Foundation of Data Science
Lecture 5, Module 1
Spring 2022

Rumi Chunara, PhD

Today

- Supervised Learning Algorithms
  - Decision Trees
  - Ensemble Methods
  - kNN

Decision Trees

Recap...

If we have a target in our dataset, we often wish to:

1. Determine whether a feature contains important information about the target (in other words, does a given feature reduce the uncertainty about the target?)

2. Obtain a selection of the best features for predicting the target

3. Rank each feature on its ability to predict the target

Information Theory

Fundamental tools from information theory that are the foundation for Decision Trees:

- **Entropy:** the average amount of information that is encoded by a random variable X ($H(X)$)

- **Conditional Entropy:** Given that we know X, the amount of extra information needed to encode Y ($H(Y|X)$)

- **Information Gain:** How much new information do we gain on Y by conditioning on X? **Today**

Information Gain (IG)

The Information Gain (IG) of an attribute *a* measures the change in conditional entropy for the target *T* if we split the data by *a.*

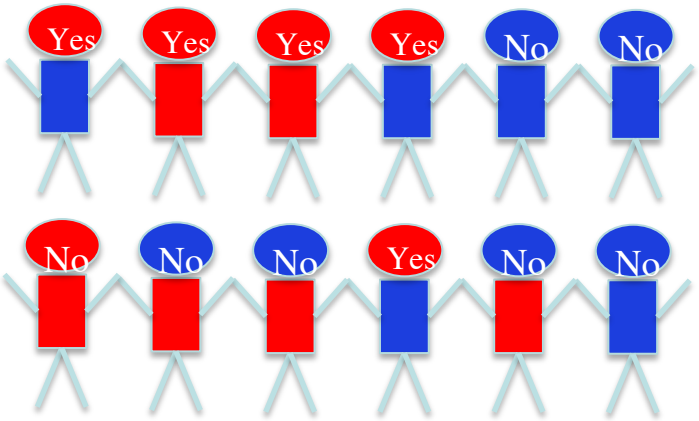
- IG(*T*, *a*) tells use how pure we can make segments of the population, with respect to the target *T*, if we split it by *a*

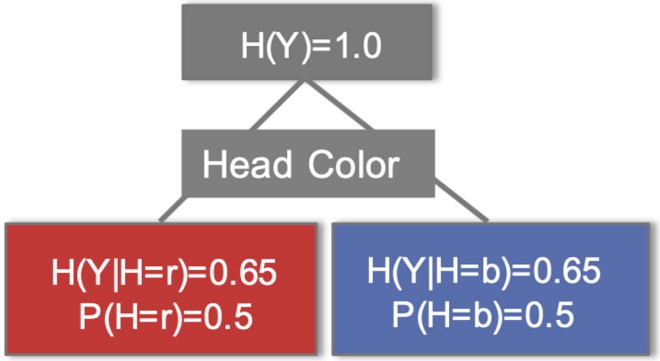
$$IG(T, a) = \mathrm{H}(T) - \mathrm{H}(T|a)$$

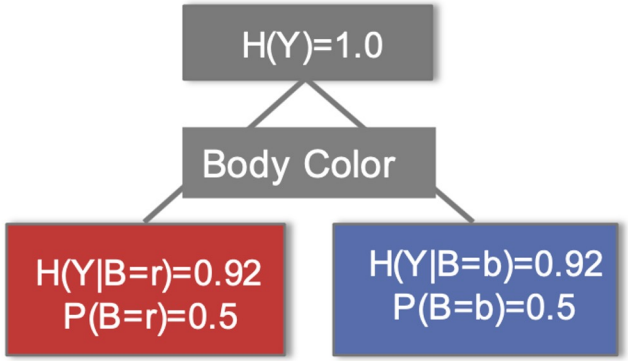
https://en.wikipedia.org/wiki/Information_gain_in_decision_trees

Information Gain - Example

Y = {yes, no}
H = {b, r}
B = {b, r}







IG(Y, H) = 1 - (0.5*0.65 + 0.5*0.65) = **0.35**
IG(Y, B) = 1 - (0.5*0.92 + 0.5*0.92) = **0.08**

We get a higher IG when splitting on **Head (i.e., H(Y|H) < H(Y|B))**

C4.5 - Decision Tree Algorithm

Given a dataset ***D=<X,Y>***, where ***Y*** is a target and ***X=<X1,X2,..Xk>*** is a k-dimensional vector of attributes, we have:

C4.5 pseudocode

1. For each feature x_i in X:
   - Compute split that produces highest Information Gain
   - Record Information Gain

2. Let X^{max} be the feature with the highest Information Gain

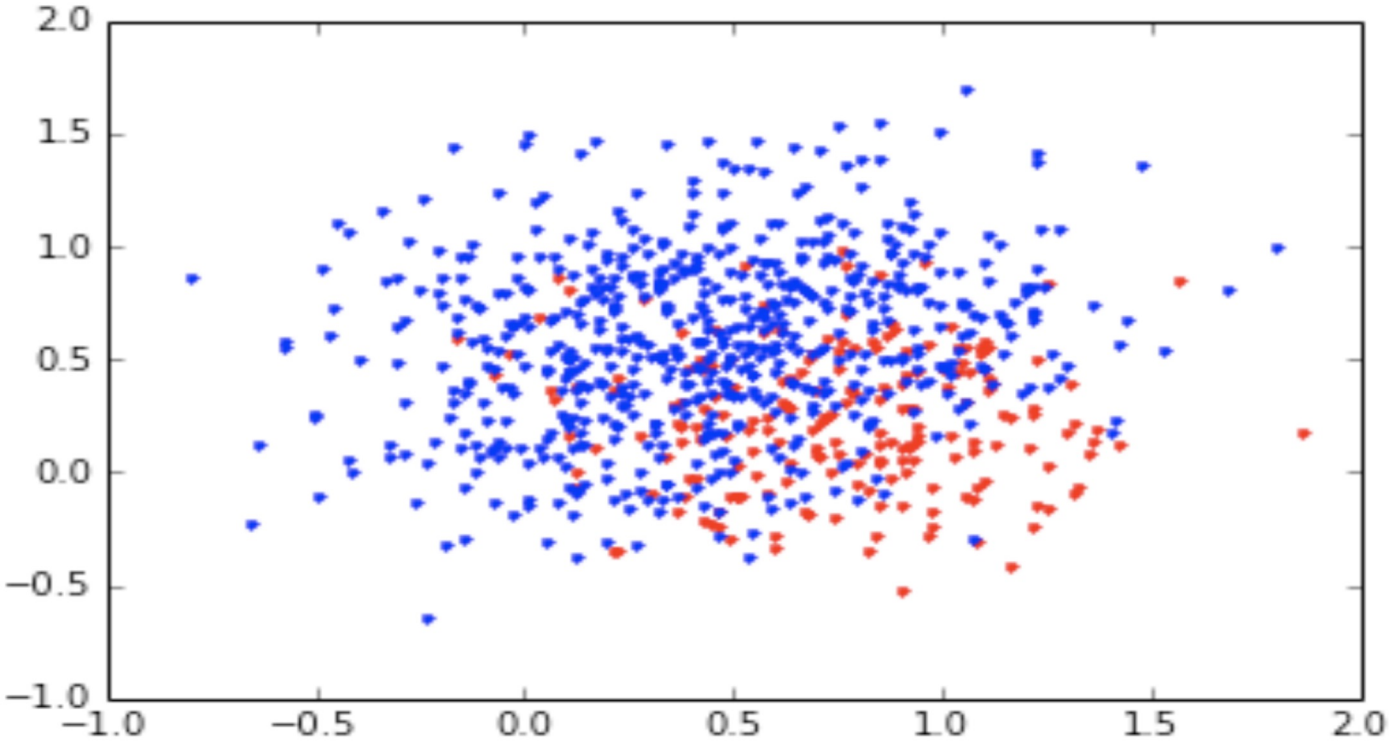
3. Create child nodes that split on the optimal splitting of X^{max}

4. Recurse on each child node from step 3 on the remaining features.

Building a Decision Tree

- 800 samples drawn from bivariate Gaussian distributions with the following means: [0.25,0.75], [0.75,0.75], [0.75,0.25], [0.25,0.25] **We are simulating two features!**
- The samples drawn from the distribution with means [0.75, 0.25] were labeled as 'red'; the others, as 'blue' **Binary target!**

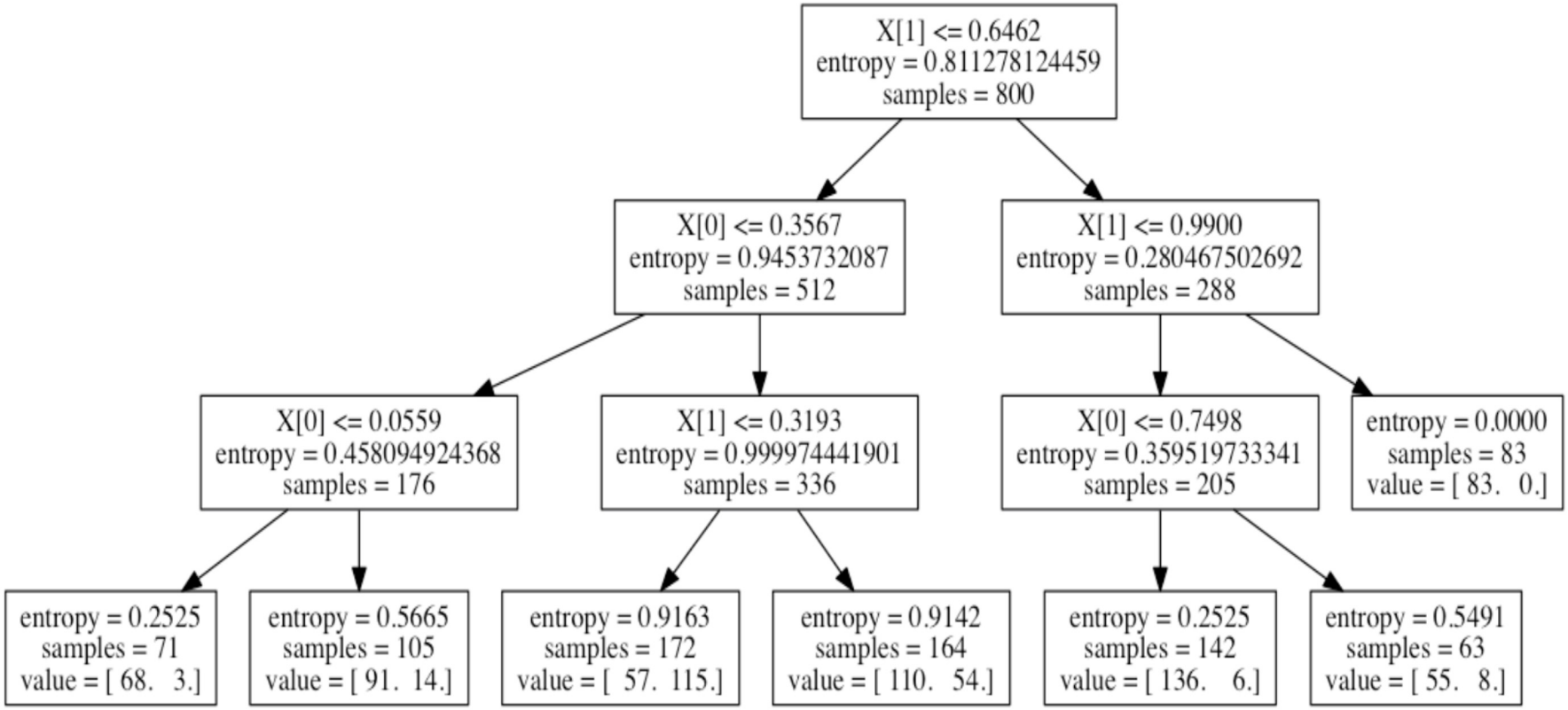
X1



X0

Decision Trees with Scikit-Learn

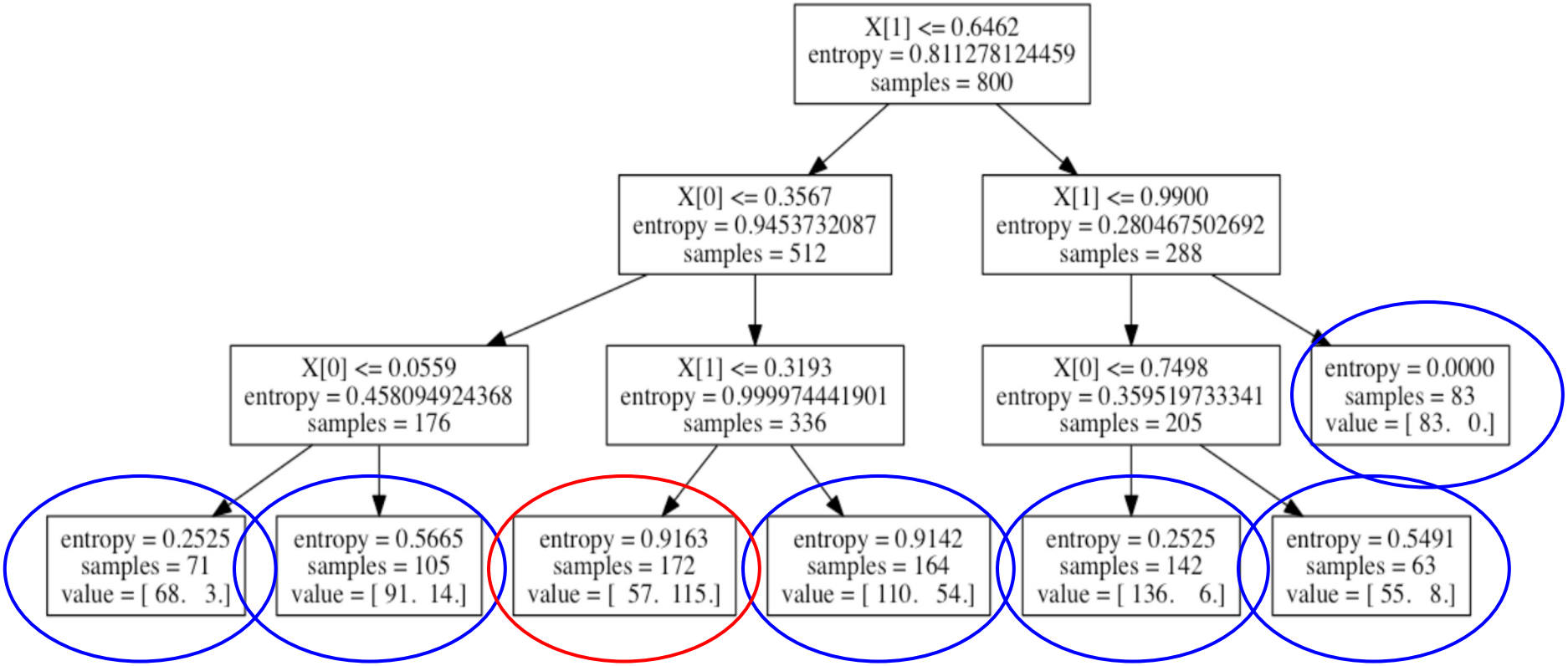
```
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(args)
clf = clf.fit(X,y)
```



If you have a lot of features or data, trees can get pretty large. In such cases, visualizing them is probably not worth it!

Understanding the Decision Tree

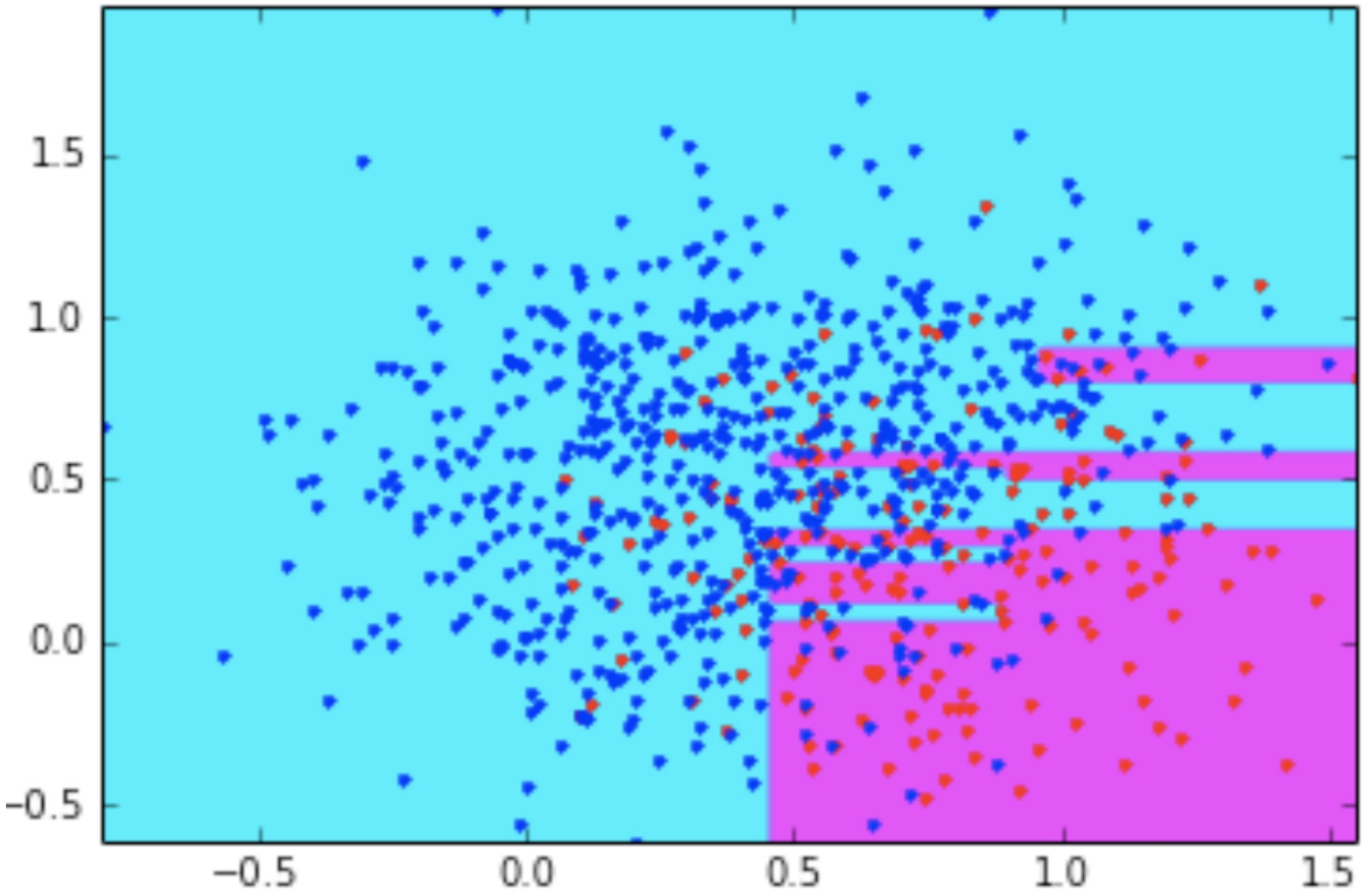
- Each parent node references the feature chosen by the splitting algo
- The node specifies a boolean condition: if True, then move to the right; else move to the left
- The final nodes (leaves) show the distribution of the target at that node
- The prediction is created by following feature values of the unseen instance



Partitioning the Feature Space

We can think of the **Decision Tree as a rectangular partitioning of the feature space.** Again, this is done with a series of boolean conditionals.

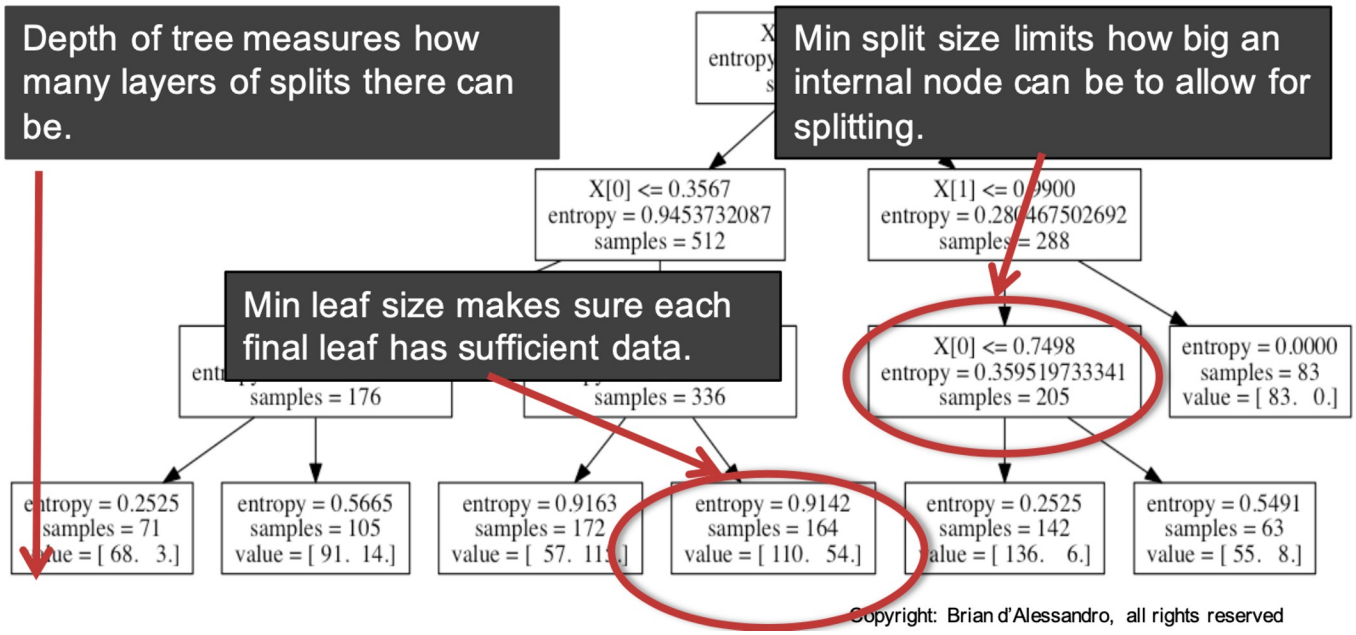
- For classification tasks, the partition determines the predicted class. This chart shows the decision boundaries for the training data



Controlling Complexity

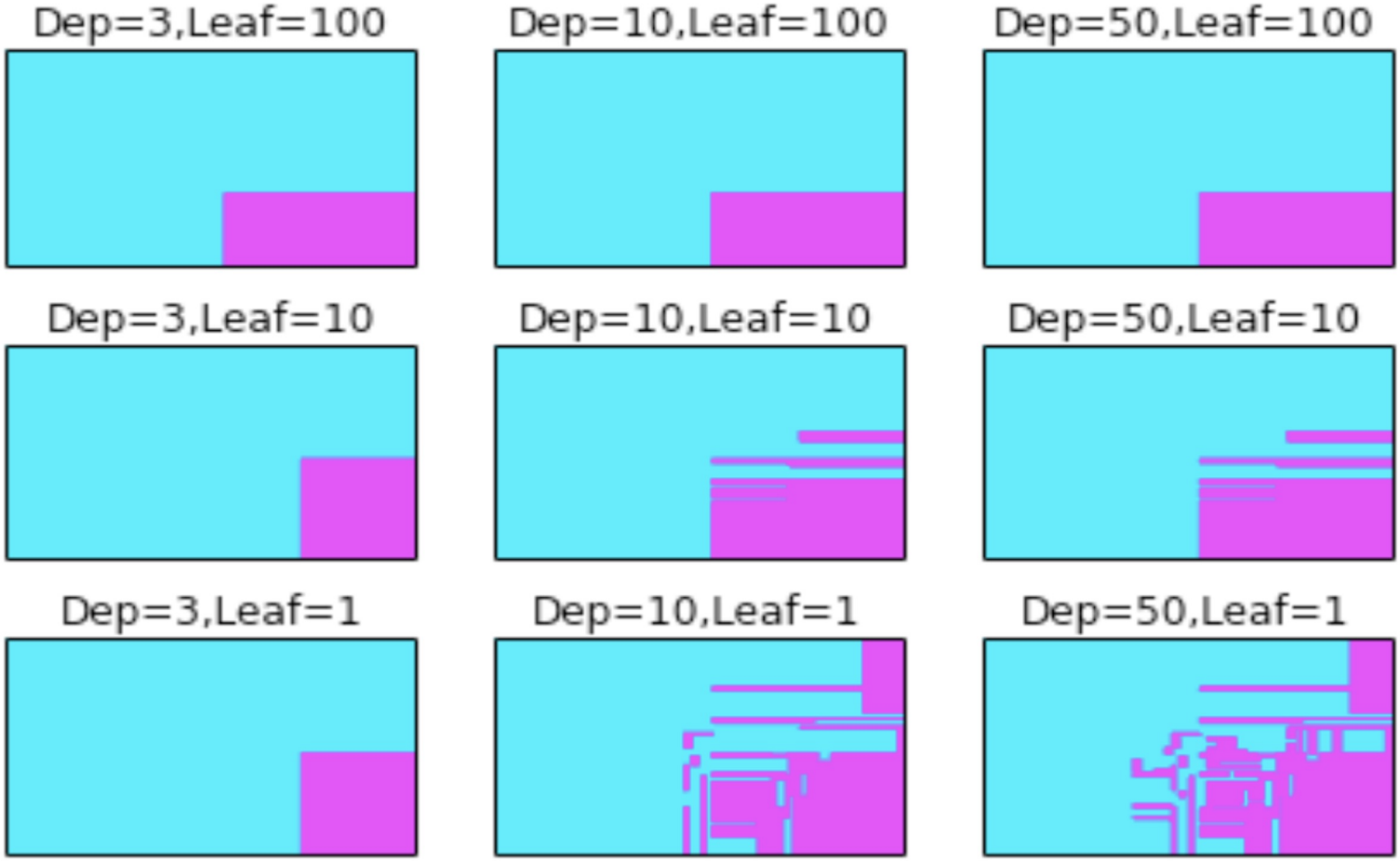
As the ratio of features to data points shrinks, we risk overfitting (learning something super specific to the training data that does not generalize to new instances)

- To avoid overfitting and high complexity models (which are also expensive), we can:
  - limit the depth of the tree
  - limit the size of internal nodes that can be split
  - specify a minimum number of instances per leaf



Controlling Complexity

Note how the partitions (decision boundary) change as we let trees grow arbitrarily complex. **Overfitting is likely to occur if we don't set any limits on the complexity.**



https://github.com/briandalessandro/DataScienceCourse/blob/master/ipython/Lectures/2020%20Intro%20to%20DS%20Module%209%20-%20Decision%20Trees.pdf

Decision Trees - Test Time

- In the previous slides, we covered the **training phase** of a Decision Tree, where it is built

- In test time, given a new instance, we:
  - get its feature values (note that this instance should have the same features of the training data ideally)
  - perform the tree tests over these values (run the instance through the tree)
  - determine the leaf in which the new instance resides
  - make a prediction for the new instance

Decision Trees - Advantages

- Easy to interpret (although not necessarily easy to visualize)
- Easy to implement - just a series of if-then rules
- Prediction is cheap once you get to the leaves
- No feature engineering (like normalizing) is needed
- Often handles both numerical and categorical data
- Detects non-linear relationships in the data (see how the boundaries can be complex)
- Works for both regression and classification problems

A very good baseline model for both regression and classification tasks!

Decision Trees - Disadvantages

- Easy to overfit – flexibility of algorithm requires careful tuning of parameters and leaf pruning
- Decision Trees are **greedy** - small changes in data can lead to very different solutions (can be adressed with bagging and boosting)
- Not good for problems where the number of samples shrinks as the number of features grows