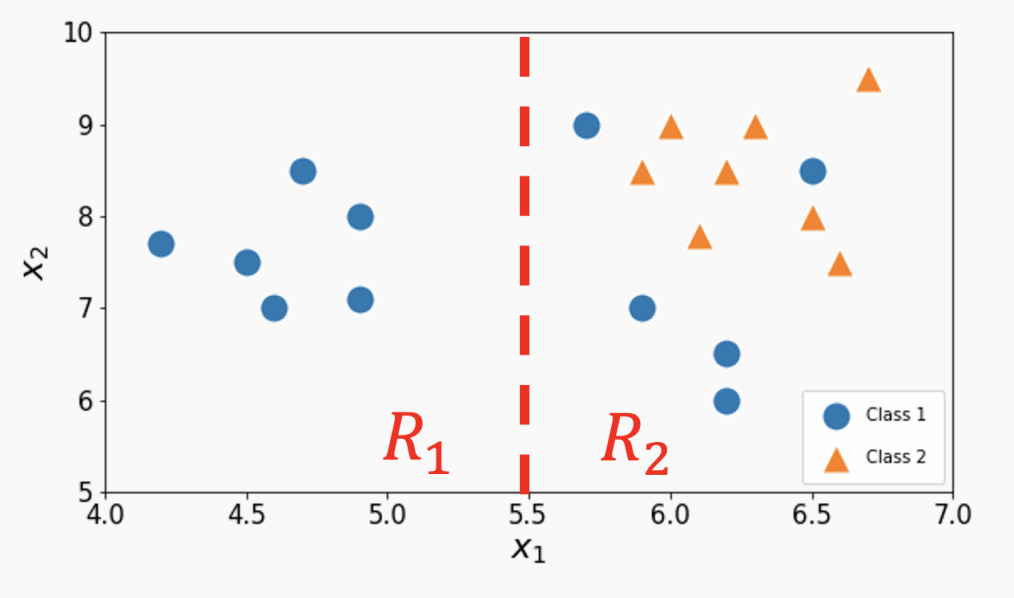
Where N_r is the number of training points inside of the region R_r.

1.1.4 Gini index

Assume we have P **predictors** and K **classes**. Suppose we select the pth predictor and split a rgion along the **threshold** $t_p \in \mathbb{R}$.

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

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



In the image above, we see that we have a certain number of each type of point in the specified regions based on our deliminator. We can then use these point tallies to deterime the index values.

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

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

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

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

1.1.5 Information Theory

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

What is entropy? In this course we are using the information theory definition. Entropy is a fundamental concept in information theory that quantifies the average amount of information or uncertainty associated with a random variable or message.
For a random variable with a discrete distribution, the entropy is computed by:

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

Entropy is therefore a value that describes how "well-mixed" or "well-separated" a data distribution is. Highly mixed distributions will have entropy near 1. Distributions with well-separated categories will have entropy near 0.

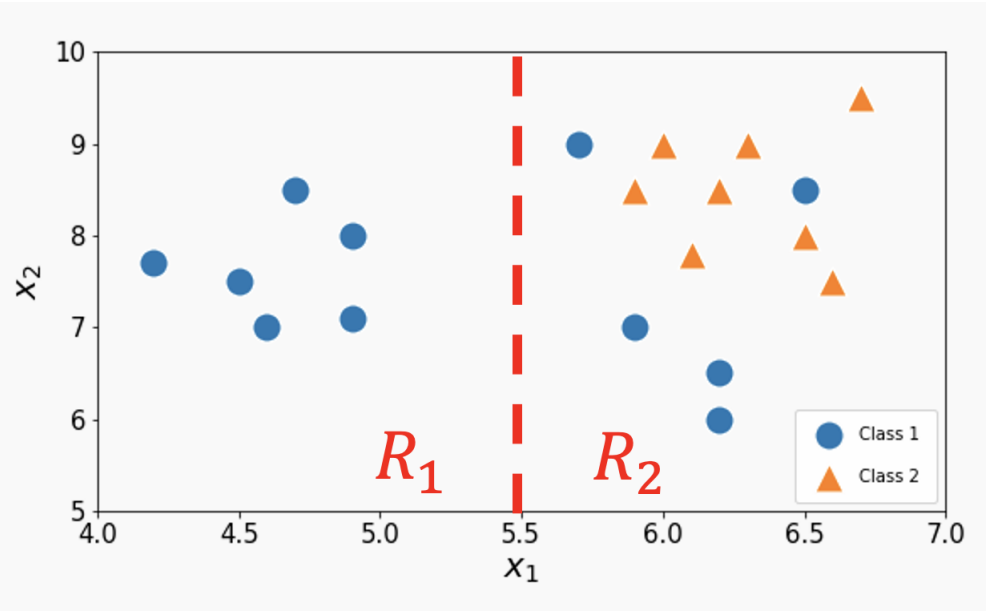
1.1.6 Entropy

Assume we have P **predictors** and K **classes**. Suppose we select the k-th predictor and split a region along the threshold $t_p \in \mathbb{R}$.

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 $= -\sum_k \Psi(k\|R_r) \log_2 \Psi(k\|R_r)$
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	$\left[\frac{5}{13}\log_2\frac{5}{13} + \frac{8}{13}\log_2\frac{8}{13}\right] \approx 1.38$

The entropy calculation here yields a value of 1.38, compared to a misclassification rate of 0.38 and a Gini index of 0.47.

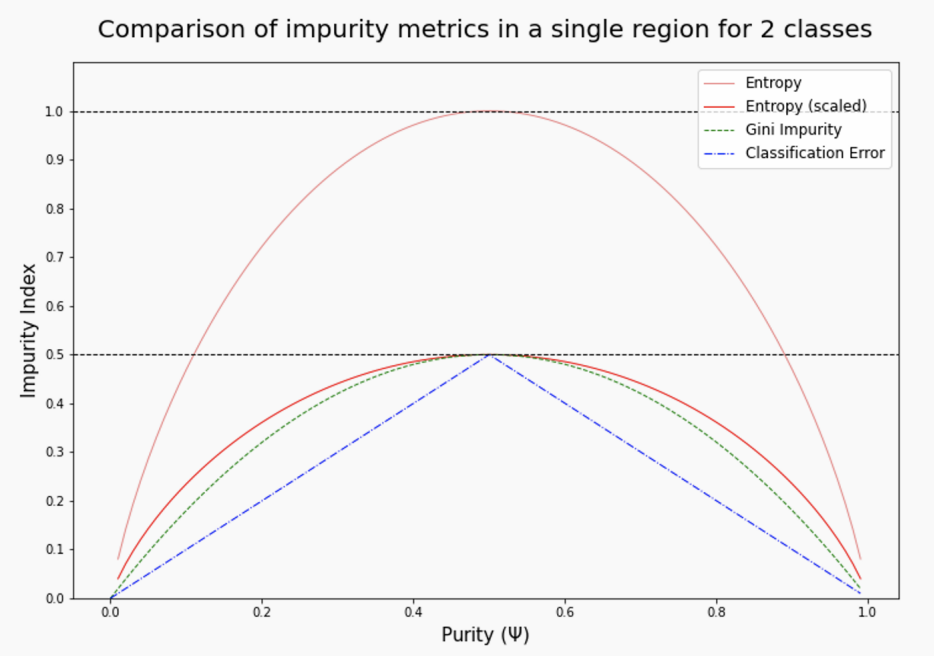
We can now try to find the predictor p and the threshold t_p minimizes the weighted average entropy over the two regions:

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

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

1.1.7 Comparison of Criteria

The graph below shows how our various impurity metrics compare for a fairly simple region with two classes. On the bottom is the purity Ψ. You can see that all three measures reach a maximum around purity 0.5, but they take different paths to get there. For entropy and Gini impurity, they are curved (concave downward), while the classification error is a straight line from zero at both ends of the axis to a maximum in the center. None of these are "better", they just give different weights to different values of Ψ.



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 that works as follows:

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 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 (Bayes' classifier).

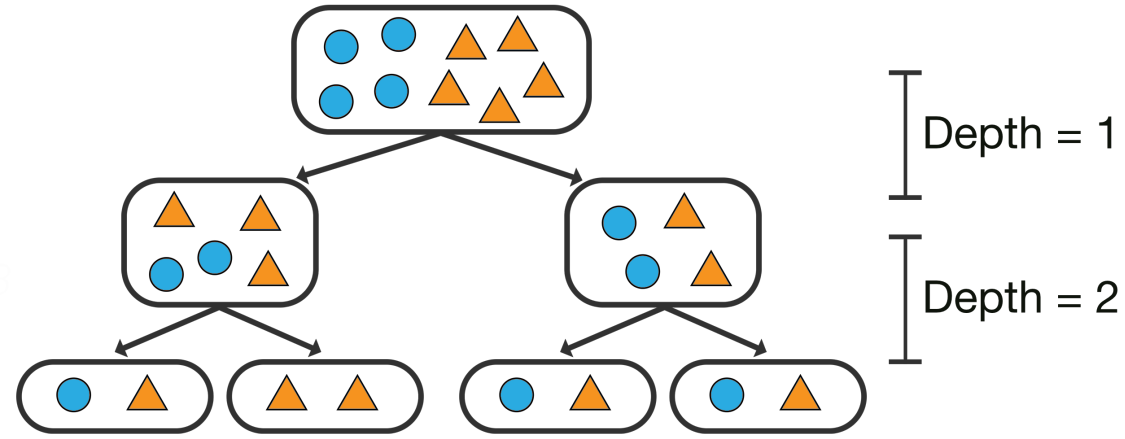
This is not guaranteed to give us the best decision tree (for any particular definition of "best"), but it is very likely to give us a good one.

1.2 Stopping Conditions

1.2.1 Common Stopping Conditions

The most common stopping criterion involves restricting the **maximum depth** (`max_depth`) of the tree.

The following diagram illustrates a decision tree trained on the same dataset as the previous one. However, a `max_depth` of 2 is employed to mitigate overfitting.

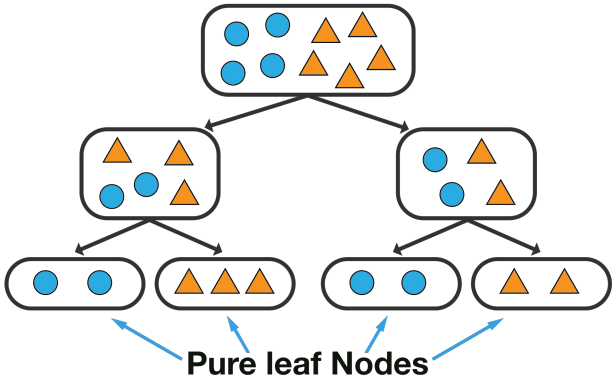


Four other common simple stopping conditions are:

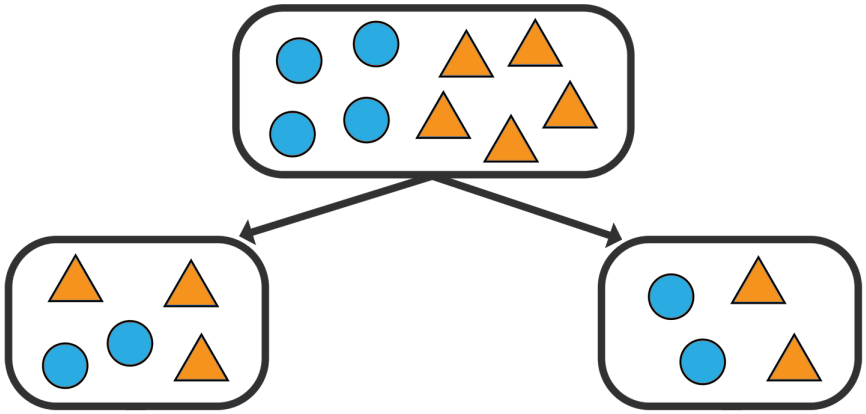
1. Do not split a region if all instances in the region **belong** to the same class.
2. Do not split a region if it would cause the number of instances in any sub-region to go below a pre-defined threshold (`min_samples_leaf`).
3. Do not split a region if it would cause the total number of leaves in the tree to exceed a pre-defined threshold (`max_leaf_nodes`).
4. Do not split if the gain is less than some predefined threshold (`min_impurity_decrease`).

Let's look at each one individually.

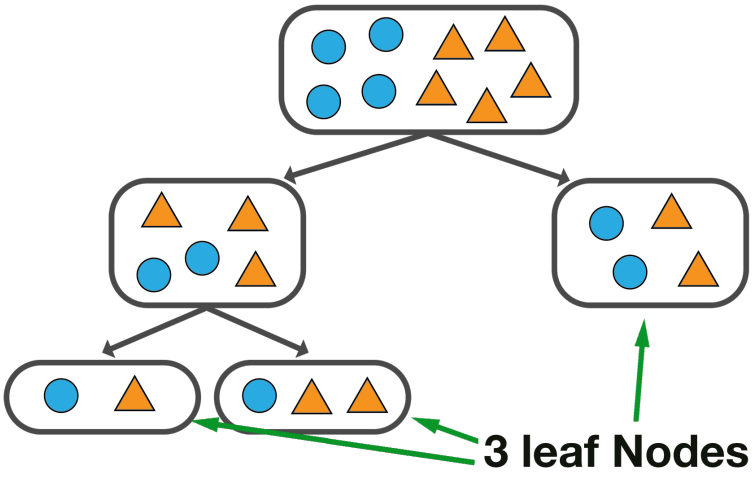
1. Do not split a region if all instances in the region belong to the same class. The diagram below displays a tree where each of the end leaf nodes are clearly of the same class, therefore we can stop at this point in growing the tree further.



2. Do not split a region if it would cause the number of instances in any sub-region to go below a pre-defined threshold (`min_samples_leaf`). In the following diagram, we set `min_samples_leaf` to be 4, and we can observe that we can't split the tree further because each leaf node already meets the minimum requirement.



3. Do not split a region if it would cause the total number of leaves in the tree to exceed a pre-defined threshold (`max_leaf_nodes`). In the diagram below, we observe a tree with a total of 3 leaf nodes, as specified as the maximum.



4. Do not split if the gain is less than some pre-defined threshold (`min_impurity_decrease`). Compute the gain in purity of splitting a region R into R_1 and R_2:

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

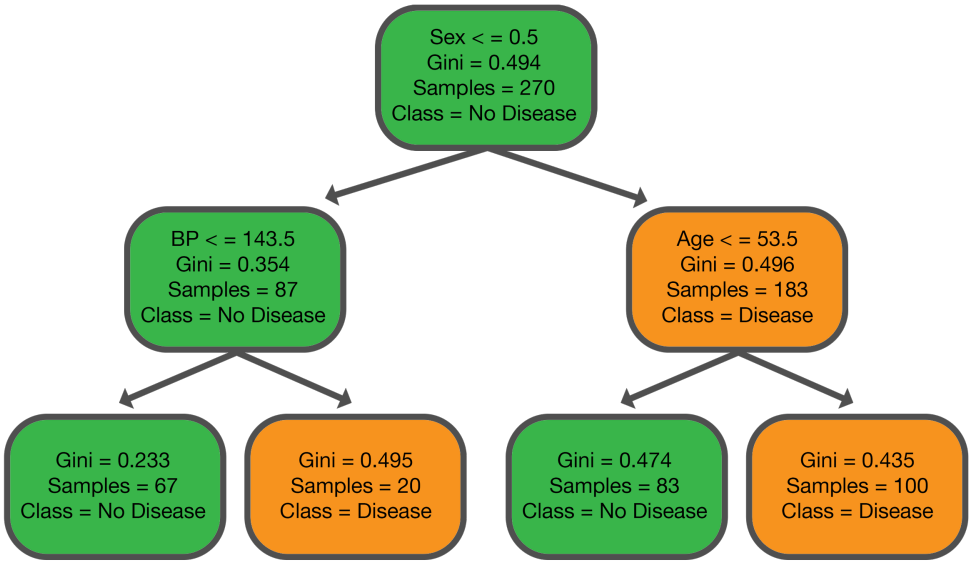
Where M is the classification error, Gini index or Entropy, depending on which is chosen to use.

Scikit-learn (`sklearn`) grows trees in **depth-first** fashion by defaul, but when `max_leaf_nodes` is specified it is grown in a best-first fashion.

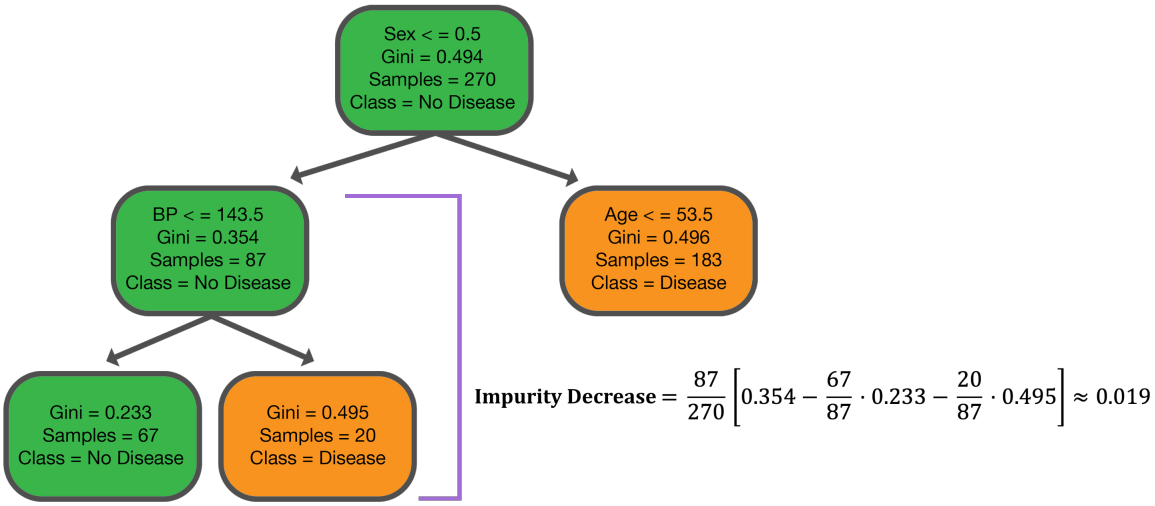
1.2.2 Depth-first and Best-first

To illustrate the difference between depth-first and best-first, we'll consider the following decision trees that predicts if a person has heart disease based on age, sex, BP and cholesterol.

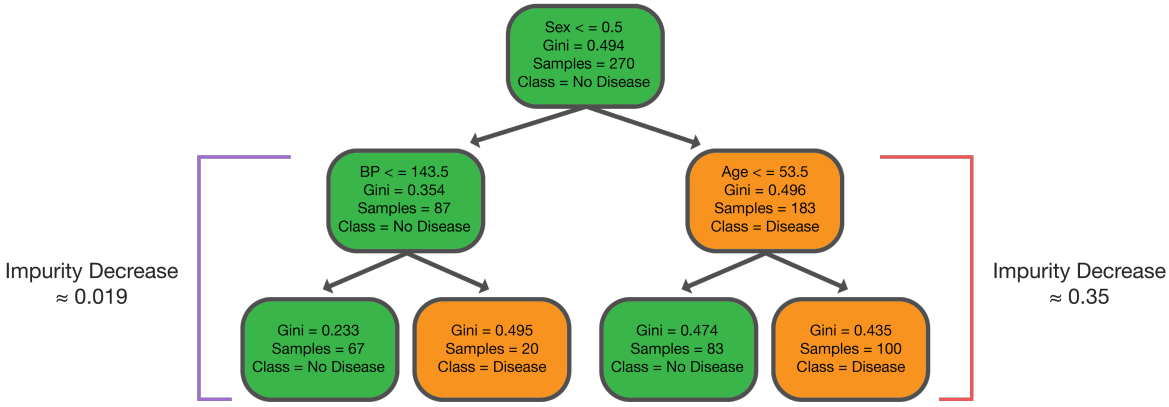
Depth-first Growth For depth-first, sklearn determines the best split based on impurity decrease, with the greatest amount of decrease being the best choice. Here we see a tree with `max_depth=2`.



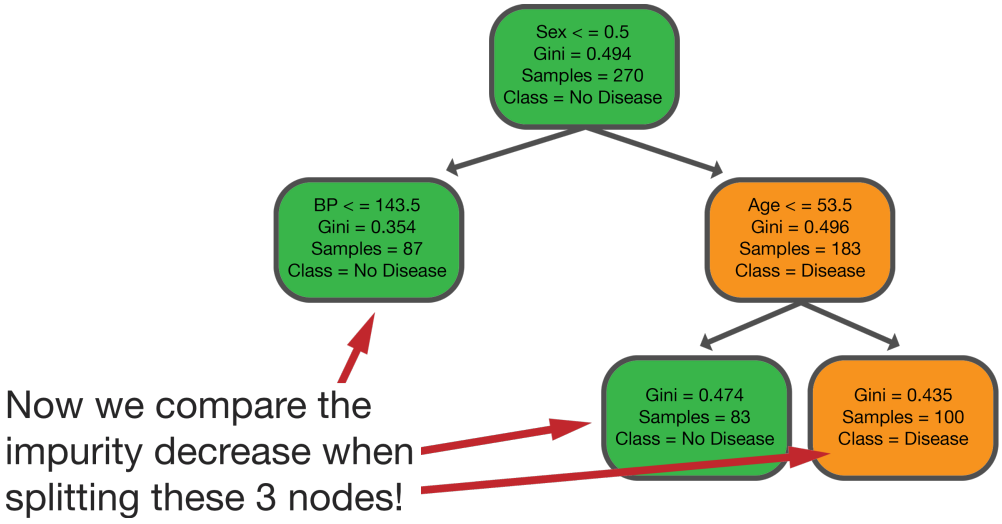
Best-first Growth Here we see a series of images showing the growth of the best first tree. Each time we calculate the impurity decrease for the threshold to determine when to stop. For the first diagram we see that the impurity decrease is minimal.



In our second split, the impurity decrease is slightly larger as compared to the split from before. We want to split where the decrease is greater; therefore, we choose the second split.

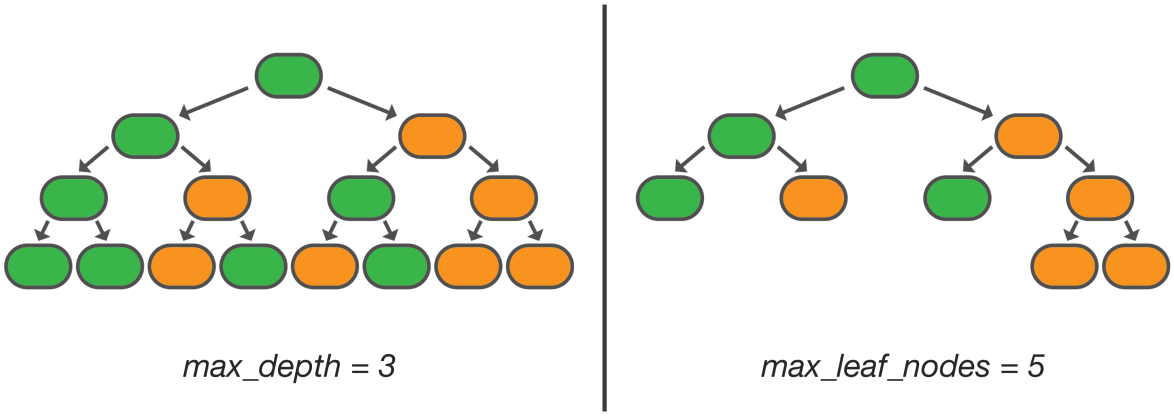


Once we have chosen the appropriate split, we can now delibrate the next split given the remaining nodes to compare. In the diagram below we see the next nodes we will compare the splits for.



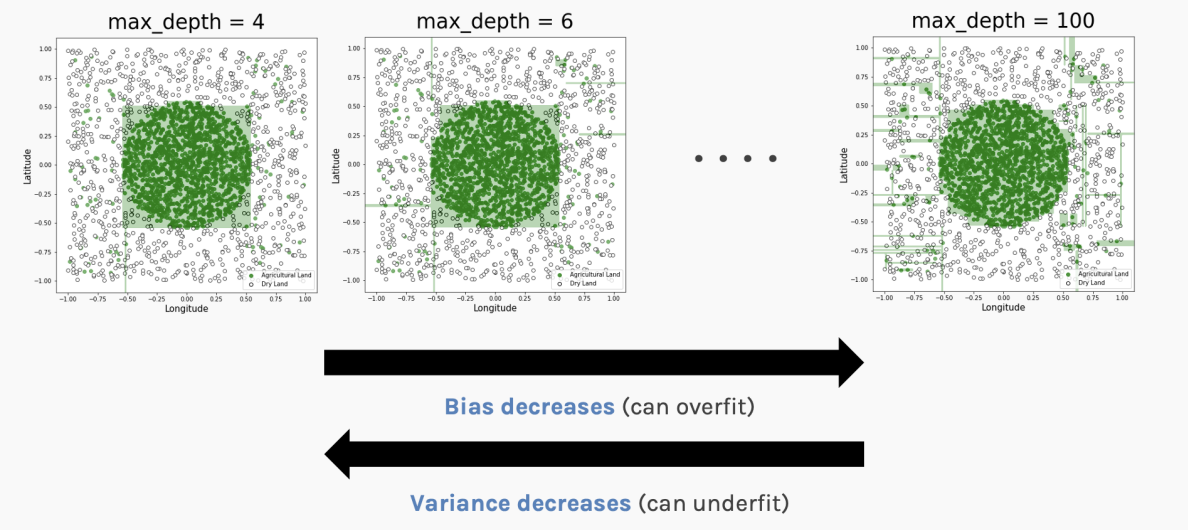
This process is repeated until `max_leaf_nodes` is reached.

Depth-first vs. Best-first The following is a comparison between depth-first or best-first growth. The image on the left shows a stopping condition for the depth. The right image has a constraint on the number of leaf nodes, which is a breadth-first stopping condition.



1.2.3 Variance vs. Bias

How do we decide what is the appropriate stopping condition?



Complex trees are also harder to interpret and more computationally expensive to train. This gives us a bias-variance tradeoff, like you saw in our previous course. Let us revisit these concepts in the context of decision trees.

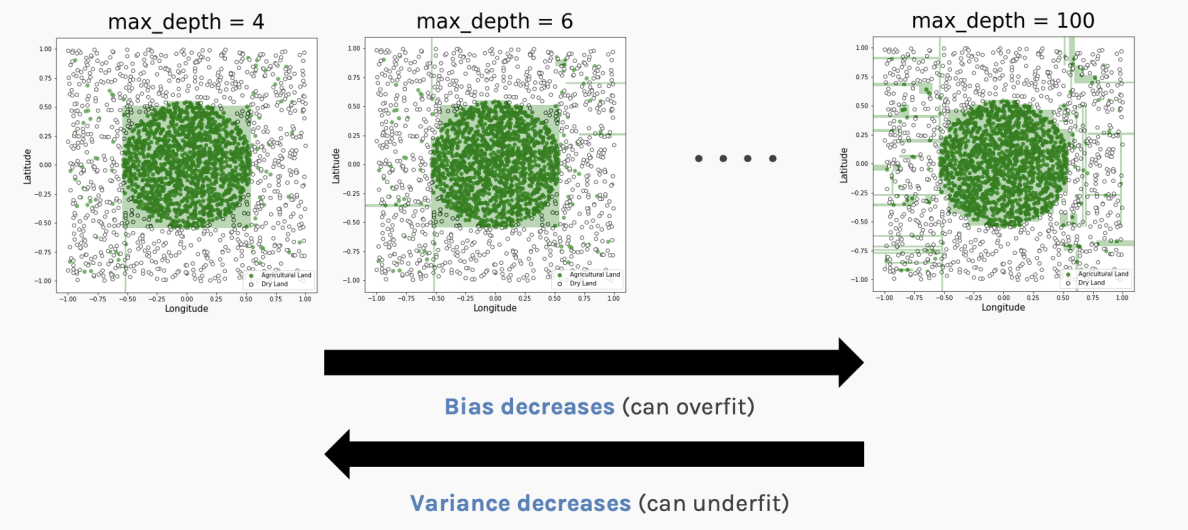
High Bias: Trees of low depth are not a good fit for the training data - it's unable to capture the nonlinear boundary separating the two classes.

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.

The image below gives us a visual example. On the left we have a circular area of green data points that we're trying to model with a tree of depth 4. It's not going to be very good - it's essentially fitting a round peg in a square hole. As the depth increases, we get a better fit, but you can see more and more places where the model predicts there should be green data points for no discernible reason. As we get to depth 100, the prediction is very messy, with "false positives" everywhere. We've committed the classic mistake of over-fitting our data, capturing every single point in the training set while making our future predictions worthless.



How Can We Determine the Appropriate Hyperparameters?

Cross-Validation is the key As always we rely on CV to find the optimal set of hyper-parameters.