Lecture 8: Supervised Learning Pt. 2

More Models, Bootstrapping, and Bagging

INFO 1998: Introduction to Machine Learning

CDS Education

Agenda

1. Decision Trees
2. Logistic Regression
3. Validation Techniques
    - Bootstrapping & Bagging

Decision Trees

Source

Asking Questions about Data

- Two Formulations:
  - How do we capture the structure of our data?
  - Or broadly, how do we make decisions?
- In both cases, we ask questions.
- Suppose we're deciding whether to play tennis
  - What questions can we ask ourselves?
  -

Scenario: Should we play tennis today?

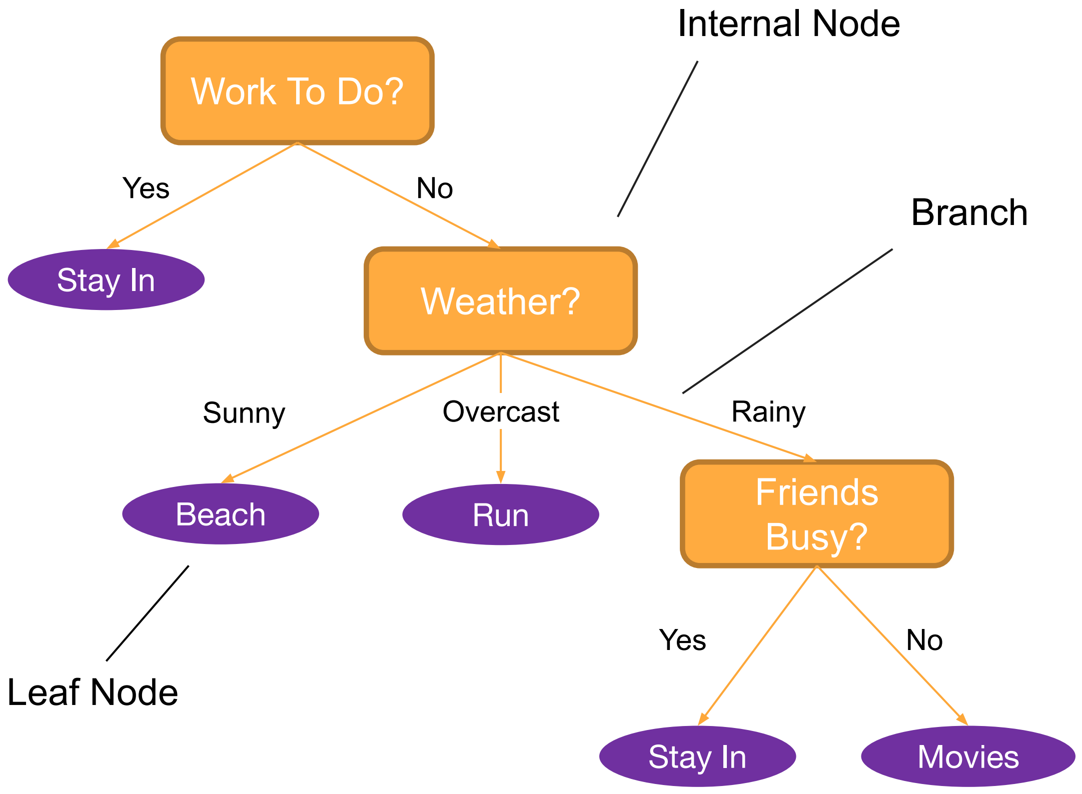
- Is it raining?
  - If so, no.
  - If not, continue…
- Is it too hot?
  - If so, no.
  - If not, continue.
- Do I have a racket?
  - If not, no.
  - If so, continue.
- Etc…

Source

The Decision tree

Decision Tree

- *supervised* machine learning model
- **breaking down** our data by **making a decision** based on **asking a series of questions** based on features



Intuition

- Suppose we have a big dataset with several features
    - We can split at several points along these features!
    - In fact, if these features are categorical we can ask as many questions as we have total possibilities!
- Every single question splits the data!
- *So the best questions will try to split the data as much as possible*
- These trees are nice and interpretable when we do them by hand
- But we can introduce ML to learn good decision trees based on our data

Learning Decision Trees

We will use a recursive partitioning algorithm

1. Start with all your data at the root node.
2. Find the best feature to split your data that creates the “purest” child nodes.
    a. What is purity?
3. Create child nodes based on this split.
4. Repeat the process for each child node until some stopping condition.
    a. Place some limit on the depth – hyperparameter.

Making Good Splits

- Entropy & Information Gain
- What is entropy in chemistry?

Entropy

- Entropy is the measure of how disorderly a dataset is. Maximum entropy is reached when all classes are equally likely to happen in our dataset.
- Minimum entropy happens when all points in the dataset are 0.

$$H = -\sum p(x) \log p(x)$$

Information Gain

- We want to create splits that minimize this entropy.
- Information gain is how much a particular split reduces entropy
- **We want to choose splits that give us the highest information gain.**

Working Example

Student	Study Hours	Attendance (%)	Result
Alex	7	85	Pass
Bailey	3	60	Fail
Casey	6	90	Pass
Dana	2	55	Fail
Eli	8	95	Pass
Fran	4	65	Fail
Glen	7	80	Pass
Harper	3	70	Fail
Indira	9	95	Pass
Jamie	2	50	Fail

Will students pass or fail: Calculate original entropy

- We have 10 students. 5 pass, 5 fail.
- The features we have are Study Hours, and Attendance.

We compute the entropy of our original dataset.

Consider Potential Splits

- Let's evaluate two possible splitting features:

1. Study Hours (with threshold > 6)
2. Study Hours (with threshold > 5)

Option 1: Split by "Study Hours > 6"

Left group (Study Hours > 6):

Alex (7 hours) → Pass
Eli (8 hours) → Pass
Glen (7 hours) → Pass
Indira (9 hours) → Pass
Result: 4 Pass, 0 Fail

Right group (Study Hours ≤ 6):

Bailey (3 hours) → Fail
Casey (6 hours) → Pass
Dana (2 hours) → Fail
Fran (4 hours) → Fail
Harper (3 hours) → Fail
Jamie (2 hours) → Fail
Result: 1 Pass, 5 Fail

Entropy = 0.65

Option 2: Split by “Study Hours > 5”

Left Group:

Alex (7 hours) → Pass
Casey (6 hours) → Pass
Eli (8 hours) → Pass
Glen (7 hours) → Pass
Indira (9 hours) → Pass
Result: 5 Pass, 0 Fail

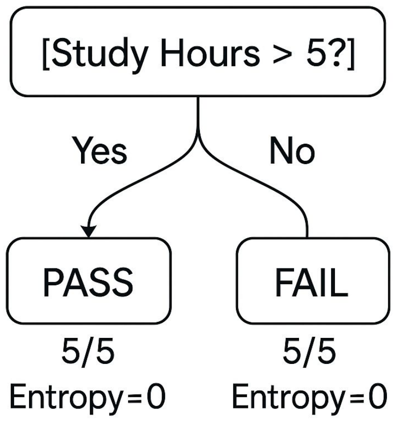
Right group (Study Hours ≤ 5):

Bailey (3 hours) → Fail
Dana (2 hours) → Fail
Fran (4 hours) → Fail
Harper (3 hours) → Fail
Jamie (2 hours) → Fail
Result: 0 Pass, 5 Fail

Entropy = 0

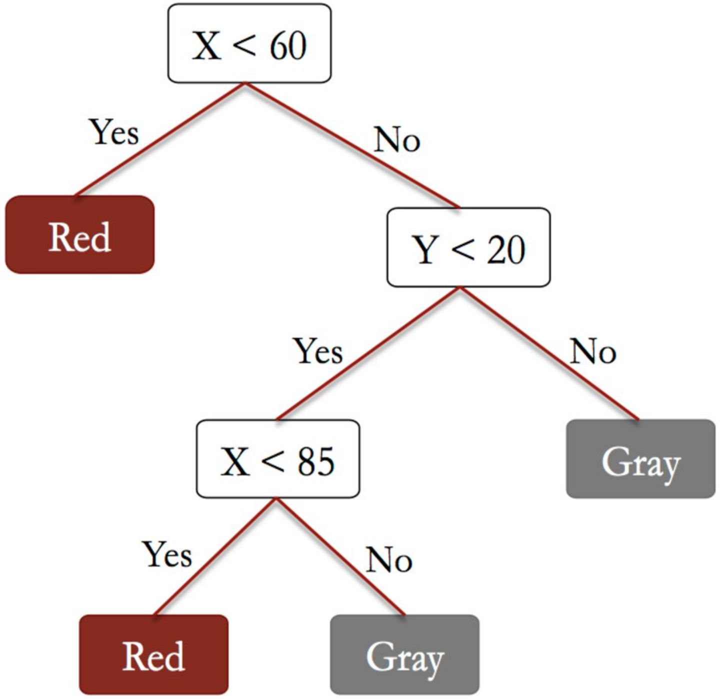
Key Insights

1. The algorithm chooses the split that maximizes information gain
2. Good splits create purer groups in terms of the target variable
3. The best splits reduce entropy as much as possible
4. In real datasets, perfect splits are rare - the algorithm seeks the best available split
5. This process is applied recursively to build the complete tree



CART (Classification and Regression Trees)

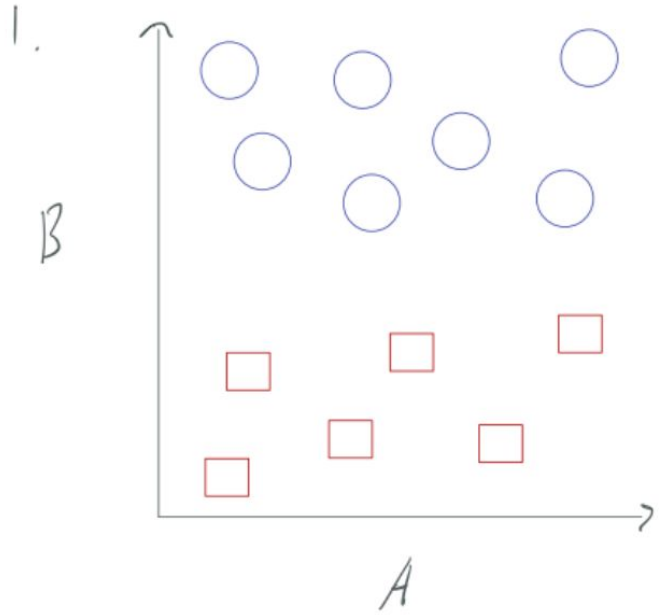
- Used for Classification and Regression
- At each node, split on variables
- Each split minimizes error/impurity function
- Very interpretable
- <u>Models a non-linear relationship!</u>

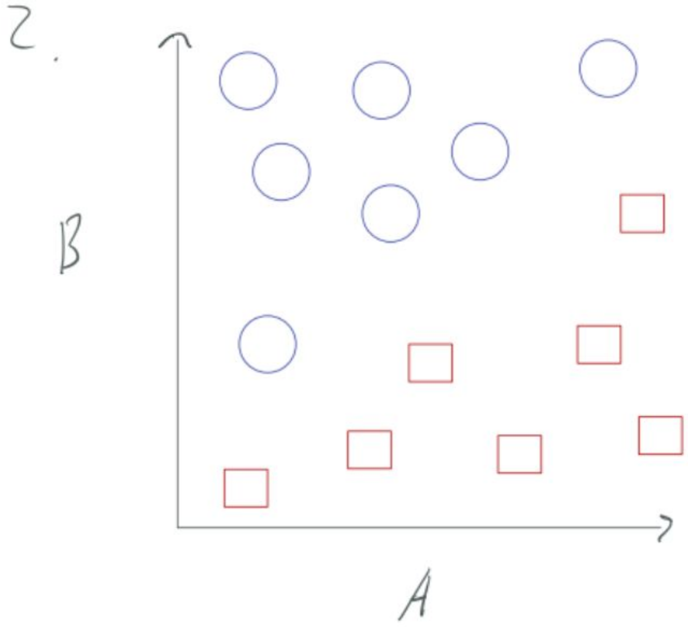


Pros and Cons of Using Decision Trees

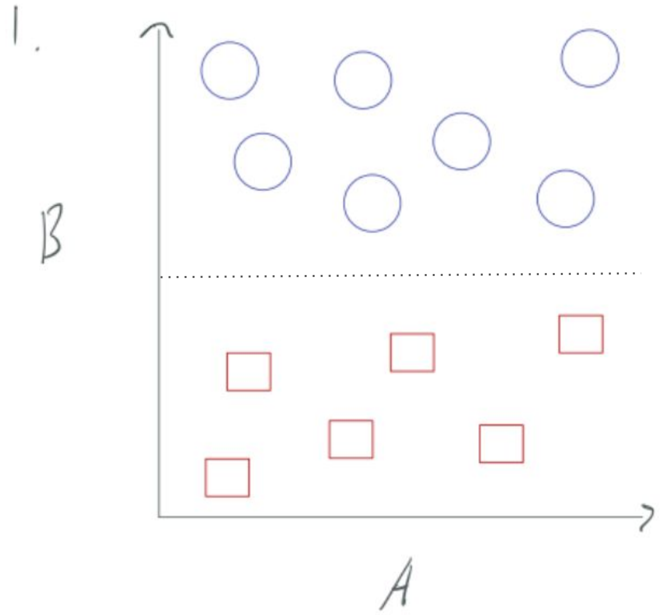
Pros	Cons
Easy to interpret	Overfitting ☹️
Requires little data preparation (robust to missing data)	Requires parameter tuning (max depth)
Can use a lot of features	Can only make horizontal/vertical splits (solvable with feat. eng. / ensembling)
Can capture non-linear relationships	

What would these decision boundaries look like?

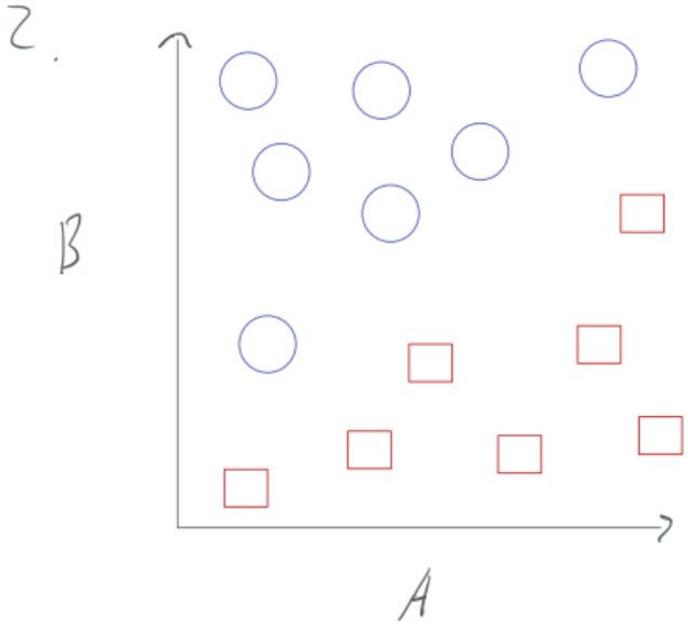




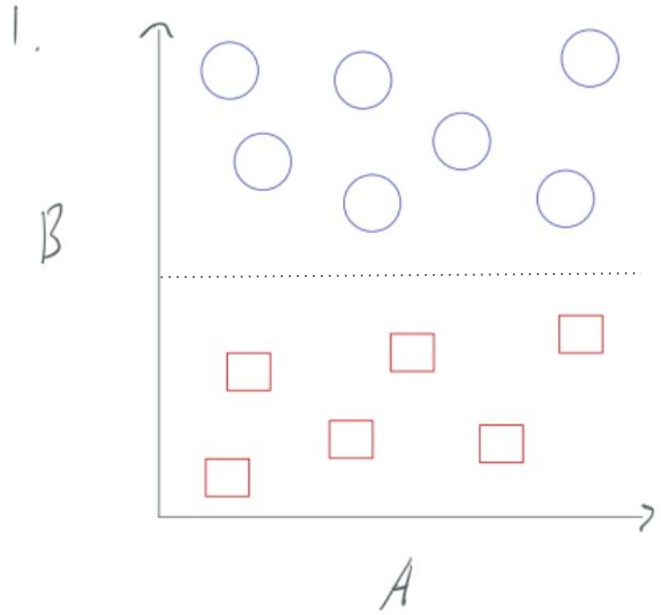
What would these decision boundaries look like?



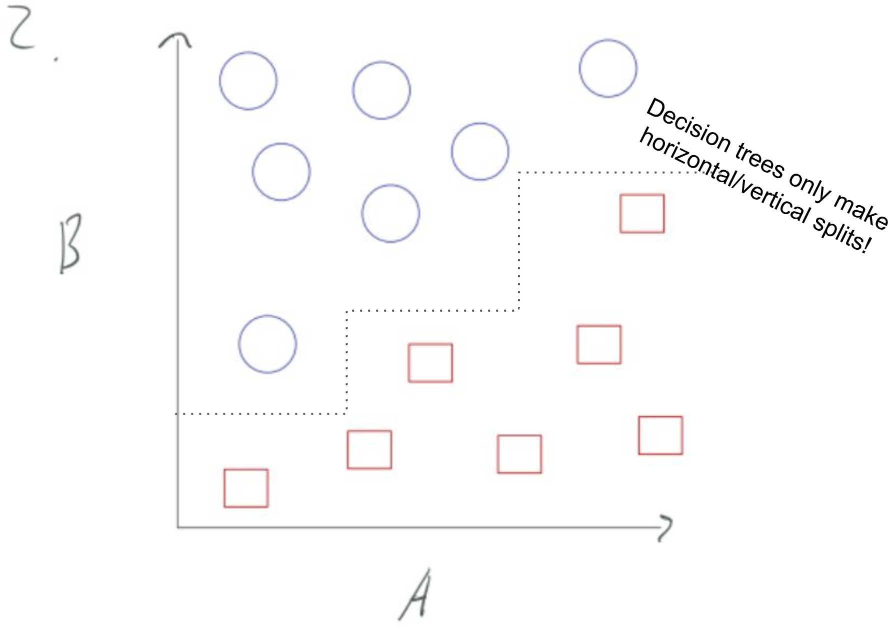
“If B less than this value, it’s a red square. Otherwise, it’s a blue circle.”



What would these decision boundaries look like?

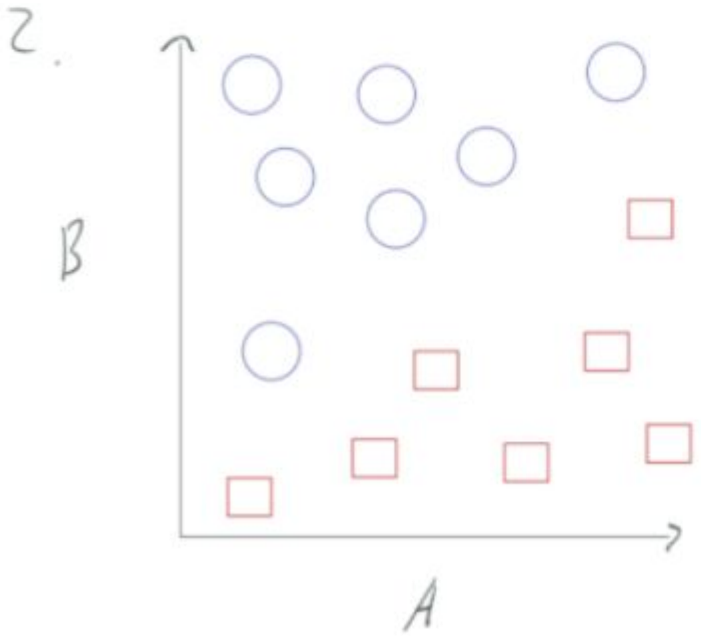


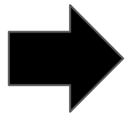
“If B less than this value, it’s a red square. Otherwise, it’s a blue circle.”



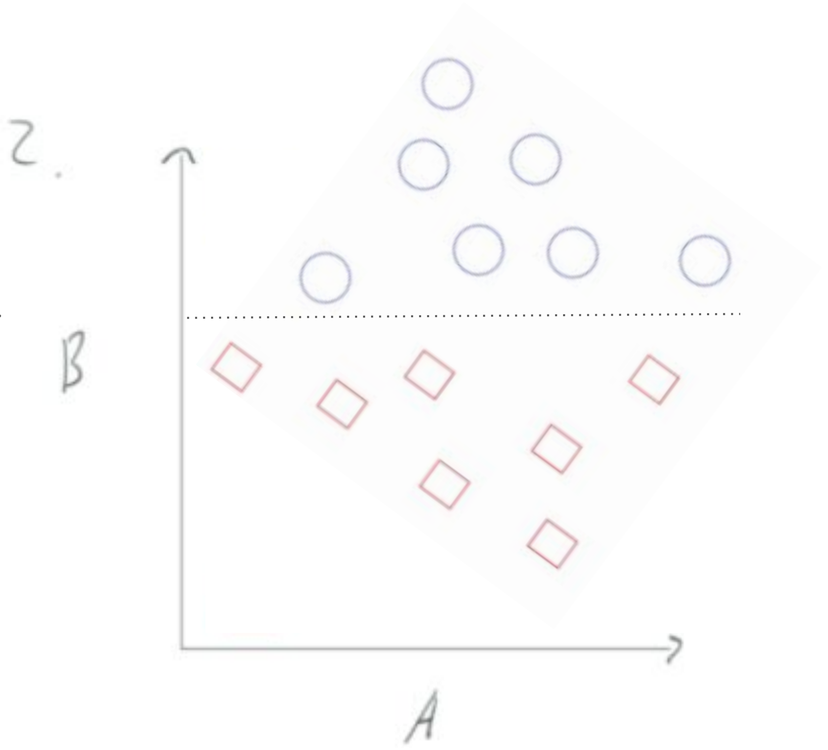
Uhhh…

But...



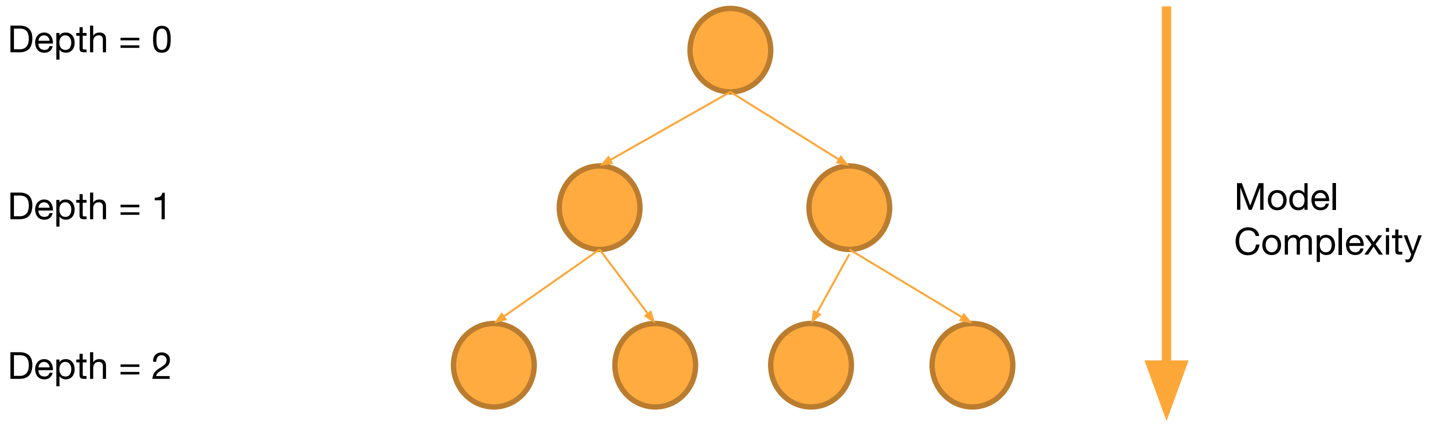


Rotate!



How to Reduce Overfitting

1. Limit the max depth of the tree

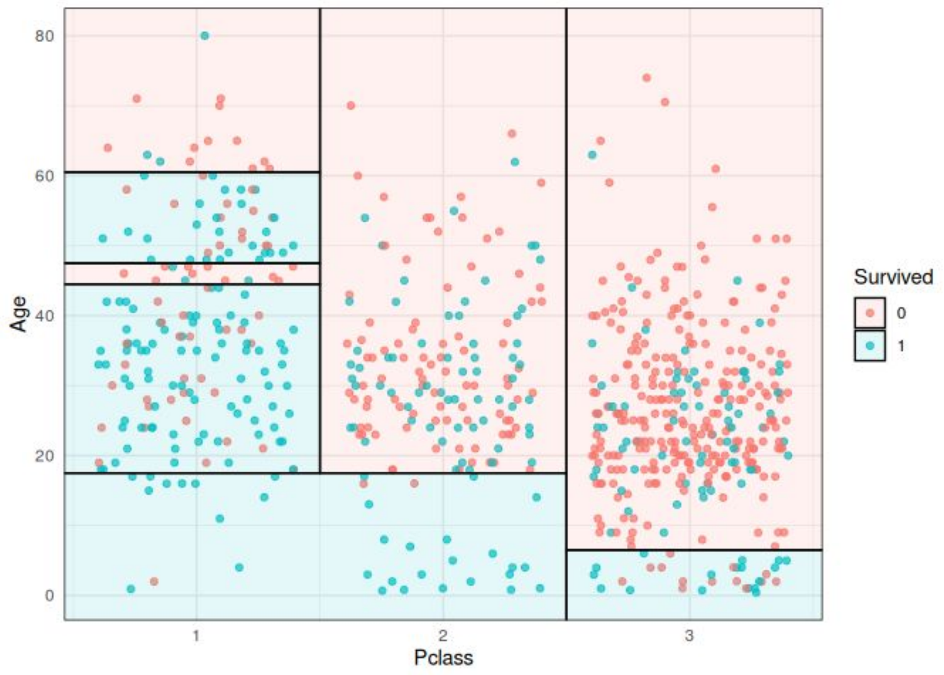


When training a decision tree, we have to specify the maximum depth a constructed tree can have

Source

How to Reduce Overfitting

- There are no “curves” for each decision tree boundary line
- Limiting the depth of the tree limits the number of lines you are splitting on

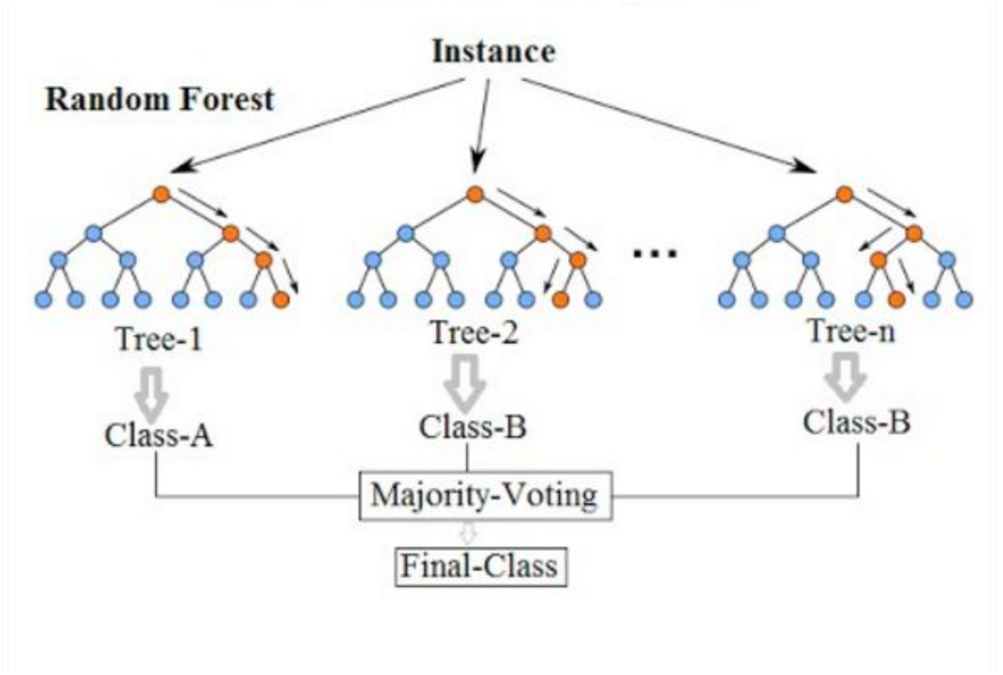


Source

How to Reduce Overfitting

2) Train multiple decision trees and determine final output based on output of each decision tree

This is called a
Random Forest Classifier



Demo

Lecture 8: Supervised Learning Pt. 2

INFO 1998: Introduction to Machine Learning

Attendance!

CDS Education

Logistic Regression

Logistic Regression

- Used for Binary Classification:

$$Y = \begin{cases} 1 \\ 0 \end{cases}$$

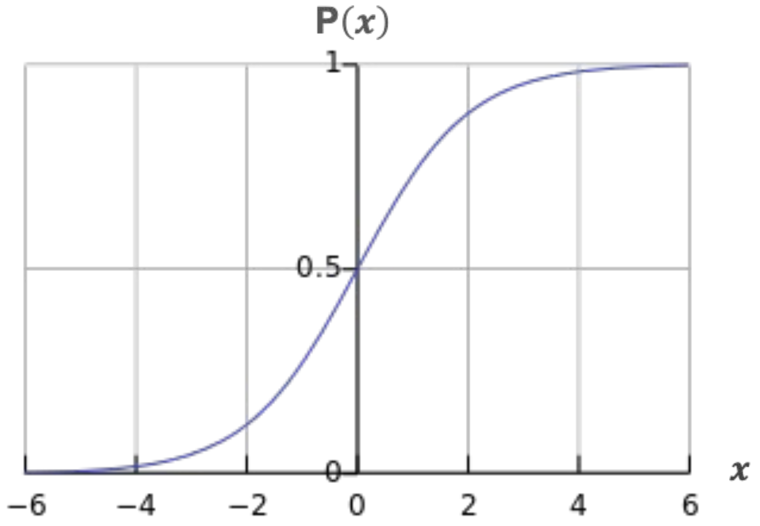
- Fits a linear relationship between the variables
- Transforms the linear relationship of probability that the outcome is 1 by using the **sigmoid function**

Formula:

$$P(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k)}} \longrightarrow \ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k$$

Logistic Function

$$P(x) = \frac{1}{1 + e^{-x}}$$



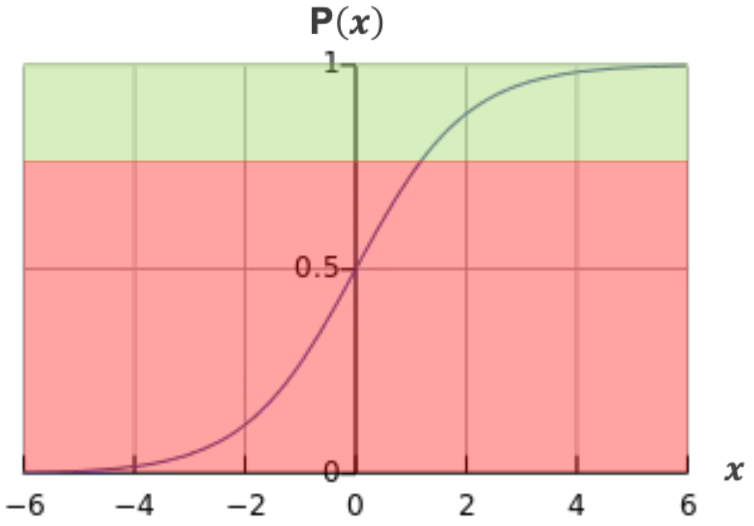
The Logistic Function "**squeezes"** numbers to be between 0 and 1

→ Allows us to interpret our prediction as a "**probability**" that something is true

Threshold

At what point do we differentiate between our classifications?

- f(x) below threshold: predict 0
- f(x) above threshold: predict 1



Pros and Cons of Using Logistic Regression

Pros	Cons
Easy to interpret (probability)	Only Capable of Binary Classification
Computationally efficient to compute	No closed form solution (requires use of optimization algorithms)
Does not require parameter tuning	

Logistic Regression is a simple model, therefore, oftentimes it is used as a good “baseline” to compare more complex models to

Validation Techniques

Review: Regression vs. Classification

Regression

- Predict Continuous Data
- “On average, **how wrong** are we?”

Classification

- Predict Discrete or Categorical data
- “**How many** points do we get wrong?”

Numbers ≠ Continuous

Leave-P-out

Let **D** be our whole dataset

Choose a **P**

For every combination of **P** points in **D**:

    Use a train/test split with those **P** points as test, the rest as train

Leave-P-out: different from K-fold!

Let's say **D** has a size of 4. There are four data points: *a*, *b*, *c*, and *d*.

K-fold:

- K = 2.
- Each fold has a size of 2: {*a*,*b*} and {*c*,*d*}
- So, we only have 2 possible test sets:

  {*a*,*b*} and {*c*,*d*}

Leave-P-out:

- P = 2.
- We have 6 possible test sets:

  {*a*,*b*}, {*a*,*c*}, {*a*,*d*}, {*b*,*c*}, {*b*,*d*}, and {*c*,*d*}

Leave-P-out

Pros:

- Dependable (not random)
- Representative — checks all combinations

Cons:

- Slow!
    - Runtime <u>increases</u> with larger datasets
    - Runtime <u>explodes</u> with larger P

Monte Carlo Cross Validation

- Getting accuracy **1** time doesn’t tell us much
- Getting accuracy **2** times tells us a bit
- Getting accuracy **3** times tells us a bit more
- ...
- Getting accuracy **N** times might be good enough!

Take the average of those **N** times

Monte Carlo CV

- Need to use **new**, **random** train/test split each time
  - If you use the same train/test split each time, you're not getting any new information!
- Pros:
  - easy to implement
  - easy to make faster/slower by changing number of iterations
- Cons:
  - random -> train/test splits not guaranteed to be representative of dataset (might overlap, or miss some data)
  - harder to calculate how many iterations you need

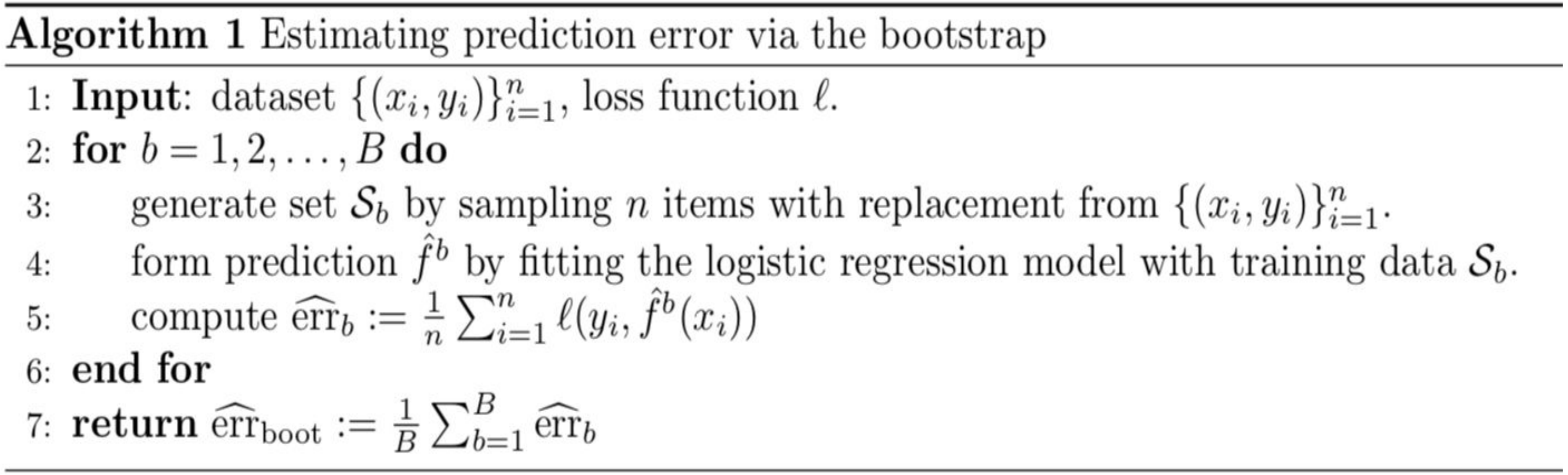
The Bootstrap

What if we don’t have enough data?

- Use **bootstrap datasets** to approximate the test error
- **Sample with replacement** from the original training dataset (with n samples) to generate **bootstrap datasets** of size n
  - Some data points may appear more than once in the generated data
  - Some data points may not appear
- Estimate of test error = average error among bootstrap datasets

Demo

Pipeline of the Bootstrap



Algorithm 1 Estimating prediction error via the bootstrap

```
1: Input: dataset $\{(x_i, y_i)\}_{i=1}^n$, loss function $\ell$.
2: for $b = 1, 2, \ldots, B$ do
3:     generate set $\mathcal{S}_b$ by sampling $n$ items with replacement from $\{(x_i, y_i)\}_{i=1}^n$.
4:     form prediction $\hat{f}^b$ by fitting the logistic regression model with training data $\mathcal{S}_b$.
5:     compute $\widehat{\mathrm{err}}_b := \frac{1}{n} \sum_{i=1}^n \ell(y_i, \hat{f}^b(x_i))$
6: end for
7: return $\widehat{\mathrm{err}}_{\mathrm{boot}} := \frac{1}{B} \sum_{b=1}^B \widehat{\mathrm{err}}_b$
```

❖ Does this give very good estimate of the prediction error?

❖ If **no,** why not? Does it **underestimate** or **overestimate** the prediction error?

Pipeline of the Bootstrap

No.

Since the bootstrap is sampling with replacement for B validation folds, each fold would have significant overlap with the original data used for training. Approximately, **2/3** of the training data would appear in each validation fold.

This leads to significant **underestimation** of the prediction error.

- ❖ Why **2/3**?

Proof

Suppose there are n samples *(yi, xi)* in the training set data, then for each of the bootstrap validation fold b, every sample has probability **1/n** of being selected into b. The probability of each data sample *(yi, xi)* not being in fold b is **(1 - 1/n)**, respectively.

Probability of **avoid selecting** all n data points in fold b: **(1 - 1/n)^n**.

When n gets large, we have this probability converges to 1/e.
(Why this converges? Probably should review Calc I, or see:
https://math.stackexchange.com/questions/882741/limit-of-1-x-nn-when-n-tends-to-infinity.)

Therefore, the fraction of overlapping data points in each fold is **(1-1/e)**, which is about **2/3**.

How we fix the problem

- Requires some thoughts when generating validation set, especially for real-world complex data.
- Can partly fix this problem by only using predictions for those observations that did not (by chance) occur in the current bootstrap sample.
- But the method gets complicated.

Bootstrap vs. k-fold

In K-fold validation, each of the K folds is distinct from the other $(K - 1)$ folds used for training: there is **no overlap**.

This is crucial for its success in estimating prediction error.

Why do we still use Bootstrap?

- Bootstrap allows us to use a computer to mimic the process of obtaining new data sets.
- Can be used to quantify the uncertainty associated with a given estimator or statistical learning method.
- Provides an estimate of the standard error of a coefficient, or a confidence interval for that coefficient.
  - i.e. the variability of the model!

Bagging (Bootstrap Aggregating)

What if we don’t have enough data?

- Bagging is a common technique that builds on Bootstrapping

- Main Idea: Do Bootstrapping a bunch and make a classifier for each bootstrap, then choose majority prediction.

- Many weak learners aggregated typically outperform a single learner over the entire set, and overfits less.
    - Principle behind Random Forests (“forest” of decision trees)

Coming Up

- **Assignment 7:** Due tomorrow at 11:59pm
- **Assignment 8**: Due April 14th
- **Next Lecture**: Applications of *Un*supervised Learning

CDS Education