GENERAL ASSEMBLY

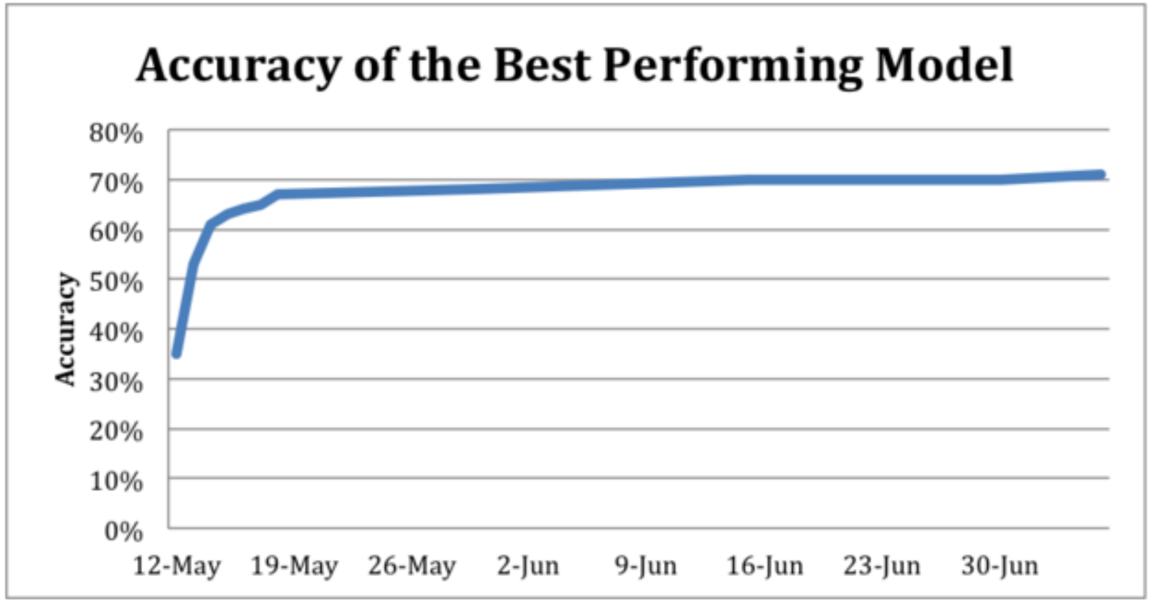
DECISION TREES

Brian Chung

QUESTIONS FROM LAST TIME - SUPPORT VECTOR MACHINES

- **RBF!?!? What the hell is that?**
- **Tradeoff between slack and misclassified samples?**
- **Applying SVM to classify N classes**
- **More data preparation techniques**

WHAT CAN YOU DO WITH AN 80% ALGORITHM?



A few months ago, my company, CrowdFlower, ran a machine learning competition on Kaggle. It perfectly highlighted the biggest opportunity (and challenge) with machine learning: What do you do with an 80% accurate algorithm?

We uploaded data collected on our platform and Kaggle sent it out to over 1,000 data scientists, who competed to see who could build the best search model.

The simplest approach gave a baseline accuracy of 32%. Within hours a team beat that with a 35% accurate model. By the next morning, one team already had a 53% accurate model.

With Microsoft's Azure ML and IBM's investment in Watson, making models is easier than ever. Companies no longer need a Google-size R&D budget to make machine learning applicable to their business. These models aren't perfect, but they're useful. The new challenge for businesses is how to integrate an imperfect machine-learning algorithm into their existing workflow.

http://www.computerworld.com/article/3023708/data-analytics/the-machine-learning-problem-of-the-next-decade.html

DECISION TREE AGENDA

I. DECISION TREES
II. FITTING DECISION TREES
III. OBJECTIVE FUNCTIONS
IV. REGULARIZATION
V. PROS/CONS & USAGE
VI. DECISION TREE LAB
VII. RANDOM FOREST (TIME PERMITTING)

DECISION TREE LEARNING OBJECTIVES

1. Understand how decision trees function for classification problems
2. Learn the objective functions used in building decision trees
3. Learn the regularization process in decision tree building

I. DECISION TREES

TYPES OF ML SOLUTIONS

	Continuous	Categorical
Supervised	Regression	Classification
Unsupervised	Dimension Reduction	Clustering

WHAT ARE DECISION TREES

Decision trees are automatically generated rules based models that can be used in both classification and regression. We will focus on classification trees due to their superior performance.

WHAT ARE DECISION TREES

Decision trees are automatically generated rules based models that can be used in both classification and regression. We will focus on classification trees due to their superior performance.

As with any other model, we begin with a set of features **X**, and a set of categorical response variables **y**

WHAT ARE DECISION TREES

Decision trees are automatically generated rules based models that can be used in both classification and regression. We will focus on classification trees due to their superior performance.

As with any other model, we begin with a set of features **X**, and a set of categorical response variables **y**

The features can be both continuous and categorical and do not require any scaling or preprocessing

WHAT ARE DECISION TREES

At its heart, a decision tree classifier is a non-parametric hierarchical classification technique

WHAT ARE DECISION TREES

At its heart, a decision tree classifier is a non-parametric hierarchical classification technique

non-parametric: no parameters (beta), no distribution assumptions

WHAT ARE DECISION TREES

At its heart, a decision tree classifier is a non-parametric hierarchical classification technique

non-parametric: no parameters (beta), no distribution assumptions

hierarchical: consists of a sequence of questions which yield a class label when applied to any record

HOW ARE DECISION TREES REPRESENTED?

Q: How is a decision tree represented?

HOW ARE DECISION TREES REPRESENTED?

Q: How is a decision tree represented?

A: Using a configuration of **nodes** *and* **edges**.

HOW ARE DECISION TREES REPRESENTED?

Q: How is a decision tree represented?

A: Using a configuration of **nodes** *and* **edges***.*

More concretely, as a multiway tree, which is a type of (directed acyclic) **graph***.*

HOW ARE DECISION TREES REPRESENTED?

Q: How is a decision tree represented?
A: Using a configuration of **nodes** *and* **edges**.

More concretely, as a multiway tree, which is a type of (directed acyclic) **graph**.

*In a decision tree, the nodes represent questions (***test conditions)** *and the edges are the answers to these questions.*

HOW ARE DECISION TREES REPRESENTED?

The top node of the tree is called the **root node**. *This node has 0 incoming edges, and 2+ outgoing edges.*

HOW ARE DECISION TREES REPRESENTED?

The top node of the tree is called the **root node***. This node has 0 incoming edges, and 2+ outgoing edges.*

An **internal node** *has 1 incoming edge, and 2+ outgoing edges. Internal nodes represent test conditions.*

HOW ARE DECISION TREES REPRESENTED?

The top node of the tree is called the **root node**. *This node has 0 incoming edges, and 2+ outgoing edges.*

An **internal node** *has 1 incoming edge, and 2+ outgoing edges. Internal nodes represent test conditions.*

A **leaf node** *has 1 incoming edge and, 0 outgoing edges. Leaf nodes correspond to class labels.*

HOW ARE DECISION TREES REPRESENTED?

The top node of the tree is called the **root node**. *This node has 0 incoming edges, and 2+ outgoing edges.*

An **internal node** *has 1 incoming edge, and 2+ outgoing edges. Internal nodes represent test conditions.*

A **leaf node** *has 1 incoming edge and, 0 outgoing edges. Lea*

correspond to class labels.

NOTE

The nodes in our tree are connected by *directed edges*.

These directed edges lead from *parent nodes* to *child nodes.*

EXAMPLE - DATA

Table 4.1. The vertebrate data set.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	reptile
salmon	cold-blooded	scales	no	yes	no	no	no	fish
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	amphibian
komodo dragon	cold-blooded	scales	no	no	no	yes	no	reptile
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	bird
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	fish
turtle	cold-blooded	scales	no	semi	no	yes	no	reptile
penguin	warm-blooded	feathers	no	semi	no	yes	no	bird
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	fish
salamander	cold-blooded	none	no	semi	no	yes	yes	amphibian

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

EXAMPLE – DATA

Table 4.1. The vertebrate data set.

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hiber-nates	Class Label
human	warm-blooded	hair	yes	no	no	yes	no	mammal
python	cold-blooded	scales	no	no	no	no	yes	non-mammal
salmon	cold-blooded	scales	no	yes	no	no	no	non-mammal
whale	warm-blooded	hair	yes	yes	no	no	no	mammal
frog	cold-blooded	none	no	semi	no	yes	yes	non-mammal
komodo dragon	cold-blooded	scales	no	no	no	yes	no	non-mammal
bat	warm-blooded	hair	yes	no	yes	yes	yes	mammal
pigeon	warm-blooded	feathers	no	no	yes	yes	no	non-mammal
cat	warm-blooded	fur	yes	no	no	yes	no	mammal
leopard shark	cold-blooded	scales	yes	yes	no	no	no	non-mammal
turtle	cold-blooded	scales	no	semi	no	yes	no	non-mammal
penguin	warm-blooded	feathers	no	semi	no	yes	no	non-mammal
porcupine	warm-blooded	quills	yes	no	no	yes	yes	mammal
eel	cold-blooded	scales	no	yes	no	no	no	non-mammal
salamander	cold-blooded	none	no	semi	no	yes	yes	non-mammal

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

EXAMPLE - DECISION TREE

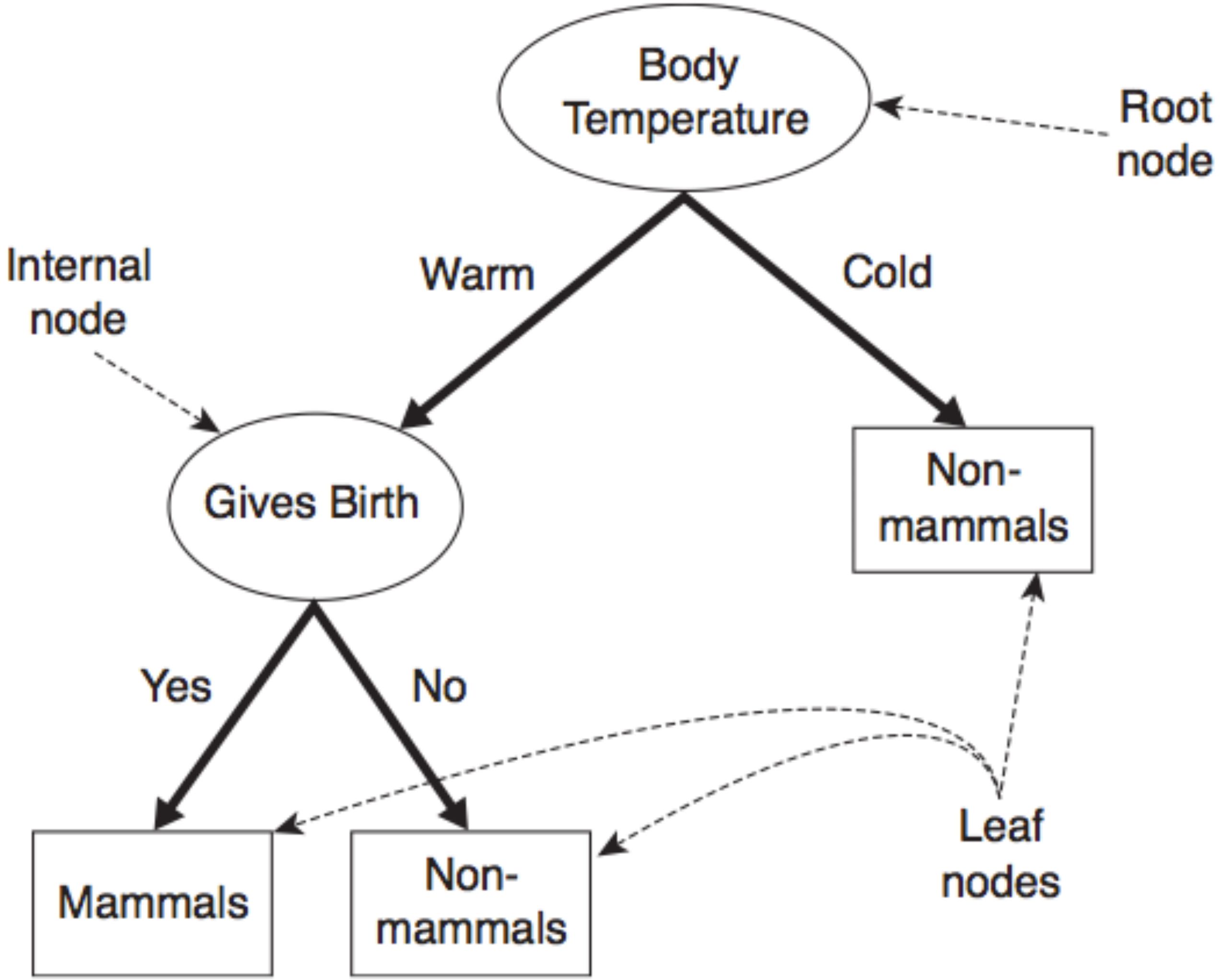
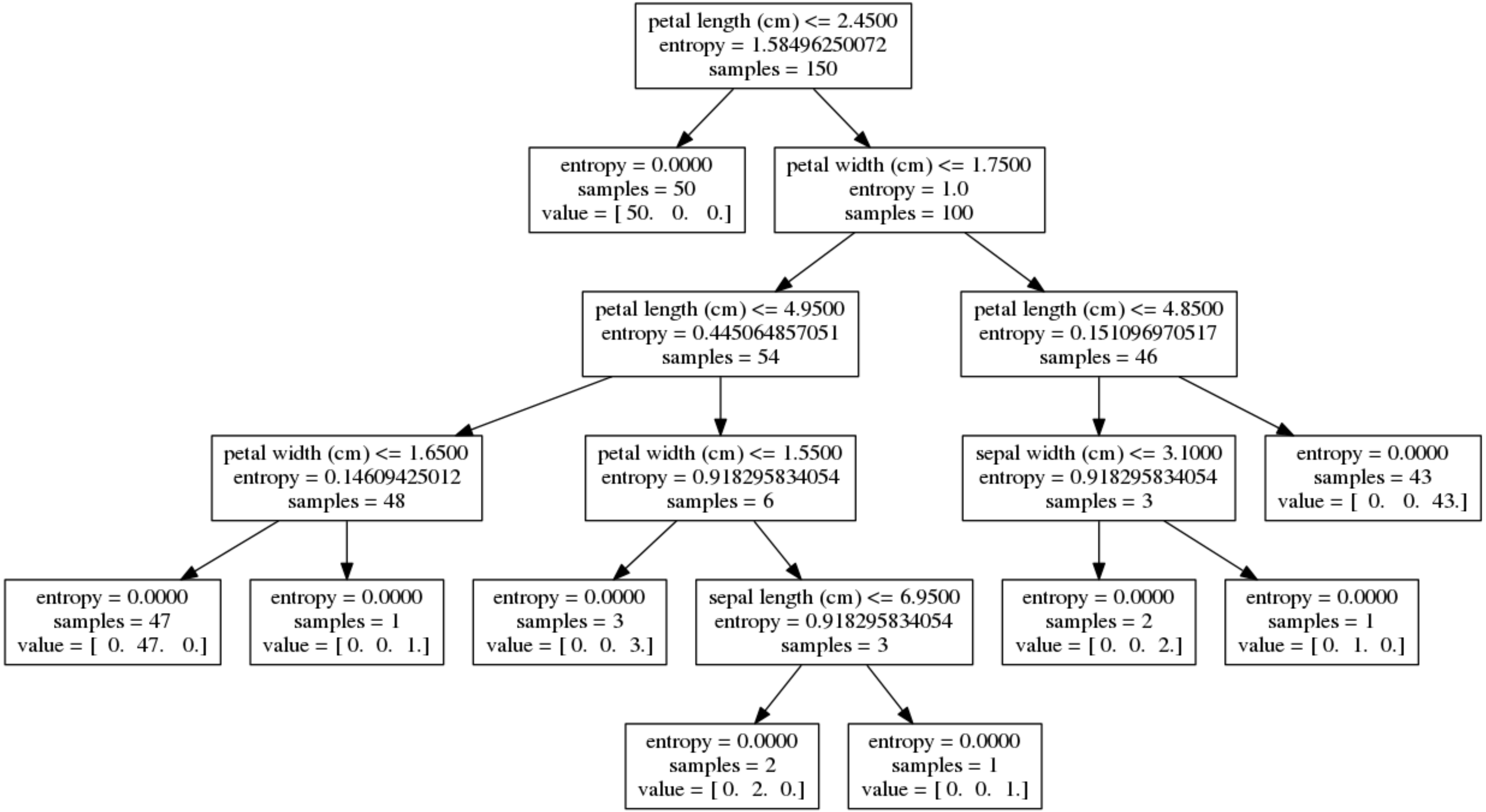


Figure 4.4. A decision tree for the mammal classification problem.

source: http://www-users.cs.umn.edu/~kumar/dmbook/ch4.pdf

EXAMPLE - DECISION TREE WITH IRIS DATASET



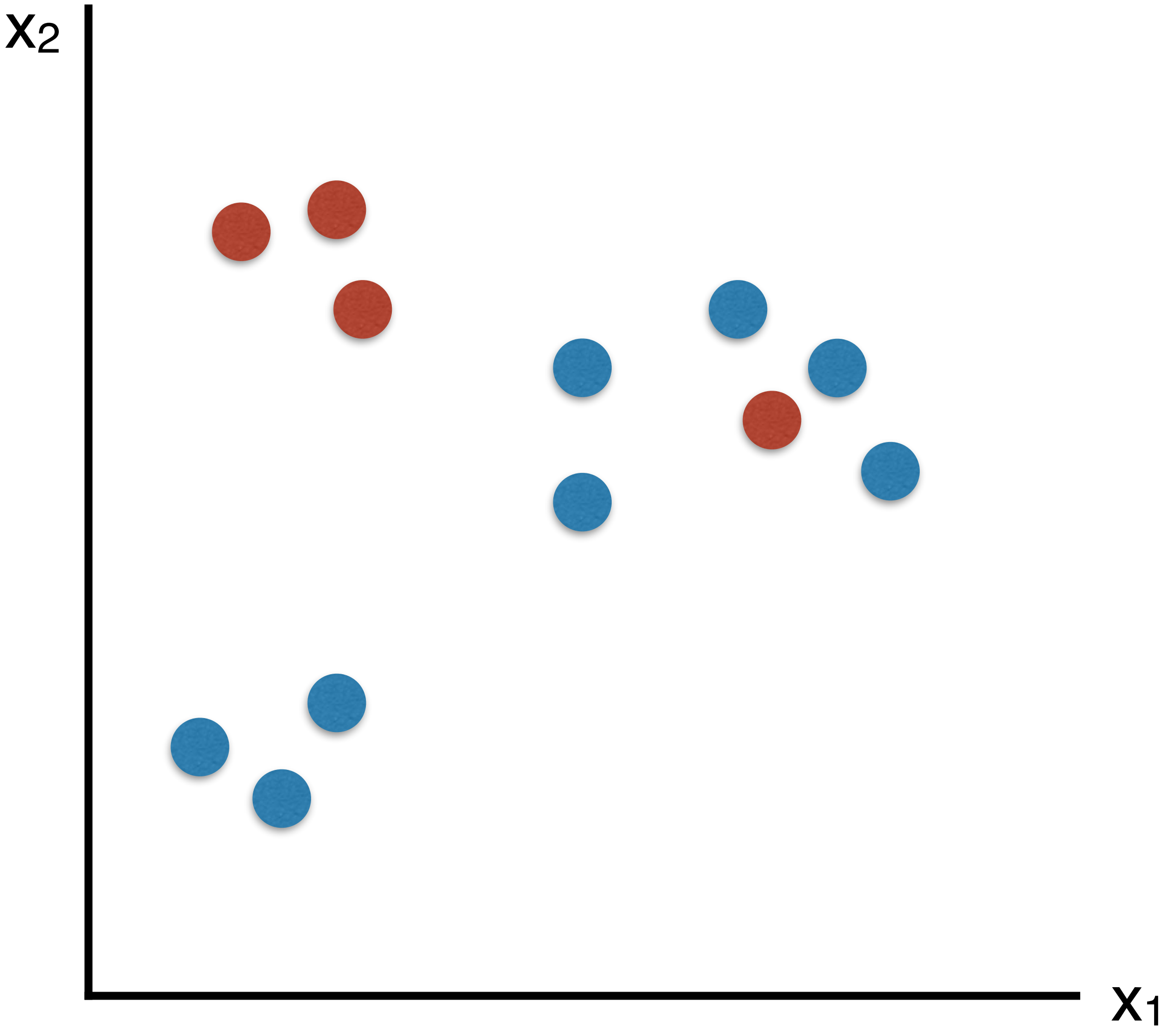
II. BUILDING DECISION TREES

LET'S GO OVER OUR OWN

Let's go over an example of how one might be built with binary classes in R^2 space (i.e. two features).

- Our goal will be to split the space into **regions**.
- Inside each region, we'll classify any point in that space as the **max voted class** within that region
- We will only consider constant splits (no linear splits based on more than one feature)
- We can finish when either every leaf is pure, we have reached a max depth, or when there are less than x amount of data points at a node

LET'S GO OVER OUR OWN



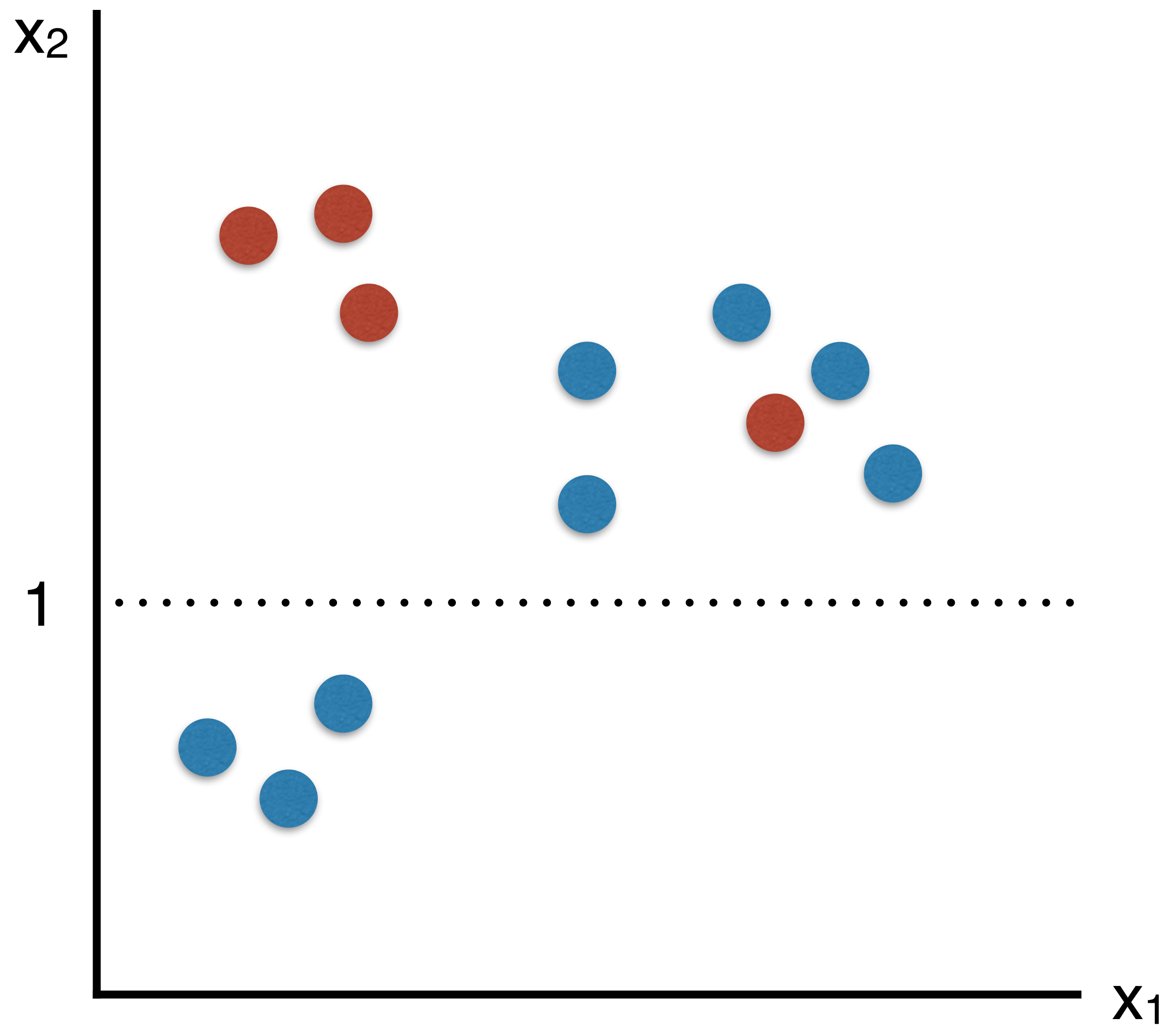
Within a region:

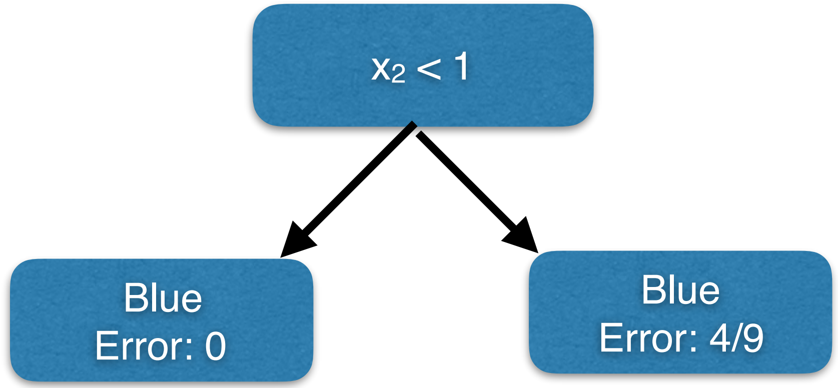
Prediction: Most common class label

Error: Fraction of misclassified labels

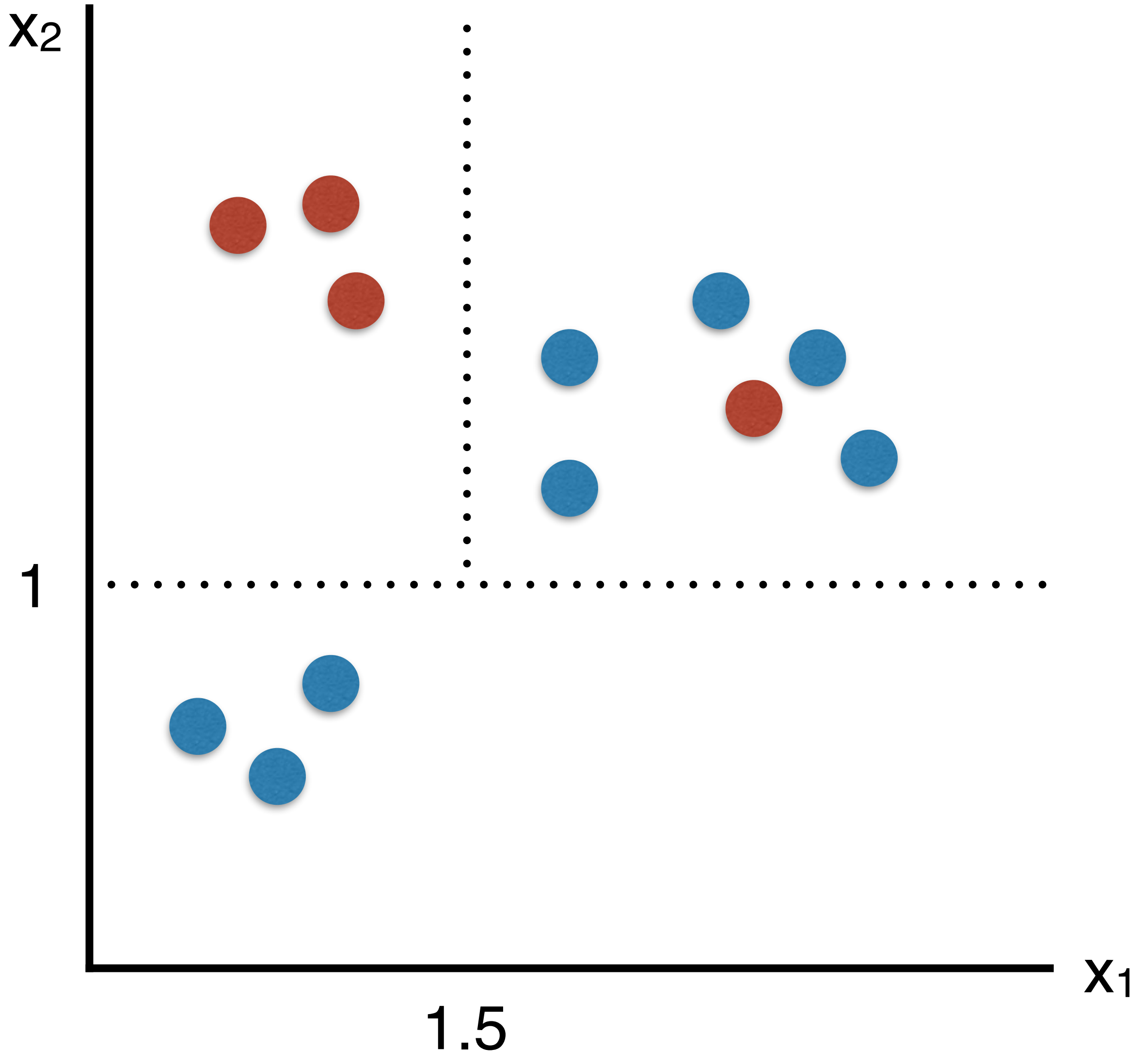
In the entire space:
N = 12 points
Prediction = Blue
Error = 4 / 12 = 33% misclassification error

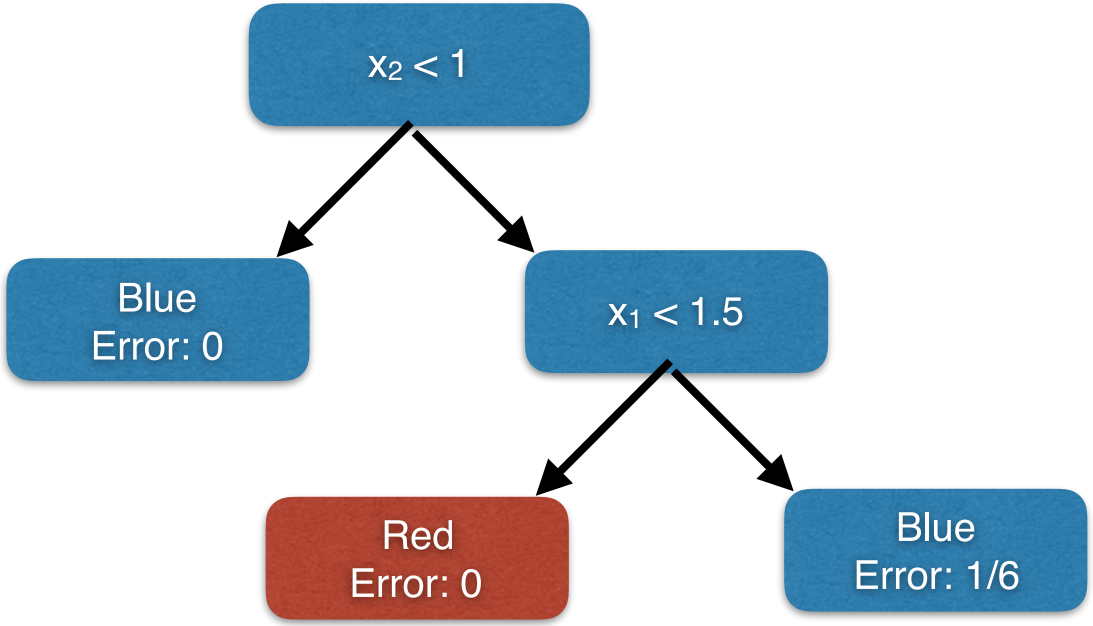
LET'S GO OVER OUR OWN



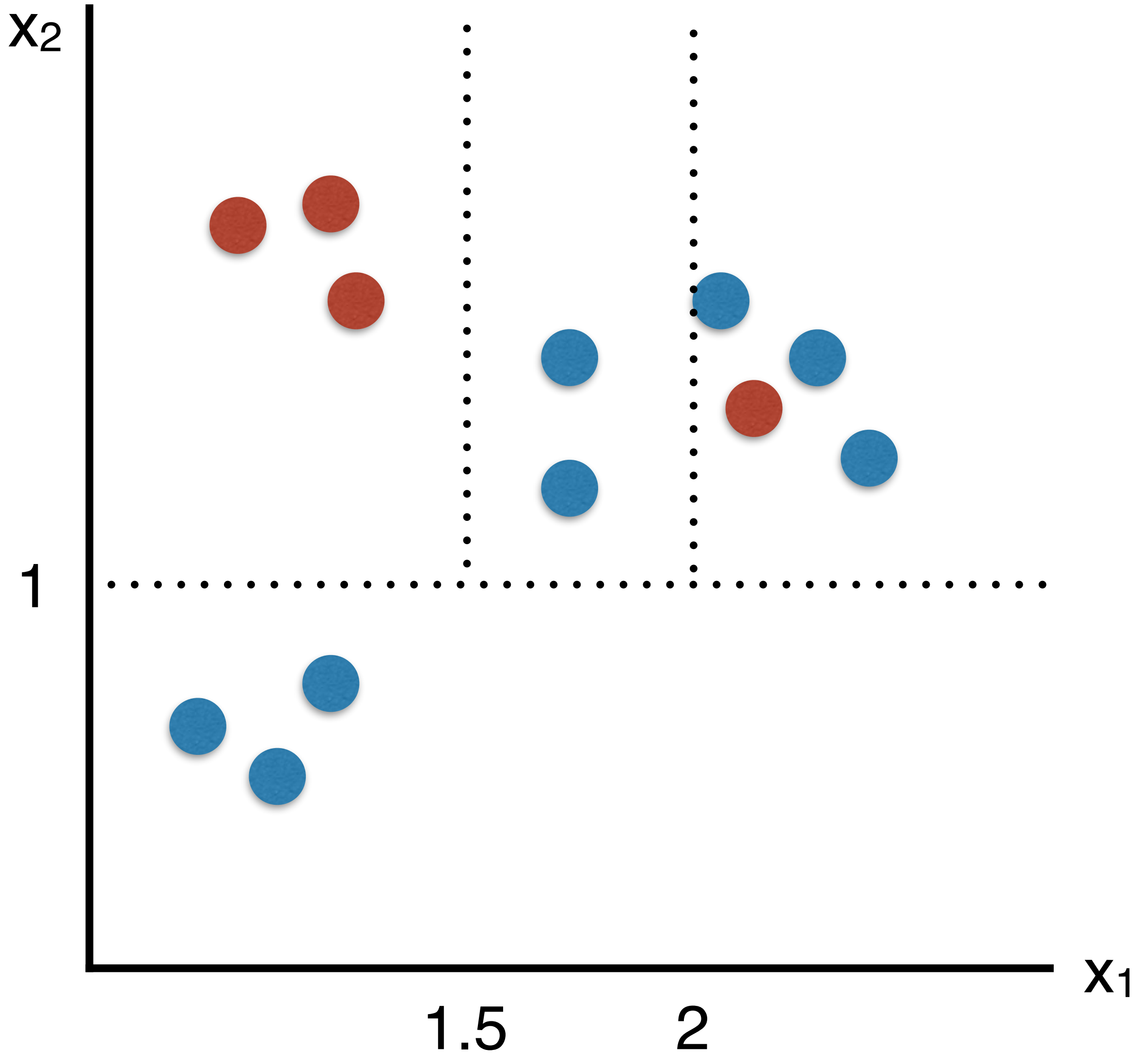


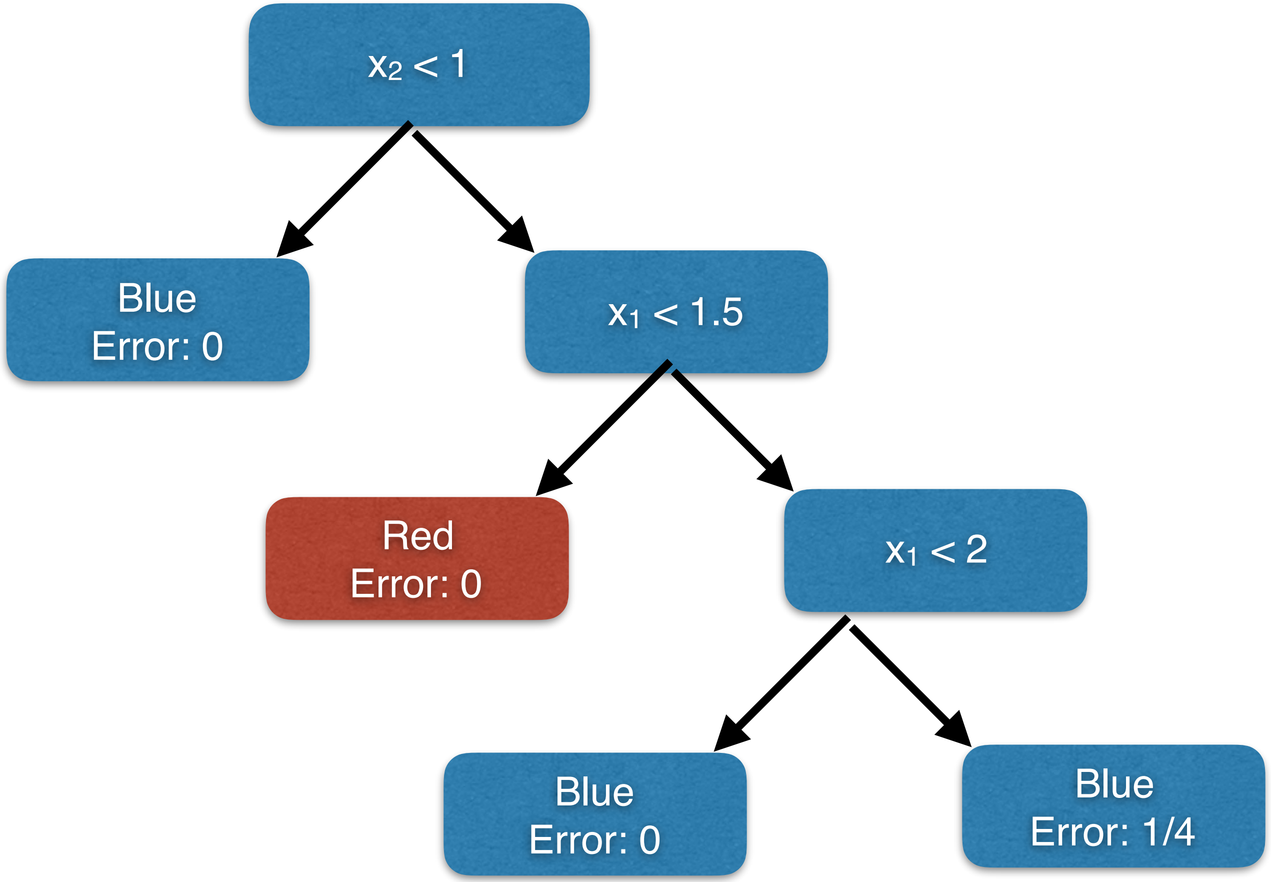
LET'S GO OVER OUR OWN



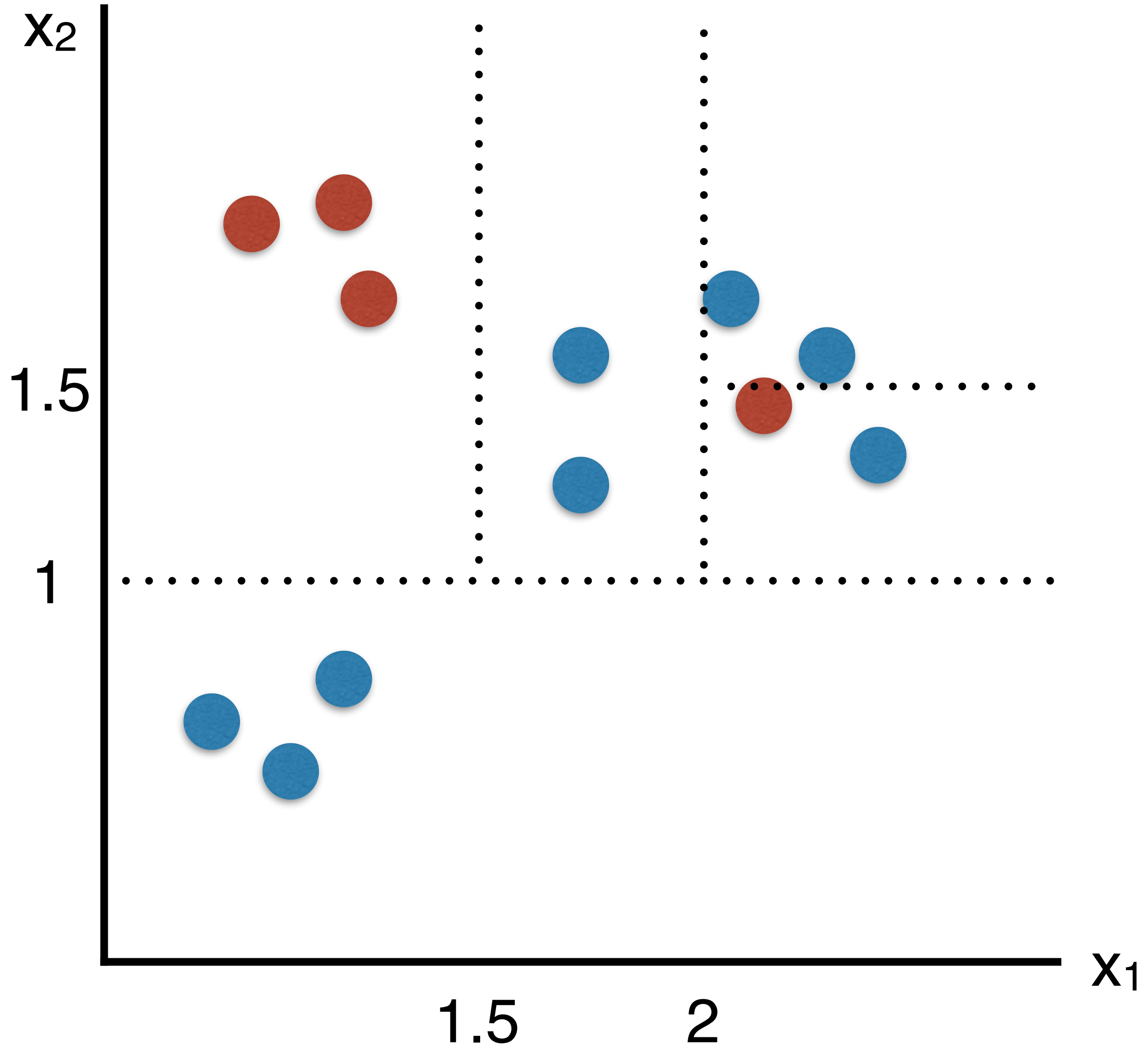


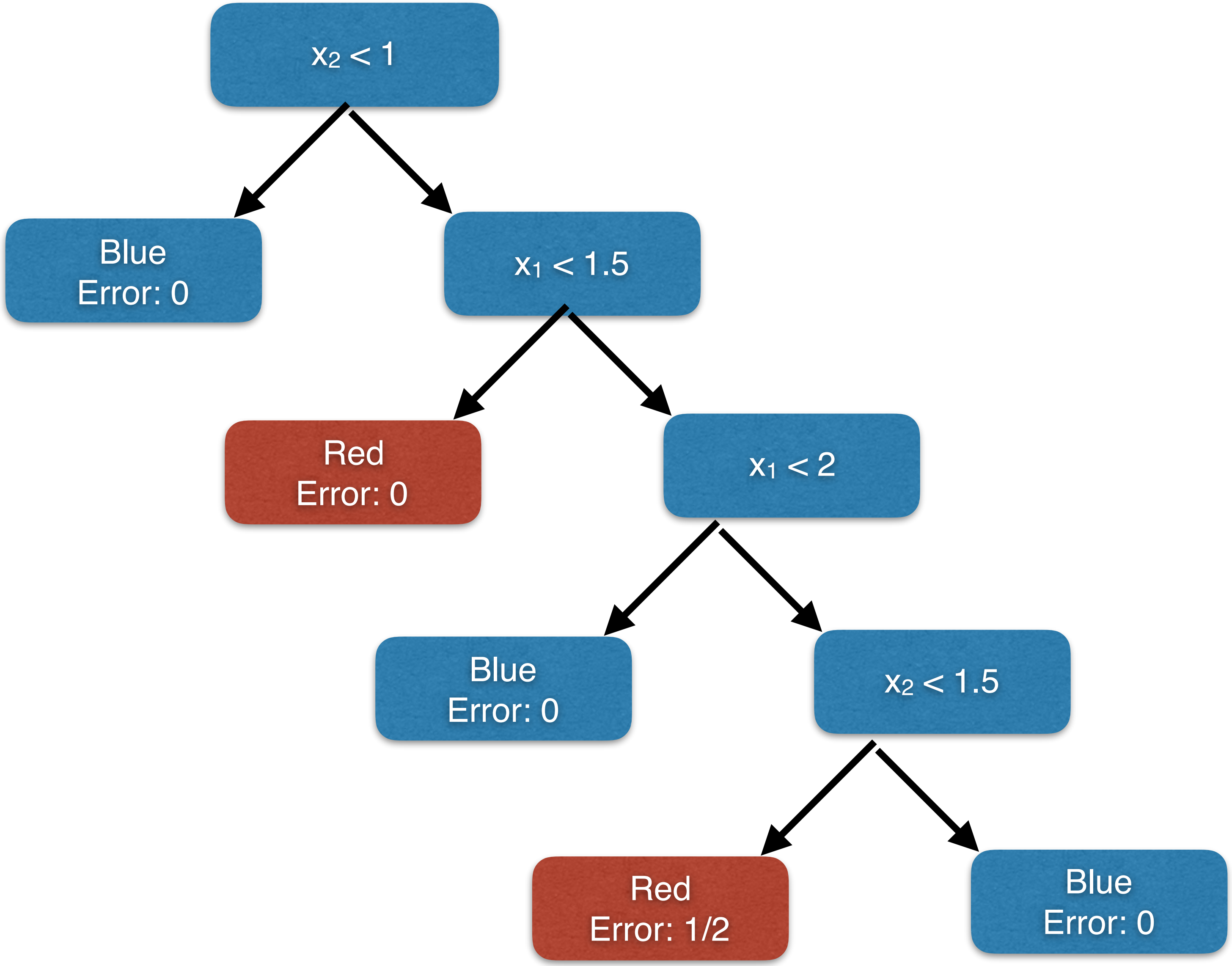
LET'S GO OVER OUR OWN





LET'S GO OVER OUR OWN





BUILDING A DECISION TREE

Okay, so we went over an example process of what the subdivision of the regions looked like. Let's formalize this more.

BUILDING A DECISION TREE

Q: How do we build a decision tree?

BUILDING A DECISION TREE

Q: How do we build a decision tree?

A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

BUILDING A DECISION TREE

Q: How do we build a decision tree?

A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

But this is generally too complex to be practical → $O(2^n)$.

BUILDING A DECISION TREE

Q: How do we build a decision tree?
A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

But this is generally too complex to be practical → $O(2^n)$.

Q: How do we find a practical solution that works?

BUILDING A DECISION TREE

Q: How do we build a decision tree?

A: One possibility would be to evaluate all possible decision trees (eg, all permutations of test conditions) for a given dataset.

But this is generally too complex to be practical → $O(2^n)$.

Q: How do we find a practical solution that works?

A: Use a **heuristic** *algorithm.*

BUILDING A DECISION TREE

The basic method used to build (or "grow") a decision tree is **Hunt's algorithm***.*

BUILDING A DECISION TREE

The basic method used to build (or "grow") a decision tree is **Hunt's algorithm**.

This is a **greedy recursive** *algorithm that leads to a* **local optimum**.

BUILDING A DECISION TREE

The basic method used to build (or "grow") a decision tree is **Hunt's algorithm***.*

This is a **greedy recursive** *algorithm that leads to a* **local optimum***.*

greedy – *algorithm makes locally optimal decision at each step*
recursive – *splits task into subtasks, solves each the same way*
local optimum – *solution for a given neighborhood of points*

BUILDING A DECISION TREE

Hunt's algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

BUILDING A DECISION TREE

Hunt's algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

The partitioning decision is made at each node according to a metric called **purity**.

Hunt's algorithm builds a decision tree by recursively partitioning records into smaller & smaller subsets.

The partitioning decision is made at each node according to a metric called **purity**.

A partition is 100% pure when all of its records belong to a single class.

BUILDING A DECISION TREE

Consider a binary classification problem with classes Y_1, Y_2. Given a set of records D_t at node t, Hunt's algorithm proceeds as follows:

BUILDING A DECISION TREE

Consider a binary classification problem with classes Y_1, Y_2. Given a set of records D_t at node t, Hunt's algorithm proceeds as follows:

1) If all records in D_t belong to class Y_1, then t is a leaf node corresponding to class Y_1.

BUILDING A DECISION TREE

Consider a binary classification problem with classes Y_1, Y_2. Given a set of records D_t at node t, Hunt's algorithm proceeds as follows:

1) If all records in D_t belong to class Y_1, then t is a leaf node corresponding to class Y_1.

NOTE

This is the *base case* for the recursive algorithm.

BUILDING A DECISION TREE

Consider a binary classification problem with classes Y_1, Y_2. Given a set of records D_t at node t, Hunt's algorithm proceeds as follows:

2) If D_t contains records from both classes, then a test condition is created to partition the records further. In this case, t is an internal node whose outgoing edges correspond to the possible outcomes of this test condition.

BUILDING A DECISION TREE

Consider a binary classification problem with classes Y_1, Y_2. Given a set of records D_t at node t, Hunt's algorithm proceeds as follows:

2) If D_t contains records from both classes, then a test condition is created to partition the records further. In this case, t is an internal node whose outgoing edges correspond to the possible outcomes of this test condition.

These outgoing edges terminate in **child nodes**. *A record d in D_t is assigned to one of these child nodes based on the outcome of the test condition applied to d.*

BUILDING A DECISION TREE

Consider a binary classification problem with classes Y_1, Y_2. Given a set of records D_t at node t, Hunt's algorithm proceeds as follows:

3) These steps are then recursively applied to each child node.

BUILDING A DECISION TREE

Consider a binary classification problem with classes Y_1, Y_2. Given a set of records D_t at node t, Hunt's algorithm proceeds as follows:

3) These steps are then recursively applied to each child node.

NOTE

Decision trees are easy to interpret, but the algorithms to create them are a bit complicated.

Q: How do we partition the training records?

Q: How do we partition the training records?

A: There are a few ways to do this.

BUILDING A DECISION TREE

Q: How do we partition the training records?

A: There are a few ways to do this.

Test conditions can create **binary splits***:*

BUILDING A DECISION TREE

Q: How do we partition the training records?

A: There are a few ways to do this.

Test conditions can create **binary splits***:*

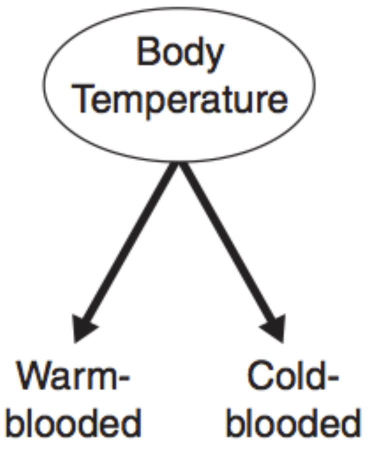
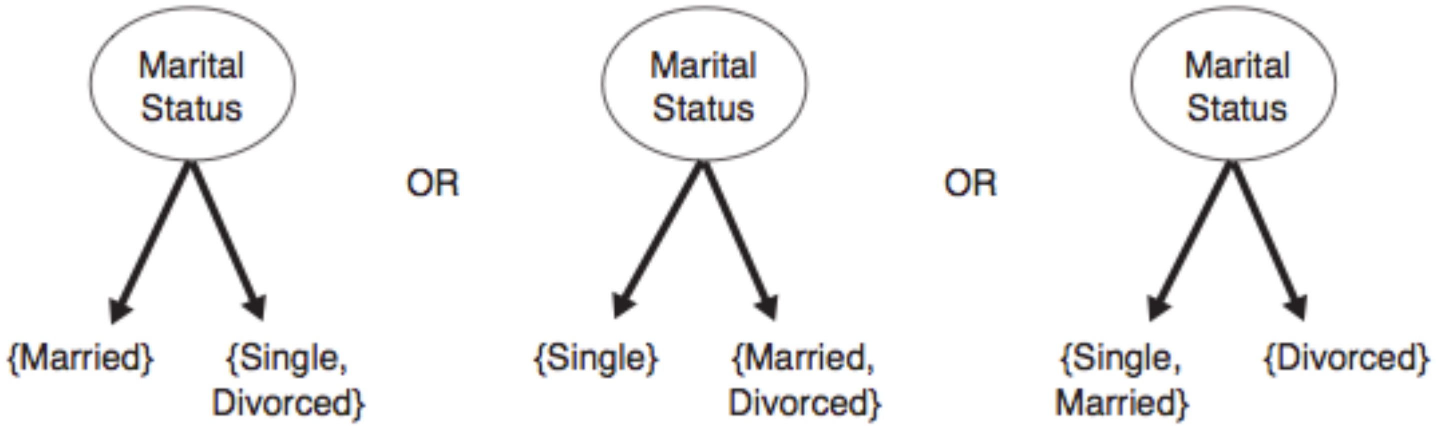


Figure 4.8. Test condition for binary attributes.

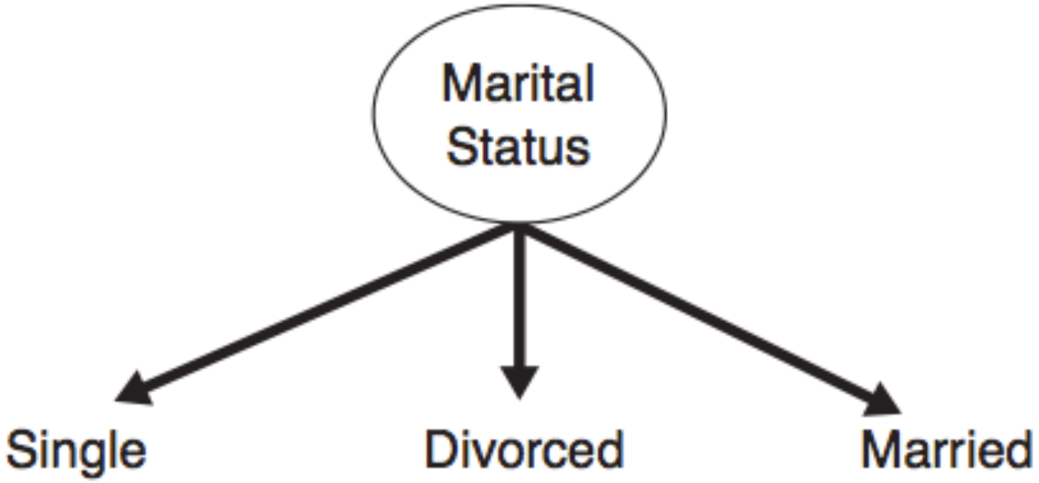


(b) Binary split {by grouping attribute values}

Q: How do we partition the training records?

A: There are a few ways to do this.

Alternatively, we can create **multiway splits***:*



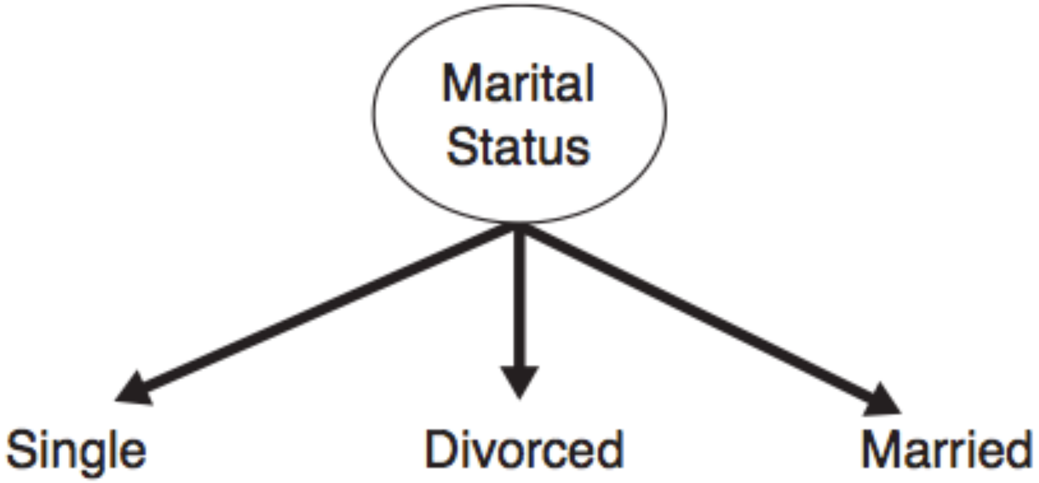
(a) Multiway split

BUILDING A DECISION TREE

Q: How do we partition the training records?

A: There are a few ways to do this.

Alternatively, we can create **multiway splits***:*



(a) Multiway split

NOTE

Multiway splits can produce purer subsets, but may lead to overfitting!

Q: How do we partition the training records?

A: There are a few ways to do this.

For continuous features, we can use either method:

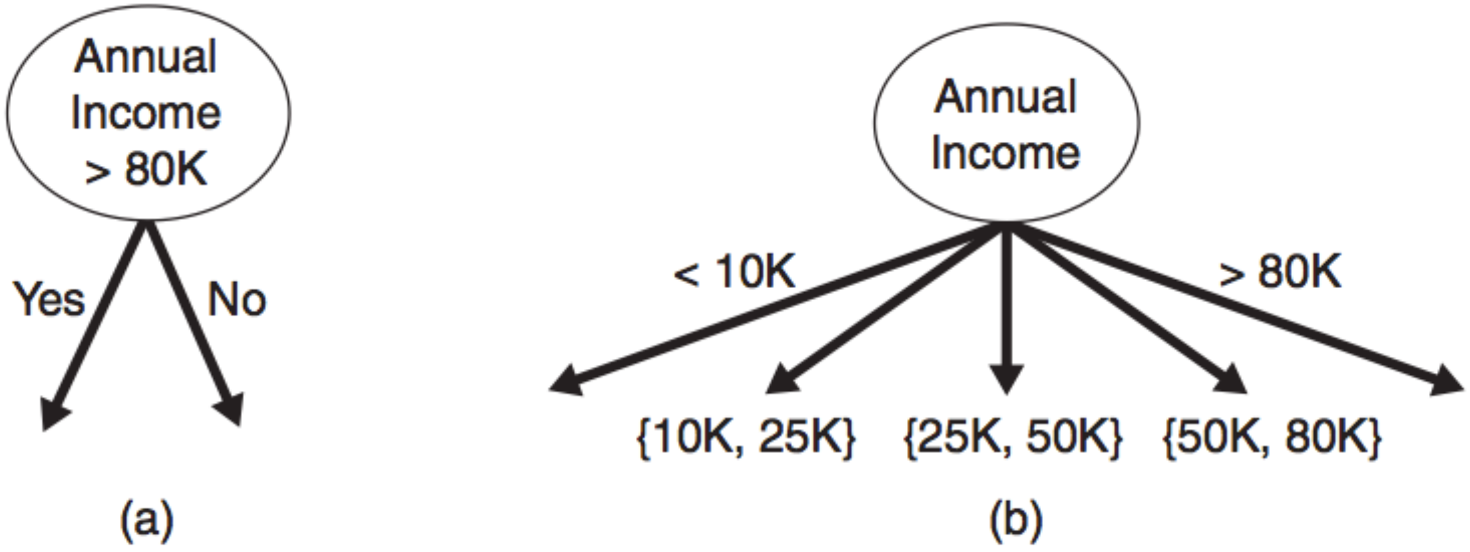


Figure 4.11. Test condition for continuous attributes.

Q: How do we partition the training records?

A: There are a few ways to do this.

For continuous features, we can use either method:

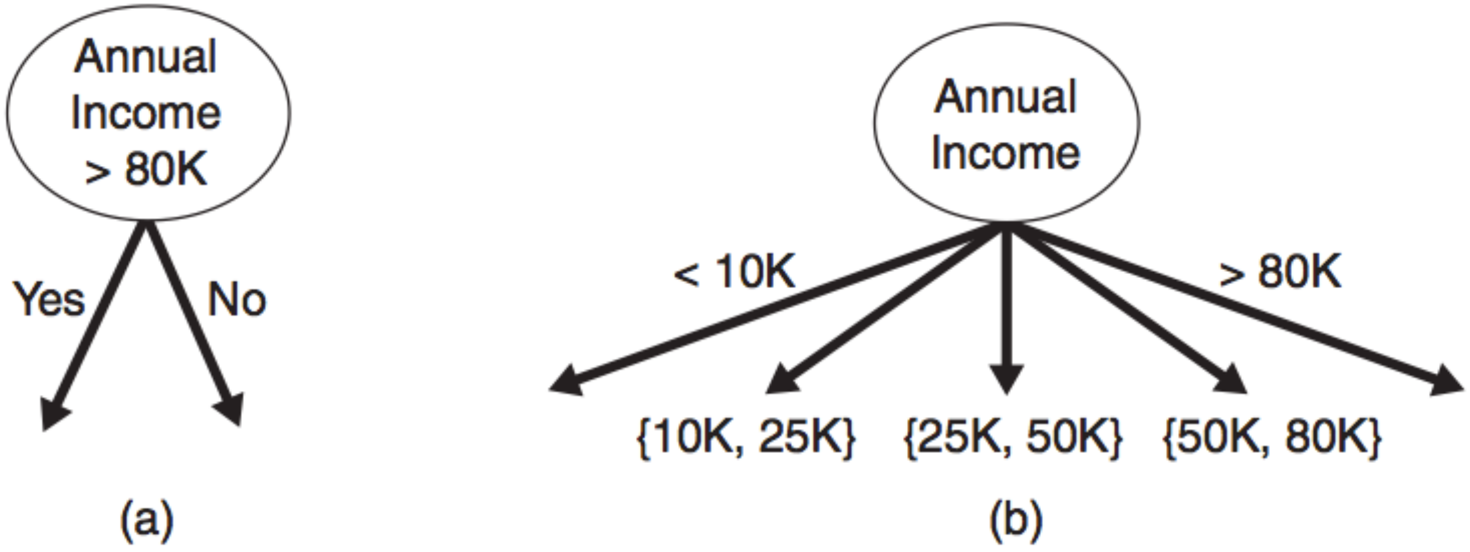


Figure 4.11. Test condition for continuous attributes.

NOTE

There are optimizations that can improve the naïve quadratic complexity of determining the optimum split point for continuous atrributes.

BUILDING A DECISION TREE

Q: How do we determine the best split?

Q: How do we determine the best split?

A: Recall that no split is necessary (at a given node) when all records belong to the same class.

BUILDING A DECISION TREE

Q: How do we determine the best split?

A: Recall that no split is necessary (at a given node) when all records belong to the same class.

Therefore we want each step to create the partition with the highest possible purity.

Q: How do we determine the best split?

A: Recall that no split is necessary (at a given node) when all records belong to the same class.

Therefore we want each step to create the partition with the highest possible purity.

We need an objective function to optimize!

III. OBJECTIVE FUNCTIONS

OBJECTIVE FUNCTIONS

We want our objective function to measure the gain in purity from a particular split.

OBJECTIVE FUNCTIONS

We want our objective function to measure the gain in purity from a particular split.

Therefore we want it to depend on the class distribution over the nodes (before and after the split).

OBJECTIVE FUNCTIONS

We want our objective function to measure the gain in purity from a particular split.

Therefore we want it to depend on the class distribution over the nodes (before and after the split).

For example, let $p(i|t)$ be the probability of class i at node t (eg, the fraction of records labeled i at node t).

OBJECTIVE FUNCTIONS

We want our objective function to measure the gain in purity from a particular split.

Therefore we want it to depend on the class distribution over the nodes (before and after the split).

For example, let $p(i|t)$ be the probability of class i at node t
fraction of records labeled i at node t).

NOTE

We are using the *frequentist* definition of probability here!

Then for a binary (0/1) classification problem,

OBJECTIVE FUNCTIONS

Then for a binary (0/1) classification problem,

The minimum purity partition is given by the distribution:

$$p(0|t) = p(1|t) = 0.5$$

Then for a binary (0/1) classification problem,

The minimum purity partition is given by the distribution:

$$p(0|t) = p(1|t) = 0.5$$

The maximum purity partition is given (eg) by the distribution:

$$p(0|t) = 1 - p(1|t) = 1$$

OBJECTIVE FUNCTIONS

Some measures of impurity include:

$$\text{Entropy(r)} = -\sum_{c \in C} p(c \mid r) \log(c \mid r)$$

$$\text{Gini(r)} = \sum_{c \in C} p(c \mid r)(1 - p(c \mid r))$$

$$\text{ClassificationError(r)} = 1 - \max_{c}[p(c \mid r)]$$

OBJECTIVE FUNCTIONS

Note that each measure achieves its max at 0.5, min at 0 & 1.

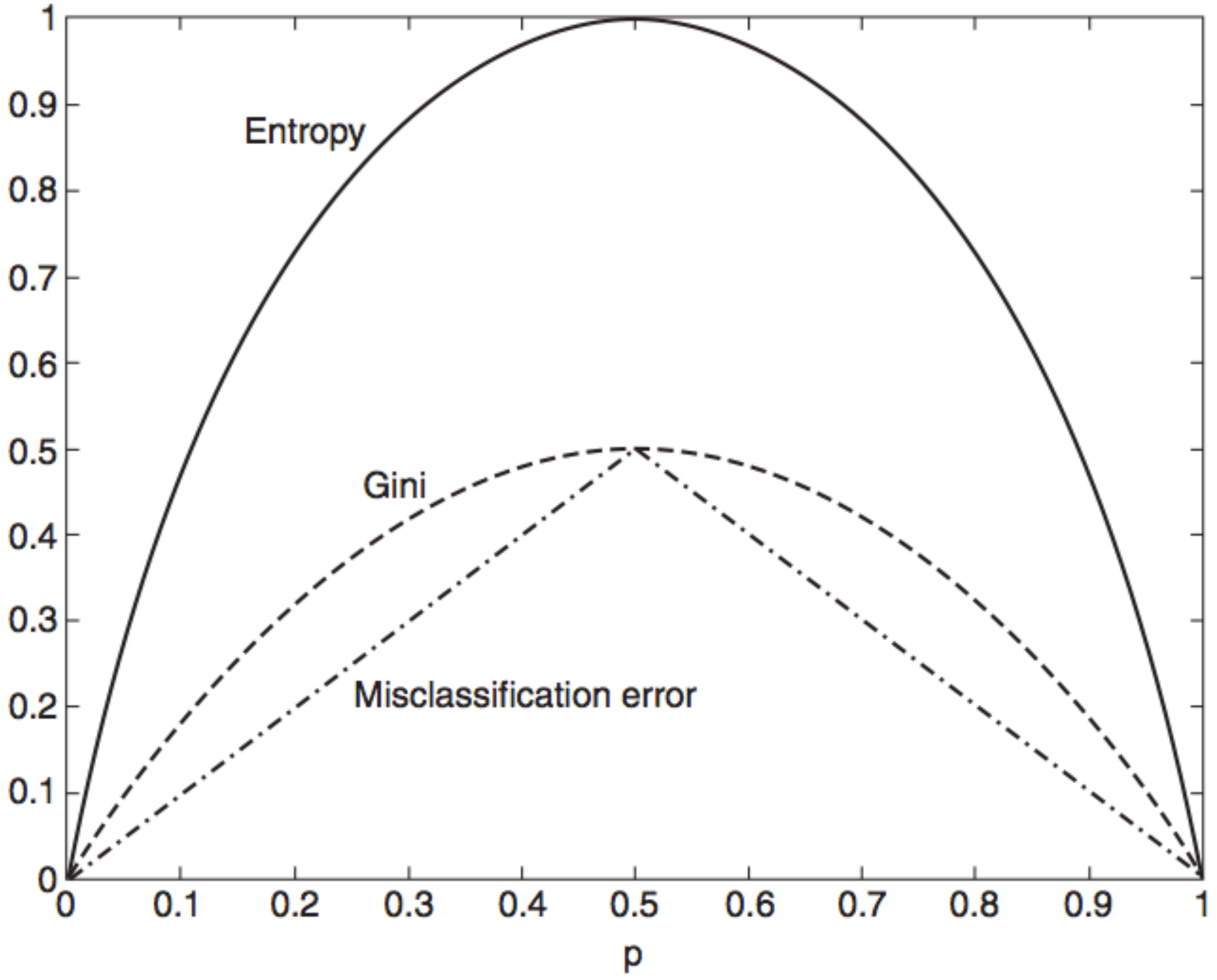


Figure 4.13. Comparison among the impurity measures for binary classification problems.

OBJECTIVE FUNCTIONS

Note that each measure achieves its max at 0.5, min at 0 & 1.

NOTE

Despite consistency, different measures may create different splits.

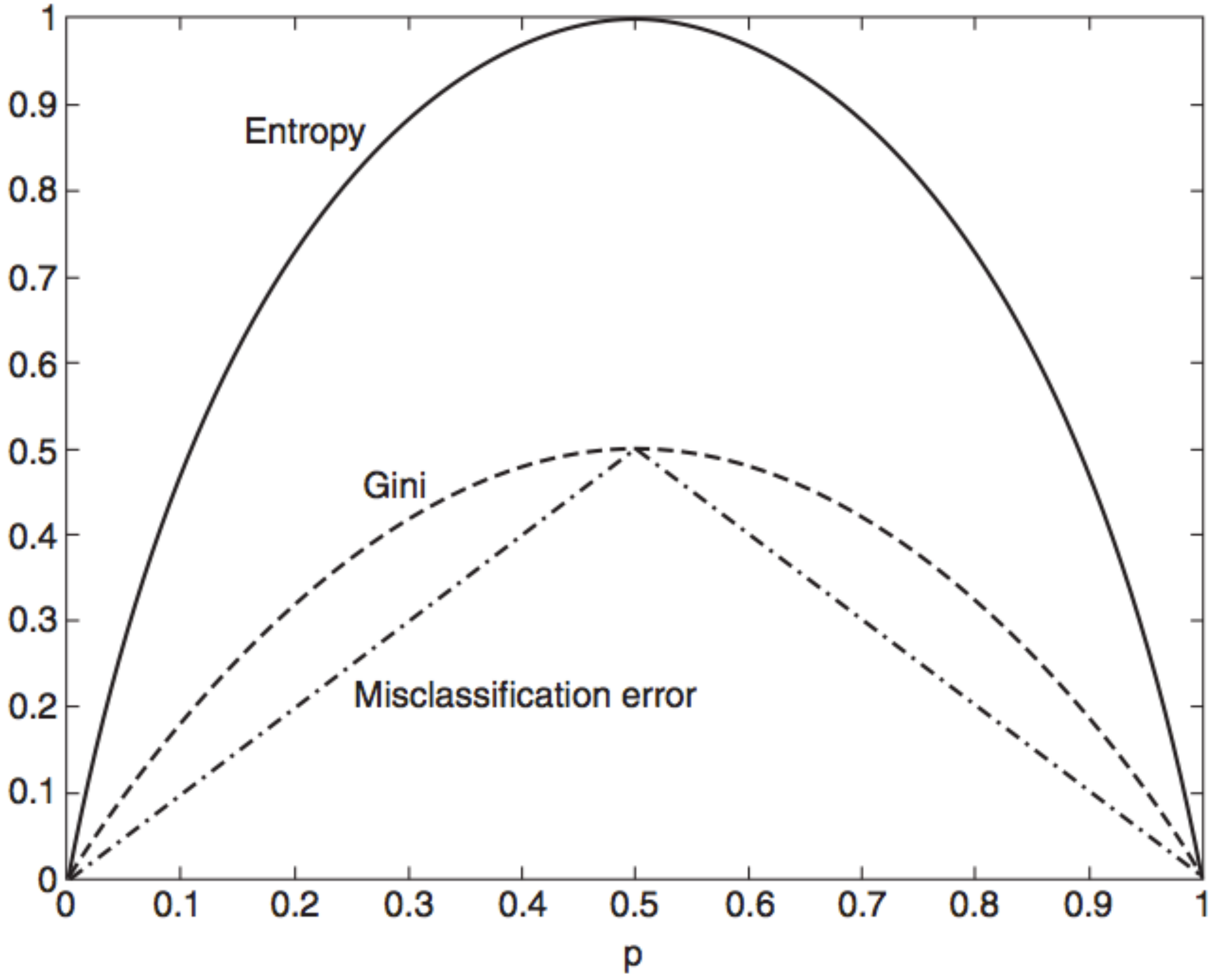


Figure 4.13. Comparison among the impurity measures for binary classification problems.

OBJECTIVE FUNCTIONS

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

OBJECTIVE FUNCTIONS

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

Q: Why is this true?

Impurity measures put us on the right track, but on their own they are not enough to tell us how our split will do.

Q: Why is this true?
A: We still need to look at impurity before & after the split.

We can make this comparison using the **gain**:

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

We can make this comparison using the ***gain****:*

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

(Here I is the impurity measure, N_j denotes the number of records at child node j, and N denotes the number of records at the parent node.)

OBJECTIVE FUNCTIONS

We can make this comparison using the **gain***:*

$$\Delta = I(\text{parent}) - \sum_{\text{children } j} \frac{N_j}{N} I(\text{child } j)$$

(Here I is the impurity measure, N_j denotes the number of records at child node j, and N denotes the number of records at the parent node.)

When I is the entropy, this quantity is called the **information gain***.*

OBJECTIVE FUNCTIONS

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

OBJECTIVE FUNCTIONS

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

One way of dealing with this is to restrict the algorithm to binary splits only (CART).

OBJECTIVE FUNCTIONS

Generally speaking, a test condition with a high number of outcomes can lead to overfitting (ex: a split with one outcome per record).

One way of dealing with this is to restrict the algorithm to binary splits only (CART).

Another way is to use a splitting criterion which explicitly penalizes the number of outcomes (C4.5)

We can use a function of the information gain called the **gain ratio** *to explicitly penalize high numbers of outcomes:*

$$\text{gain ratio} = \frac{\Delta_{info}}{-\sum p(v_i) log_2 p(v_i)}$$

(Where $p(v_i)$ refers to the probability of label i at node v)

We can use a function of the information gain called the **gain ratio** *to explicitly penalize high numbers of outcomes:*

$$\text{gain ratio} = \frac{\Delta_{info}}{-\sum p(v_i) log_2 p(v_i)}$$

(Where $p(v_i)$ refers to the probability of label i at node v)

NOTE

This is a form of regularization!

IV. REGULARIZATION

REGULARIZATION

In addition to determining splits, we also need a stopping criterion to tell us when we’re done.

REGULARIZATION

In addition to determining splits, we also need a stopping criterion to tell us when we're done.

For example, we can stop when all records belong to the same class, or when all records have the same attributes.

In addition to determining splits, we also need a stopping criterion to tell us when we're done.

For example, we can stop when all records belong to the same class, or when all records have the same attributes.

This is correct in principle, but would likely lead to overfitting.

One possibility is **pre-pruning***, which involves setting a minimum threshold on the gain, and stopping when no split achieves a gain above this threshold.*

One possibility is **pre-pruning***, which involves setting a minimum threshold on the gain, and stopping when no split achieves a gain above this threshold.*

This prevents overfitting, but is difficult to calibrate in practice (may preserve bias!)

REGULARIZATION

Alternatively we could build the full tree, and then perform **pruning** *as a post-processing step.*

Alternatively we could build the full tree, and then perform **pruning** *as a post-processing step.*

To prune a tree, we examine the nodes from the bottom-up and simplify pieces of the tree (according to some criteria).

Complicated subtrees can be replaced either with a single node, or with a simpler (child) subtree.

Complicated subtrees can be replaced either with a single node, or with a simpler (child) subtree.

The first approach is called **subtree replacement**, *and the second is* **subtree raising**.

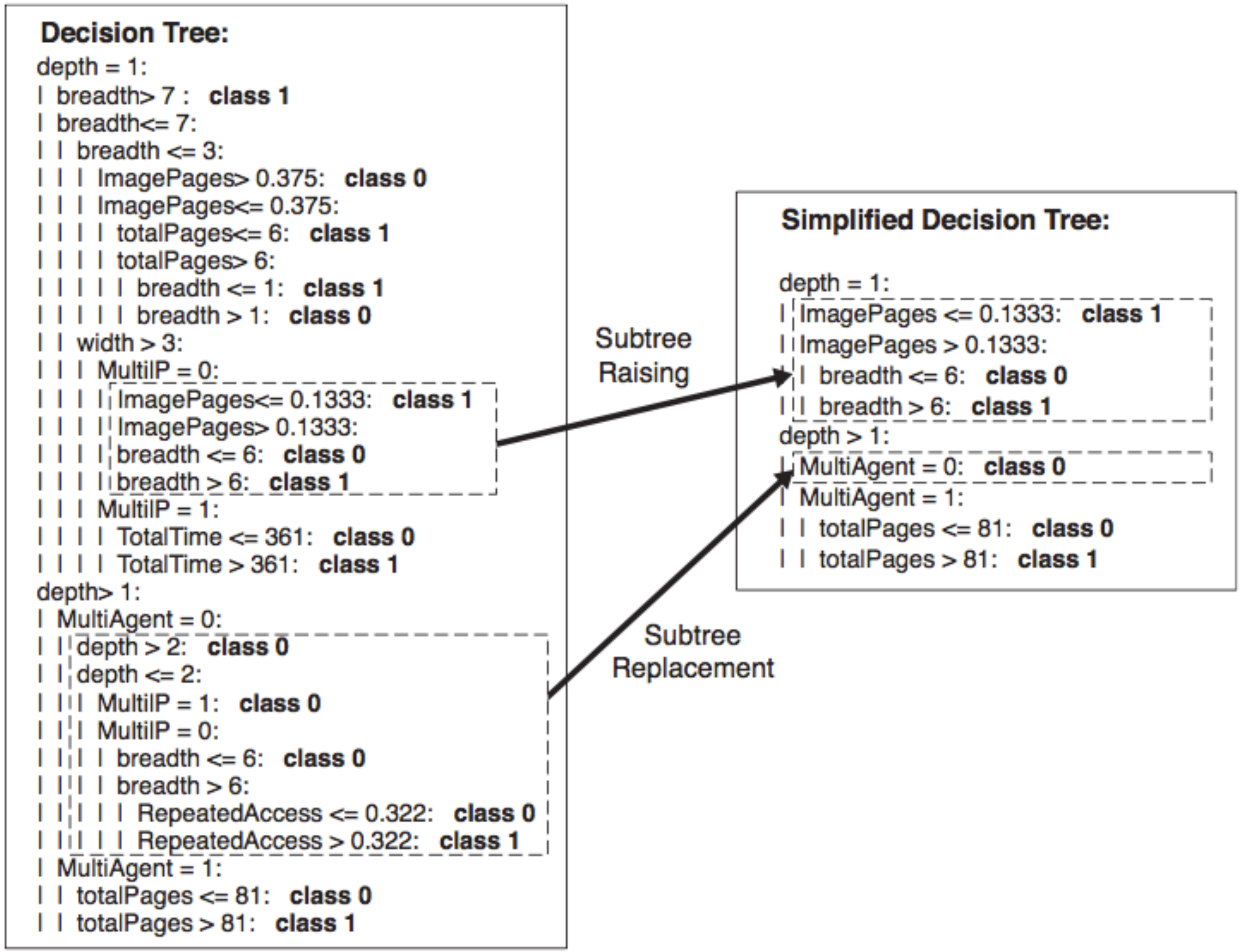


Figure 4.29. Post-pruning of the decision tree for Web robot detection.

V.PROS CONS USES

PROS AND CONS OF DECISION TREES

Pros:

- Extremely intuitive and simple to use
- Very easy to explain and visualize
- Can use any categorical or continuous variable without preprocessing!
- Can handle both numerical and categorical data and multiple classes easily

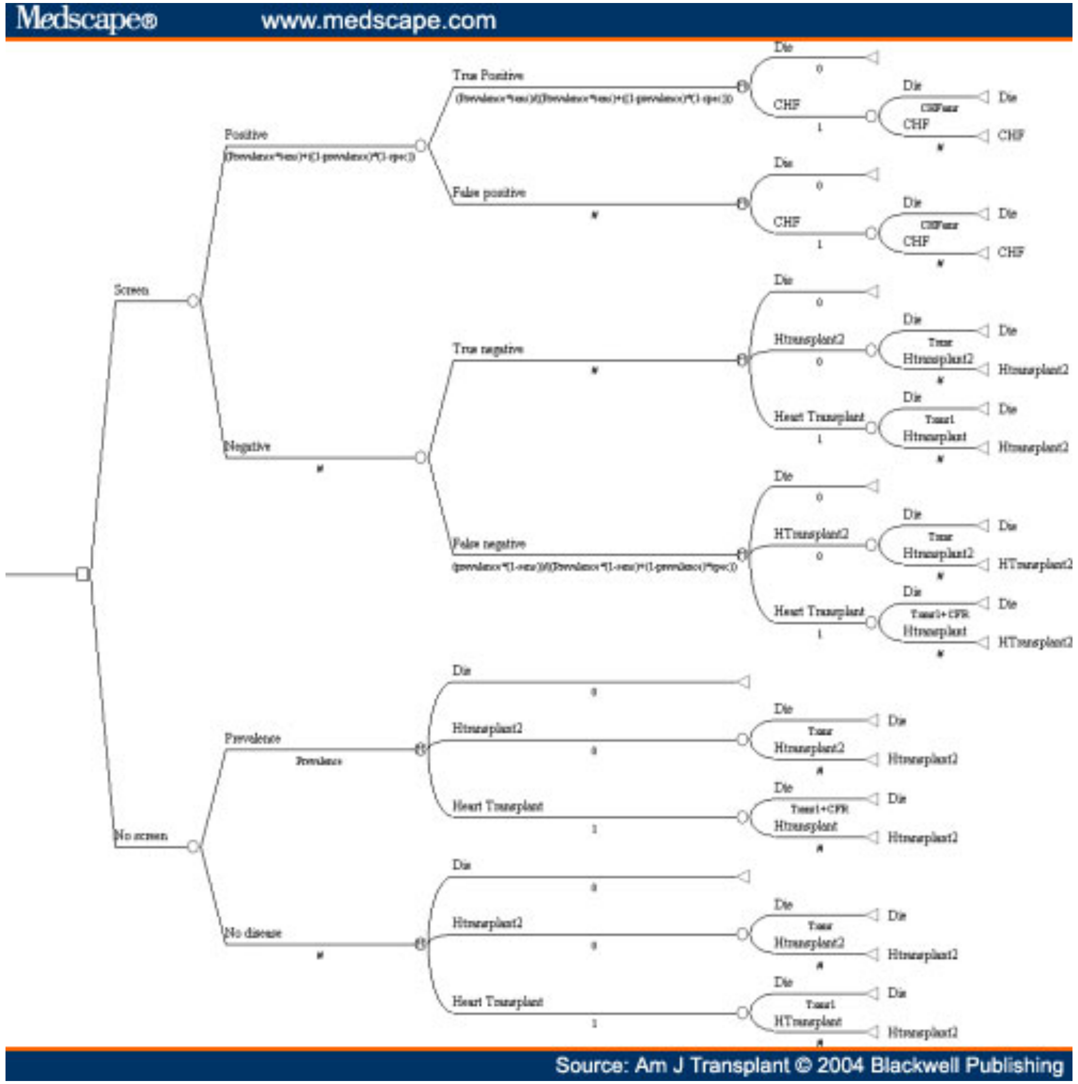
Cons:

- Prone to overfitting
- Creation of trees are unstable
- Heuristics used to create trees i.e. suboptimal
- Decision trees can create biased trees if the data is unbalanced

USES OF DECISION TREES

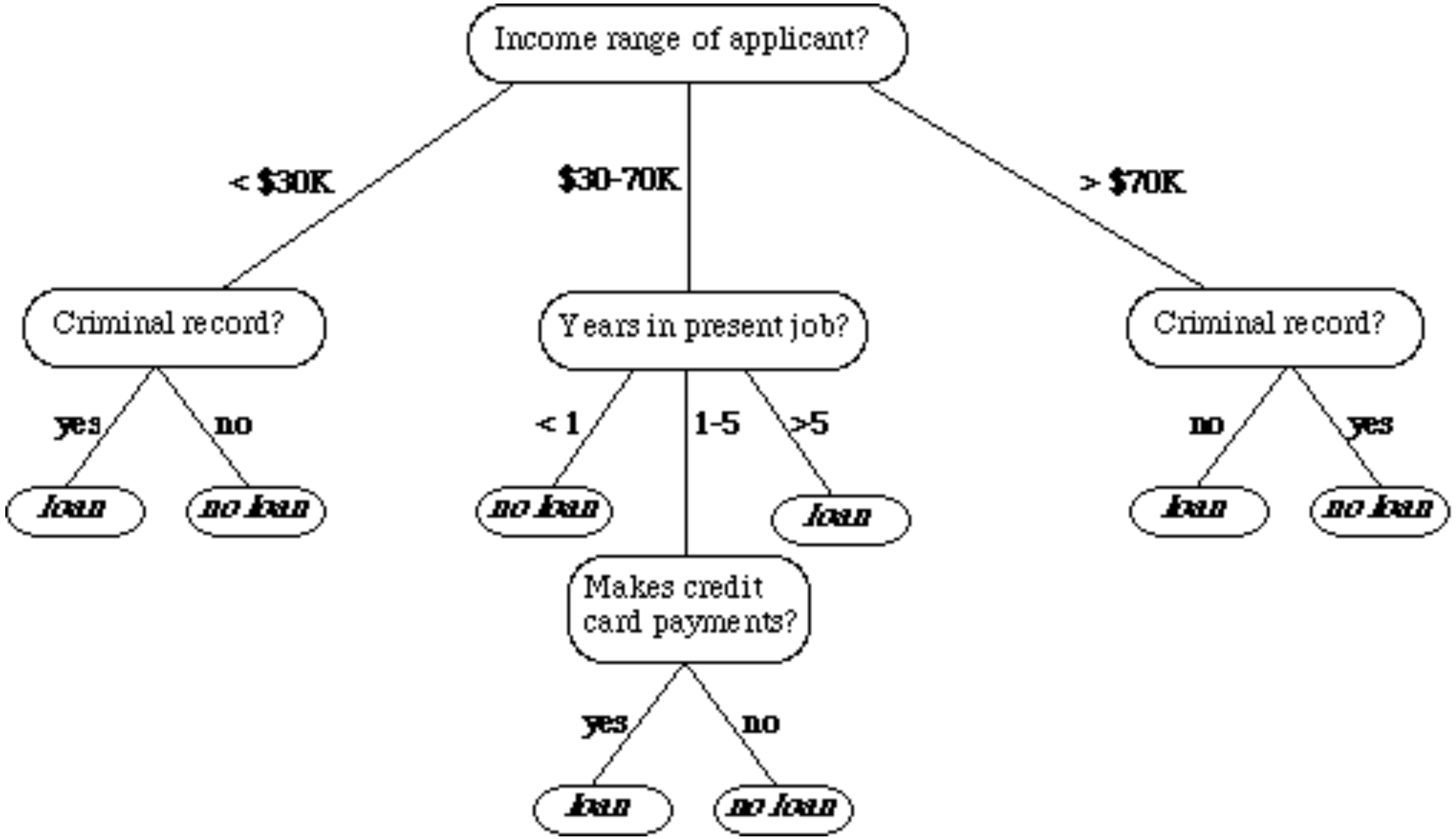
Netflix challenge

USES OF DECISION TREES



Medical diagnostics

USES OF DECISION TREES



Financial applications (i.e. intuition and explainability)

USES OF DECISION TREES

dmlc
XGBoost eXtreme Gradient Boosting

Underlying mechanism for EXTREME models

VI. DECISION TREE LAB

VII.RANDOM FORESTS

RANDOM FORESTS

A random forest is an ***ensemble*** *of decision trees where each base classifier is grown using a random effect.*

RANDOM FORESTS

*A random forest is an **ensemble** of decision trees where each base classifier is grown using a random effect.*

One way to do this is to choose the top feature amongst k randomly chosen features.

A random forest is an ***ensemble*** *of decision trees where each base classifier is grown using a random effect.*

One way to do this is to choose the top feature amongst k randomly chosen features.

For a small number of features, we can also create linear combinations of features and select splits from the enhanced feature set (Forest-RC)

*A random forest is an **ensemble** of decision trees where each base classifier is grown using a random effect.*

One way to do this is to choose the top feature amongst k randomly chosen features.

For a small number of features, we can also create linear combinations of features and select splits from the enhanced feature set (Forest-RC)
Or we can select splitting features completely at random (Forest-RI)

THAT'S IT!

- Exit Tickets: DAT1 - Lesson 12 - Trees
- Pretty good video series https://www.youtube.com/watch?v=p17C9q2M00Q
- Milestone 3 is due Feb 17 https://github.com/brianchandbound/ga-ds/blob/master/extra/project.md
- Peer Reviews
- Next week, we build upon trees into Ensemble methods