Decision Trees

CS109A Introduction to Data Science

Pavlos Protopapas, Natesh Pillai

Outline

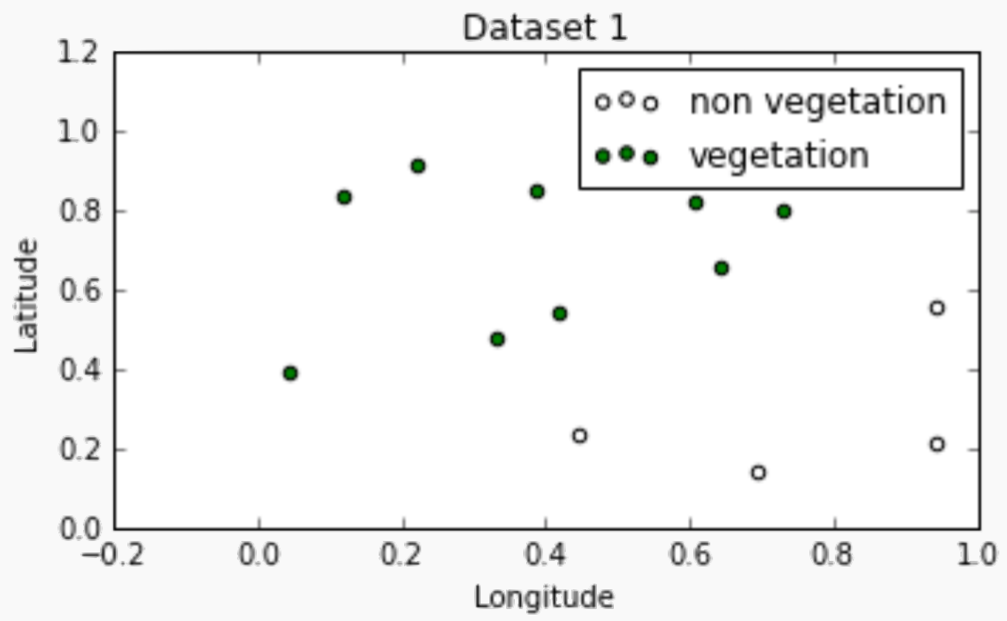
- Motivation
- Decision Trees
- Classification Trees
- Splitting Criteria
- Stopping Conditions & Pruning

Geometry of Data

Recall that:

logistic regression for building classification boundaries works best when:

- the classes are well-separated in the feature space
- have a nice geometry for the classification boundary



Geometry of Data

Recall that:

the decision boundary is defined where the probability of being in class 1 and class 0 are equal, i.e.

$$P(Y=1) = 1 - P(Y=1) \quad \Rightarrow \; P(Y=1) = 0.5$$

Which is equivalent to when the `log-odds=0`:

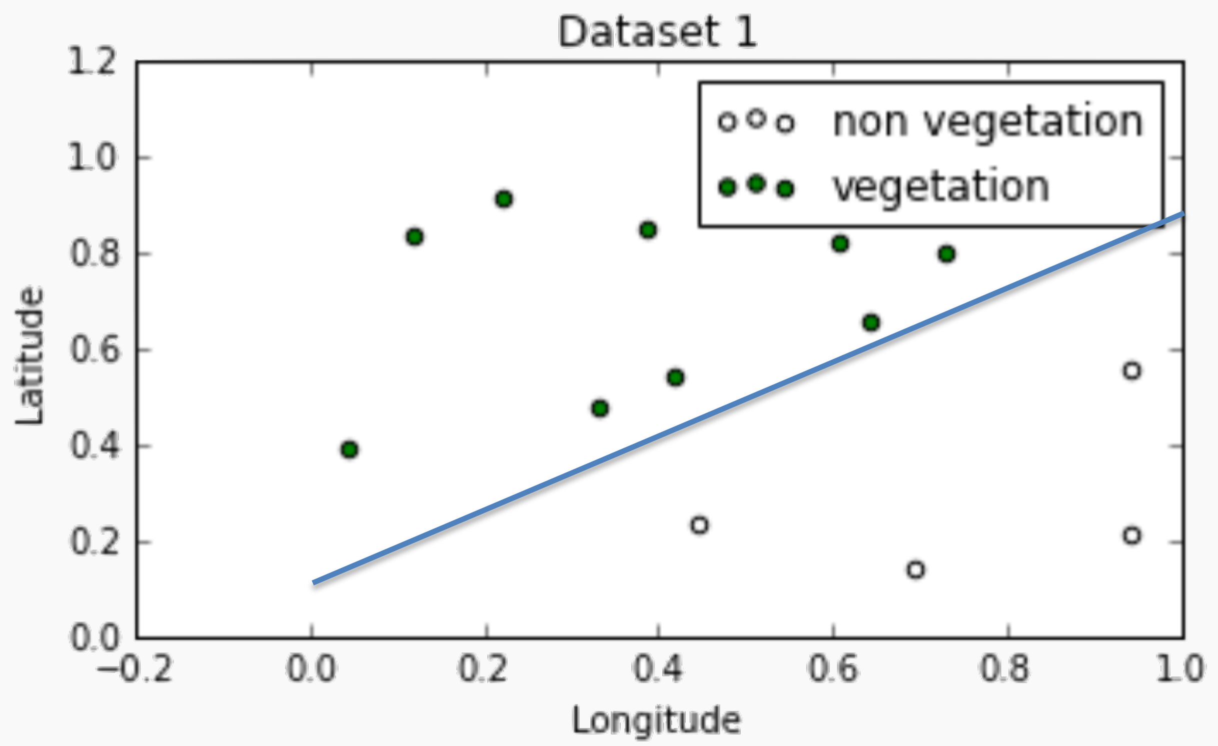
$$X\beta = 0$$

This equation defines a line or a hyperplane.

And it can be *generalized* with higher order polynomial terms.

Geometry of Data

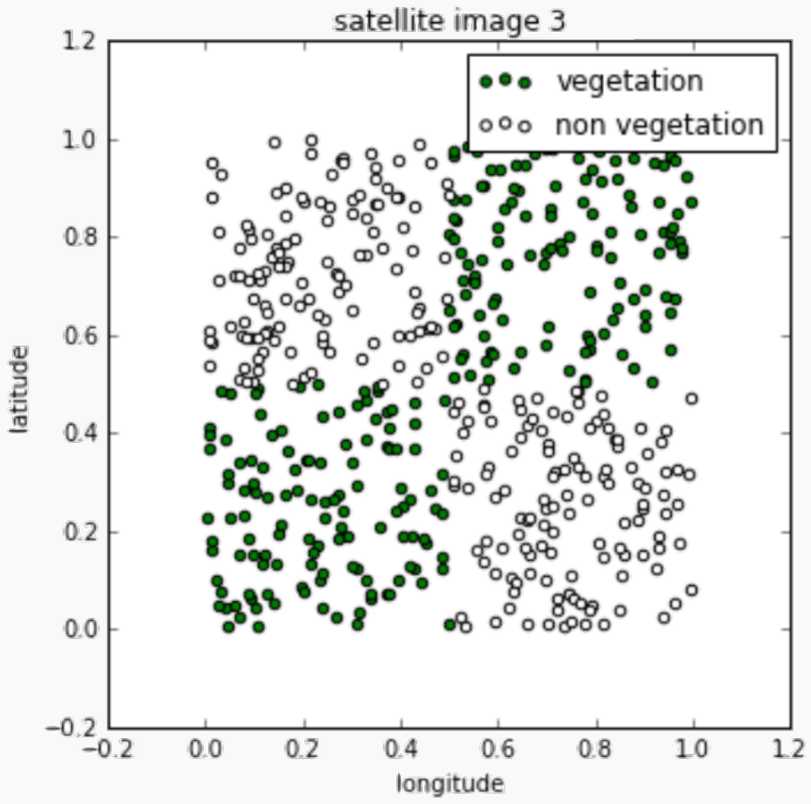
Question: Can you guess the equation that defines the decision boundary below?

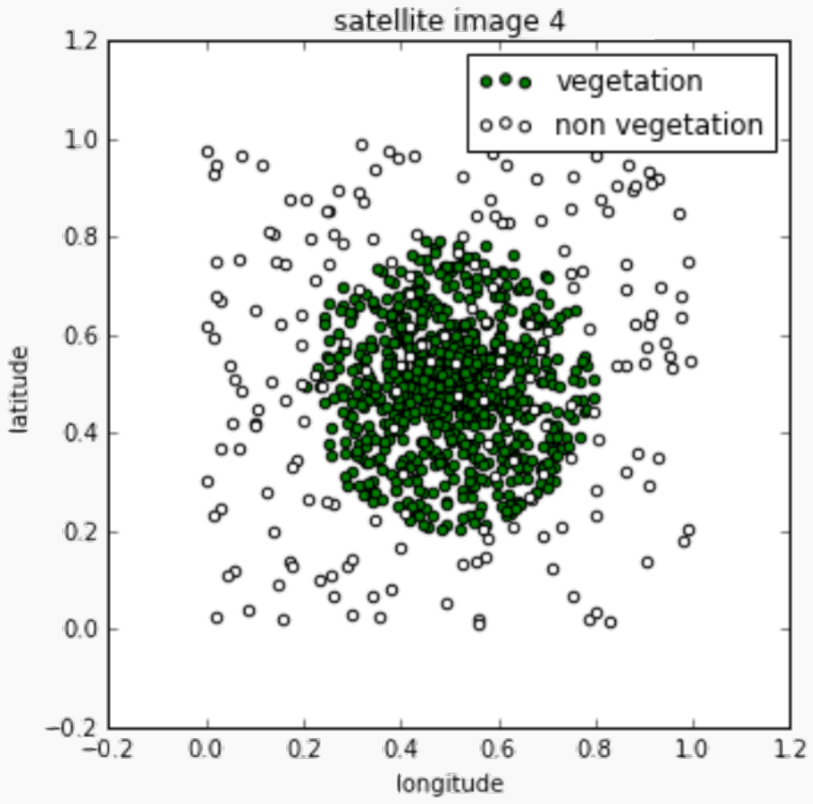


$$Latitude = 0.8\,Long + 0.1 \quad or \quad -0.8x_1 + x_2 - 0.1 = 0$$

Geometry of Data

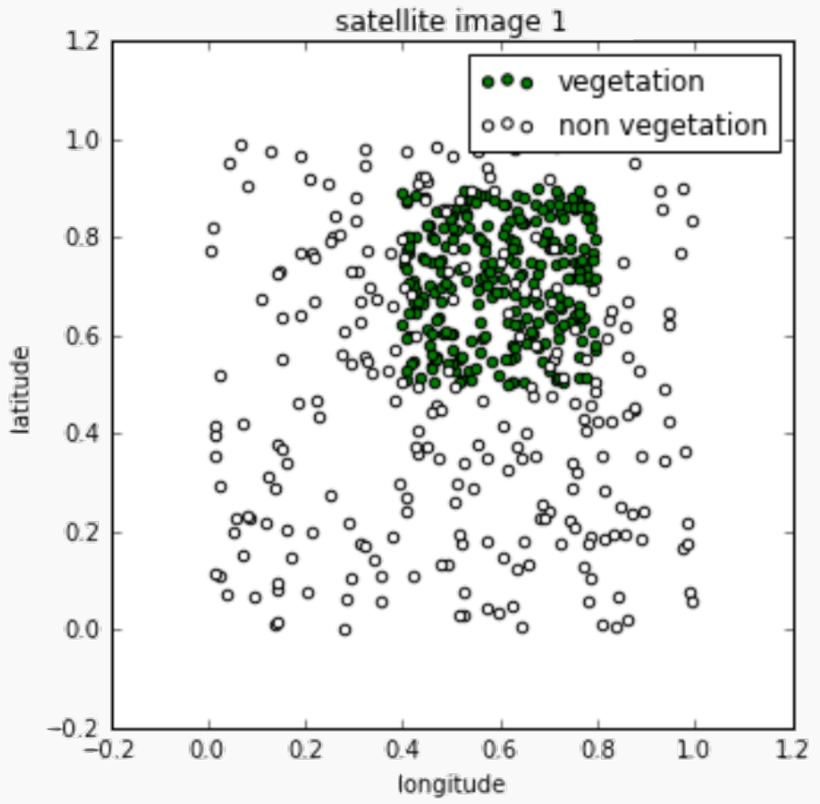
Question: How about these?

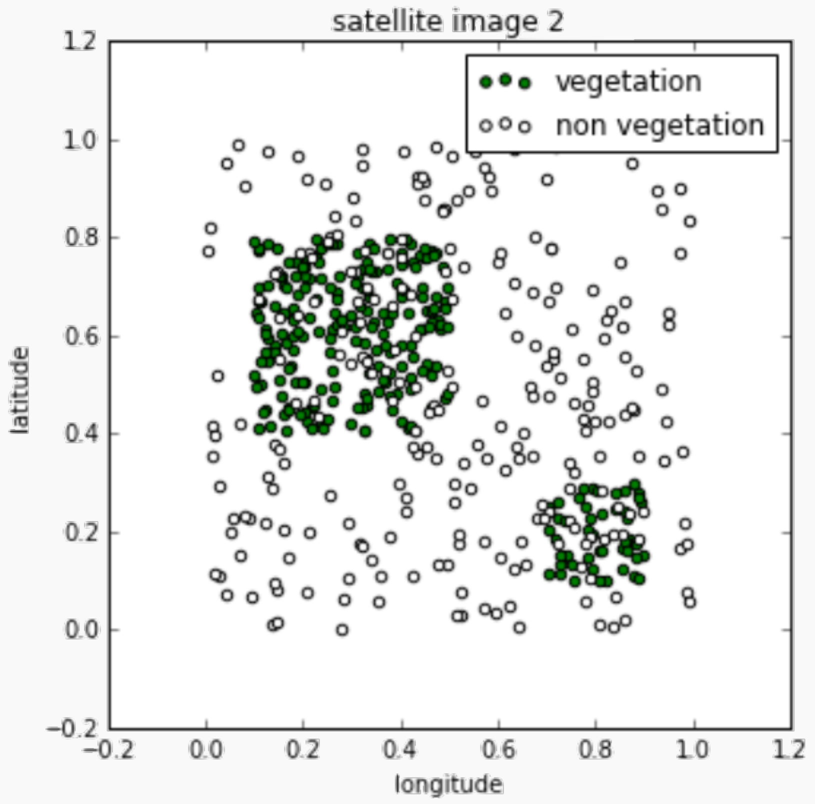




Geometry of Data

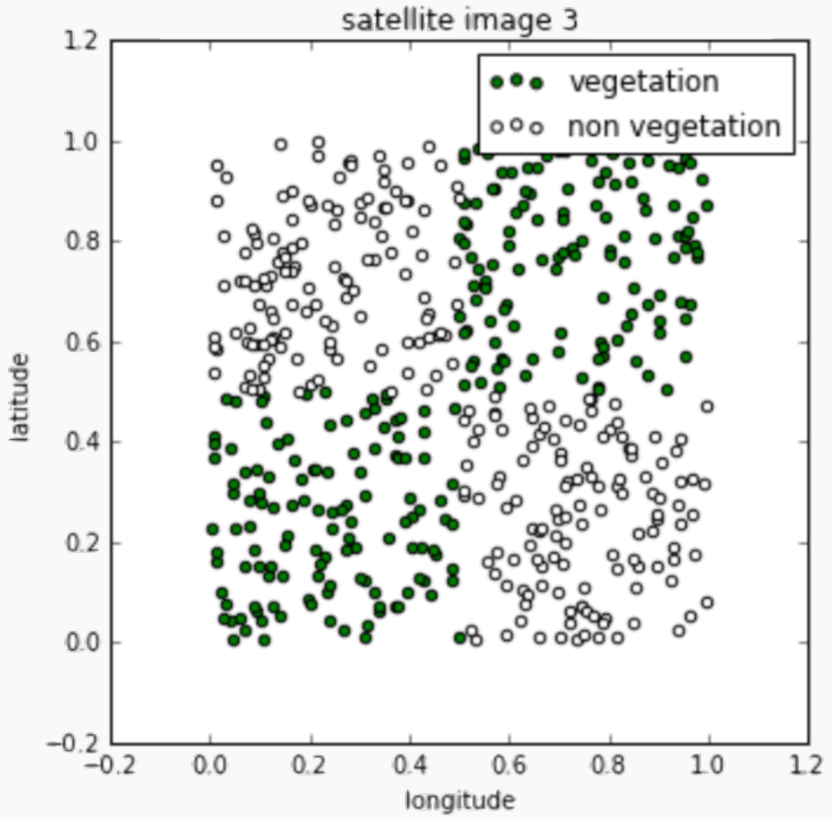
Question: Or these?





Geometry of Data

Notice that in all the datasets the classes are still well-separated in the feature space, but ***the decision boundaries cannot easily be described by a single equation***:



Geometry of Data

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 context:

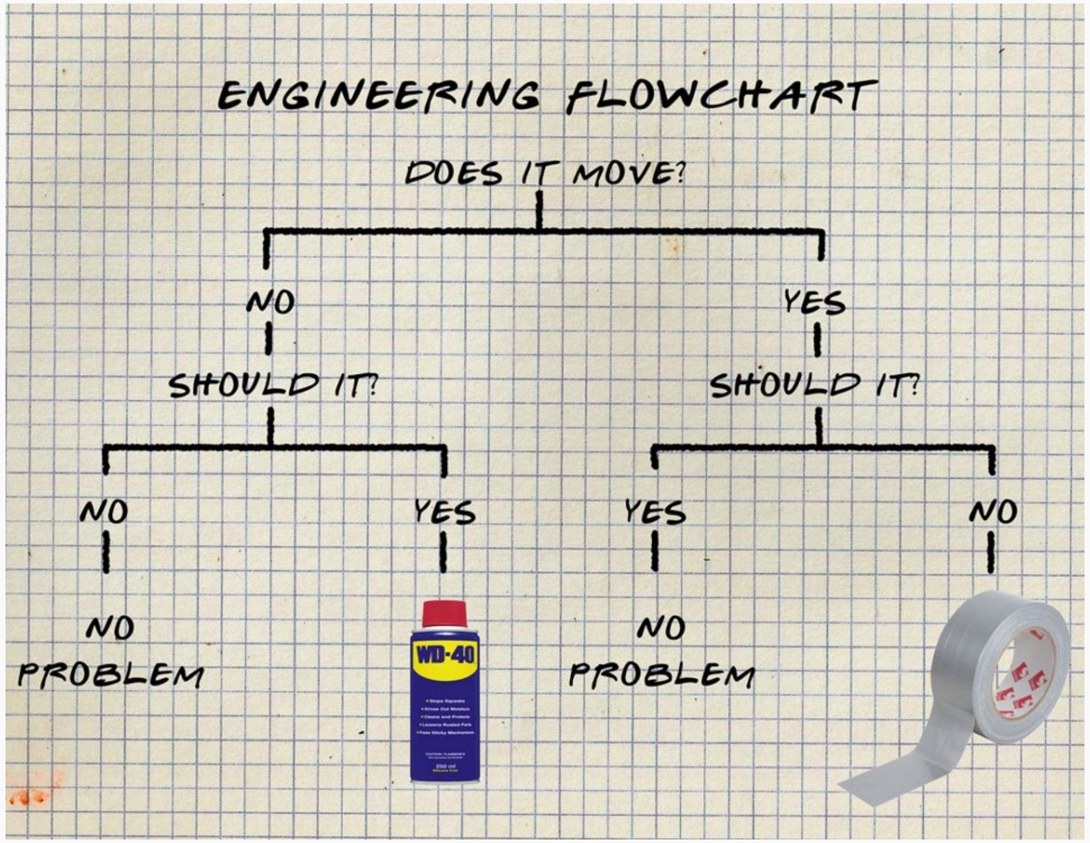
$$(x_3+2x_2) - x_1^2 + 10 = 0$$

It would be desirable to build models that:

1. allow for ***complex decision boundaries***.
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:



Decision Trees

It turns out that the simple flow charts in our examples can be formulated as mathematical models for classification and these models have the properties we desire; they are:

1. interpretable by humans
2. have sufficiently complex decision boundaries
3. the decision boundaries are locally linear, each component of the decision boundary is simple to describe mathematically.

Decision Trees

The Geometry of Flow Charts

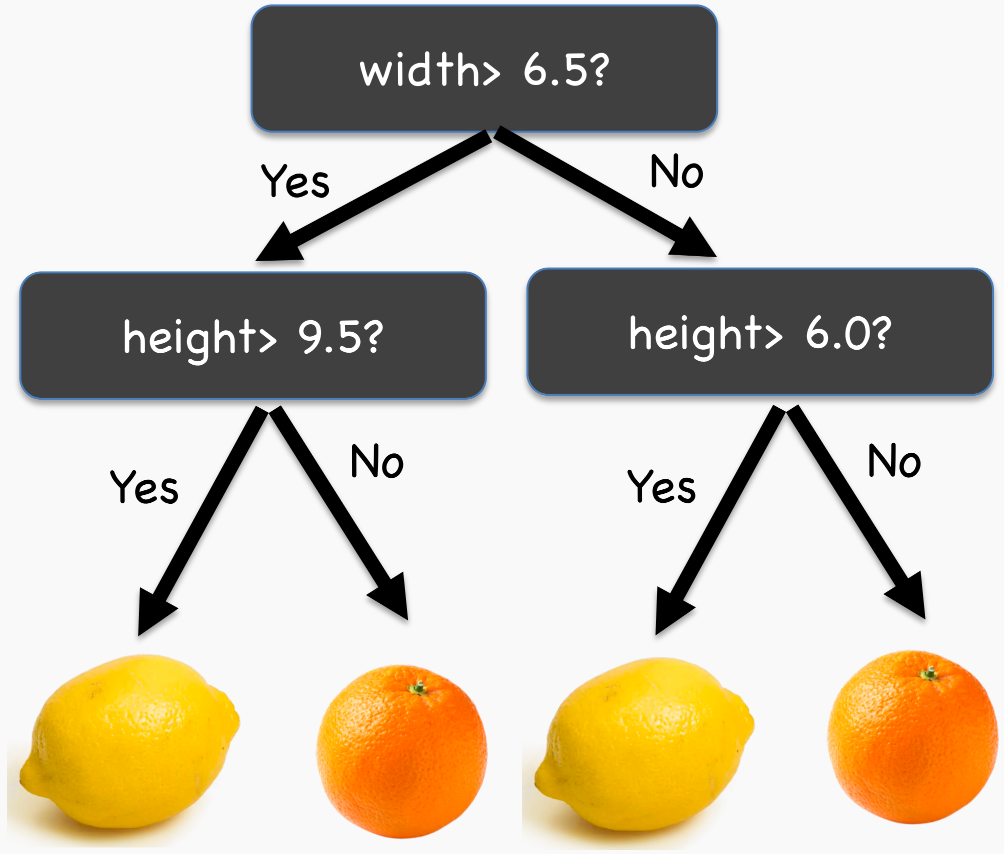
Flow charts whose graph is a tree (connected and there are no cycles) represents a model called a ***decision tree***.

Formally, a ***decision tree model*** is one in which the final outcome of the model is based on a series of comparisons of the values of predictors against threshold values.

In a graphical representation (flow chart),

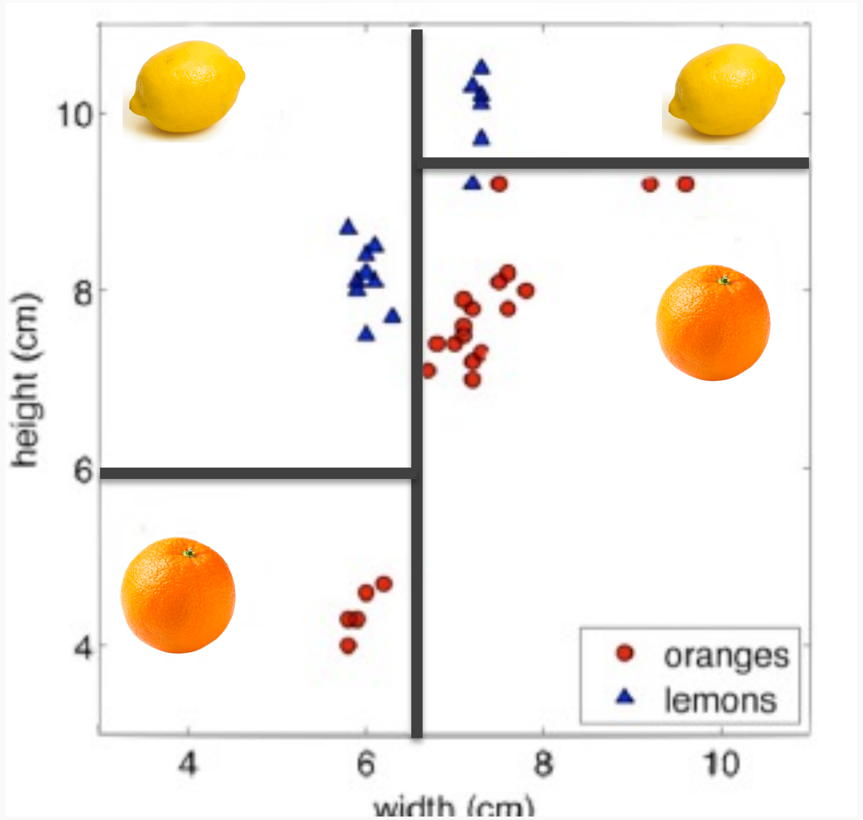
- the internal nodes of the tree represent attribute testing.
- branching in the next level is determined by attribute value (yes/no).
- terminal leaf nodes represent class assignments.

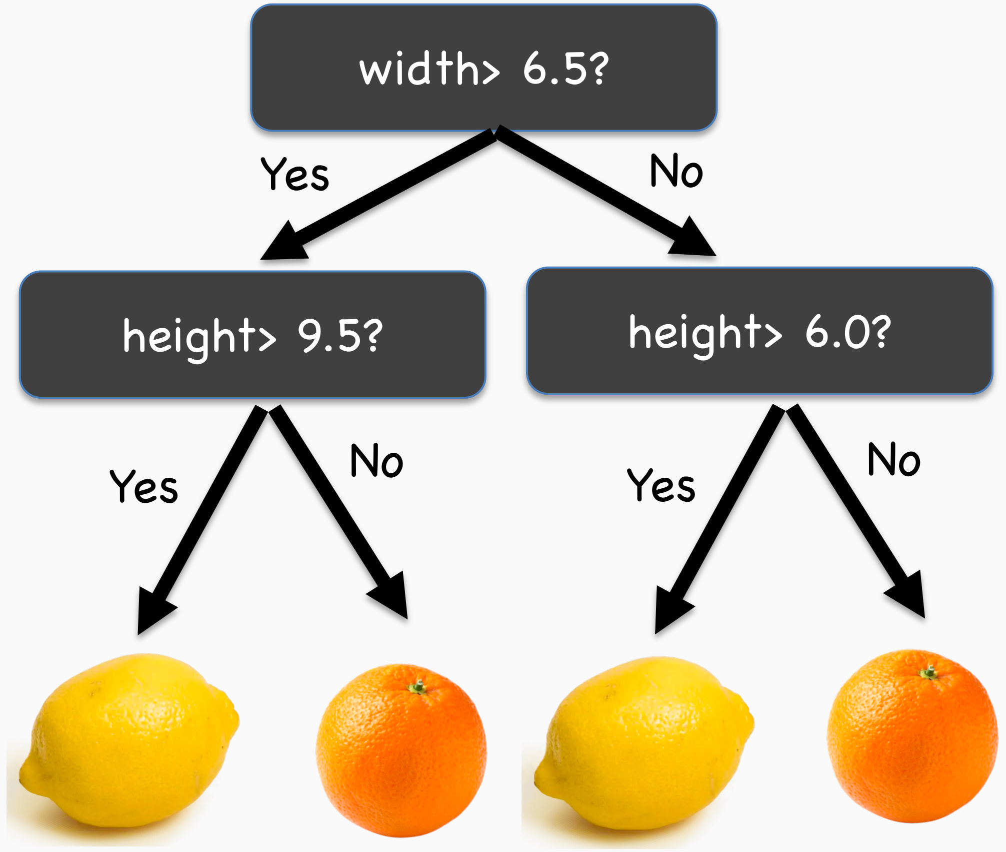
Example: Lemons and Oranges



The Geometry of Flow Charts

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





Learning the Model

Given a training set, *learning* a decision tree model for binary classification means:

- producing an ***optimal*** partition of the feature space with axis-aligned linear boundaries (very interpretable!),
- each region is predicted to have a class label based on the **largest class** of the training points in that region (Bayes' classifier) when performing prediction.

Learning the Model

Learning the smallest ‘optimal’ decision tree for any given set of data is NP complete (intractable) for numerous simple definitions of ‘optimal’. Instead, we will seek a reasonably model using 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 ***stopping condition*** is met

<u>Now, we need only define our splitting criterion and stopping condition.</u>

Splitting Criteria

Optimality of Splitting

While there is no ‘correct’ way to define an optimal split, there are some common sensical 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.
- the fitness metric of a split should take a differentiable form (making optimization possible).
- we shouldn’t end up with empty regions - regions containing no training points.

Classification Error

Suppose we have J number of predictors and K classes.

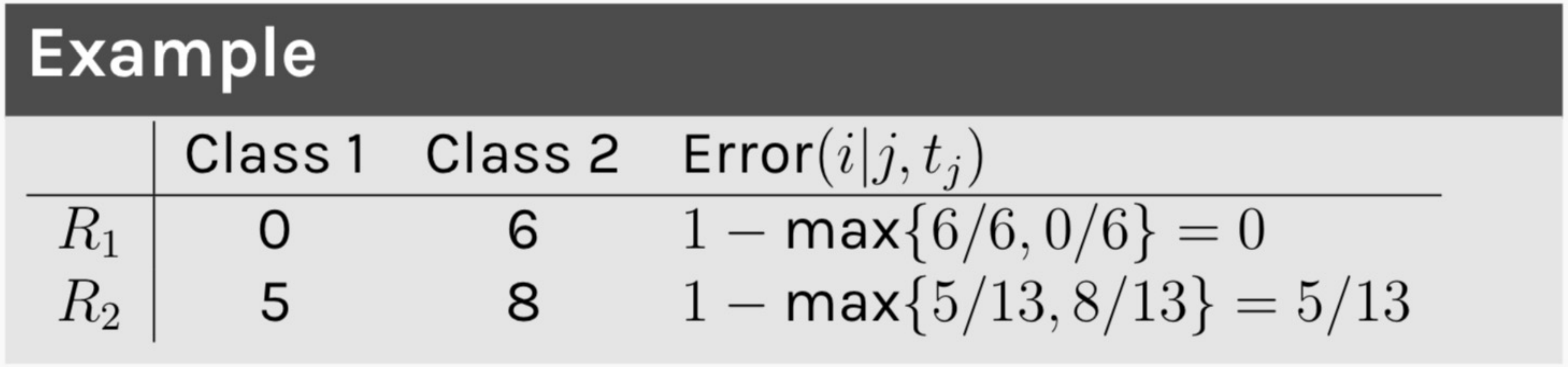
Suppose we select the j^{th} predictor and split a region containing N number of training points along the threshold $t_j \in \mathbb{R}$.

We can assess the quality of this split by measuring the **classification error** made by each newly created region, R_1, R_2:

$$\text{Error}(i|j, t_j) = 1 - \max_k p(k|R_i)$$

where $p(k|R_i)$ is the proportion of training points in R_i that are labeled class k.

Classification Error



Example

	Class 1	Class 2	Error$(i\|j, t_j)$
R_1	0	6	$1 - \max\{6/6, 0/6\} = 0$
R_2	5	8	$1 - \max\{5/13, 8/13\} = 5/13$

We can now try to find the predictor j and the threshold t_j that minimizes the average classification error over the two regions, weighted by the population of the regions:

$$\min_{j,t_j}\left\{\frac{N_1}{N}\text{Error}(1|j,t_j)+\frac{N_2}{N}\text{Error}(2|j,t_j)\right\}$$

where N_i is the number of training points inside region R_i.

Gini Index

Suppose we have J number of predictors, N number of training points and K classes.

Suppose we select the j^{th} predictor and split a region containing N number of training points along the threshold $t_j \in \mathbb{R}$.

We can assess the quality of this split by measuring the purity of each newly created region, R_1, R_2. This metric is called the **Gini Index**:

$$\text{Gini}(i|j, t_j) = 1 - \sum_k p(k|R_i)^2$$

Question: What is the effect of squaring the proportions of each class? What is the effect of summing the squared proportions of classes within each region?

Gini Index

Example

	Class 1	Class 2	$\text{Gini}(i\|j, t_j)$
R_1	0	6	$1 - (6/6^2 + 0/6^2) = 0$
R_2	5	8	$1 - [(5/13)^2 + (8/13)^2] = 80/169$

We can now try to find the predictor j and the threshold t_j that minimizes the average Gini Index over the two regions, weighted by the population of the regions:

$$\min_{j,t_j}\left\{\frac{N_1}{N}\text{Gini}(1|j,t_j) + \frac{N_2}{N}\text{Gini}(2|j,t_j)\right\}$$

where N_i is the number of training points inside region R_i.

Information Theory

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

Ideally, our decision tree should split the feature space into regions such that each region represents a single class. In practice, the training points in each region is distributed over multiple classes, e.g.:

	Class 1	Class 2
R_1	1	6
R_2	5	6

However, though both imperfect, R_1 is clearly sending a stronger 'signal' for a single class (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:

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

Higher entropy means the distribution is uniform-like (flat histogram) and thus values sampled from it are 'less predictable' (all possible values are equally probable).

Lower entropy means the distribution has more defined peaks and valleys and thus values sampled from it are 'more predictable' (values around the peaks are more probable).

Entropy

Suppose we have J number of predictors, N number of training points and K classes.

Suppose we select the j^{th} predictor and split a region containing N number of training points along the threshold $t_j \in \mathbb{R}$.

We can assess the quality of this split by measuring the entropy of the class distribution in each newly created region, R_1, R_2:

$$\text{Entropy}(i|j,k) = -\sum_k p(k|R_i)\log_2 p(k|R_i)$$

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 i.

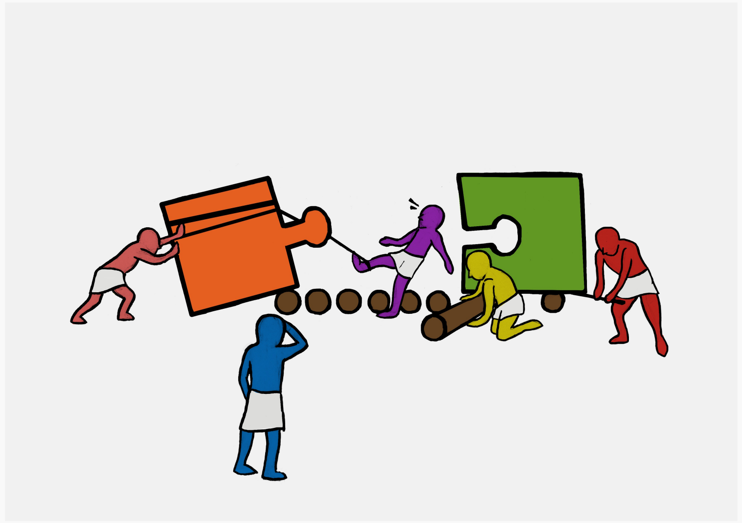
Entropy

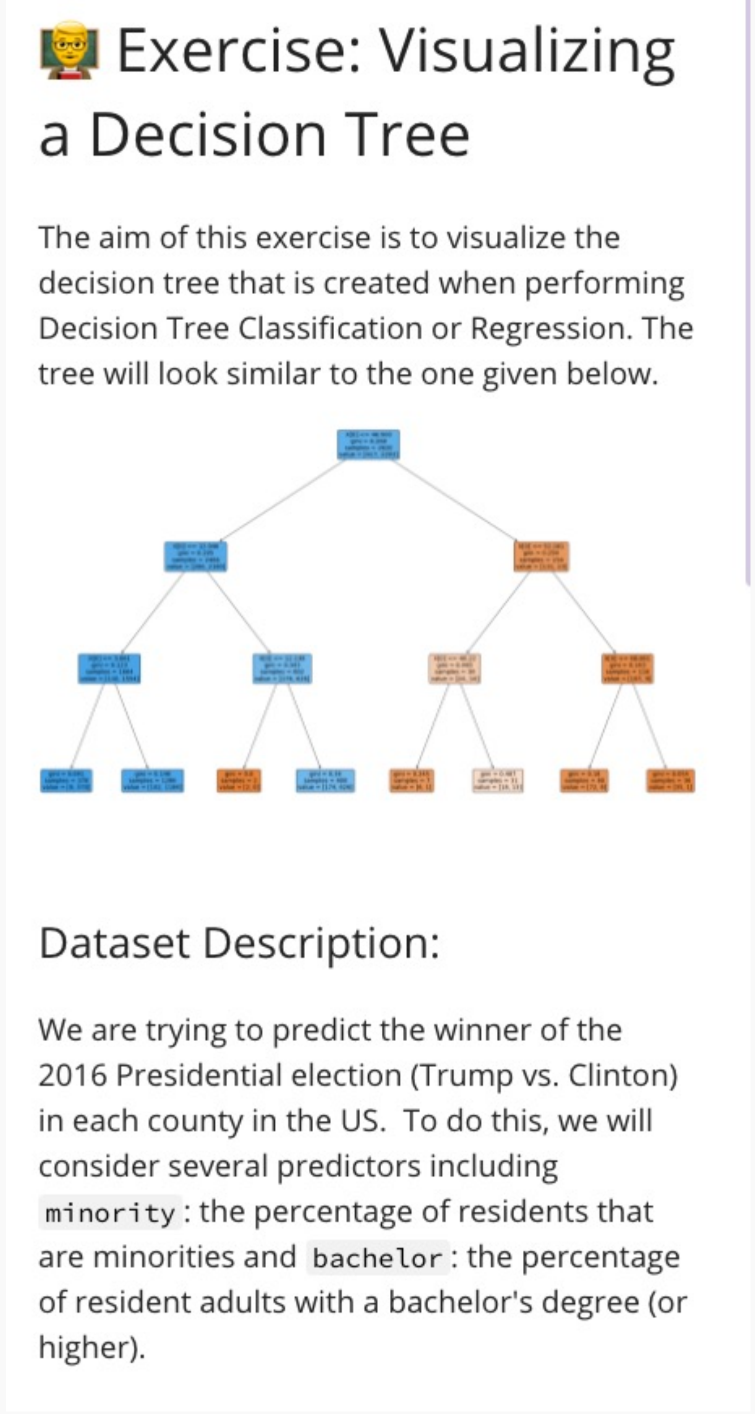
Example

	Class 1	Class 2	$\text{Entropy}(i\|j, t_j)$
R_1	0	6	$-(\frac{6}{6}\log_2\frac{6}{6} + \frac{0}{6}\log_2\frac{0}{6}) = 0$
R_2	5	8	$-(\frac{5}{13}\log_2\frac{5}{13} + \frac{8}{13}\log_2\frac{8}{13}) \approx 1.38$

We can now try to find the predictor *j* and the threshold t_j that minimizes the average entropy over the two regions, weighted by the population of the regions:

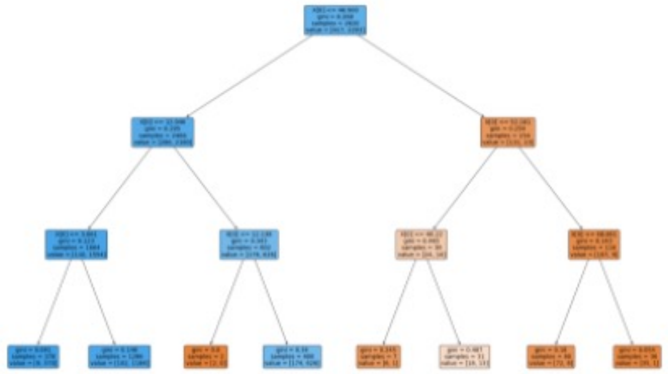
$$\min_{j,t_j}\left\{\frac{N_1}{N}\text{Entropy}(1|j,t_j) + \frac{N_2}{N}\text{Entropy}(2|j,t_j)\right\}$$





👨‍🏫 Exercise: Visualizing a Decision Tree

The aim of this exercise is to visualize the decision tree that is created when performing Decision Tree Classification or Regression. The tree will look similar to the one given below.



Dataset Description:

We are trying to predict the winner of the 2016 Presidential election (Trump vs. Clinton) in each county in the US. To do this, we will consider several predictors including `minority`: the percentage of residents that are minorities and `bachelor`: the percentage of resident adults with a bachelor's degree (or higher).