



S&DS 265 / 565  
Introductory Machine Learning

# Trees and Forests

October 3

Yale

# Reminders

- Assn 2 out; due Thursday at midnight
- Quiz 2 last week
- Midterm in class on Tuesday, October 17
- Questions?

# Classification and Regression Trees (CART)

Trees provide ways of modeling nonlinear relationships by carving out *rectangular regions* in the feature space.

# Classification and Regression Trees (CART)

Trees provide ways of modeling nonlinear relationships by carving out *rectangular regions* in the feature space.

- Response variables can be categorical or quantitative

# Classification and Regression Trees (CART)

Trees provide ways of modeling nonlinear relationships by carving out *rectangular regions* in the feature space.


- Response variables can be categorical or quantitative
- Yields a set of **interpretable decision rules**

# Classification and Regression Trees (CART)


Trees provide ways of modeling nonlinear relationships by carving out *rectangular regions* in the feature space.

- Response variables can be categorical or quantitative
- Yields a set of **interpretable decision rules**
- Predictive ability is mediocre, *but* can be improved by combining multiple trees (resampling, ensemble methods)

# Titanic data


 Search

Sign In

 Getting Started Prediction Competition

## Titanic: Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

 Kaggle · 17,691 teams · Ongoing

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Join Competition](#)

Overview

[Description](#)

Evaluation

Frequently Asked Questions

👋🎉 **Ahoy, welcome to Kaggle! You're in the right place.**

This is the legendary Titanic ML competition – the best, first challenge for you to dive into ML competitions and familiarize yourself with how the Kaggle platform works.

The competition is simple: use machine learning to create a model that predicts which passengers survived the Titanic shipwreck.

Read on or watch the video below to explore more details. Once you're ready to start competing, click on the ["Join Competition button"](#) to create an account and gain access to the [competition data](#). Then check out [Alexis Cook's Titanic Tutorial](#) that walks you through step by step how to make your first submission!

# Titanic data

- **Survived:** Outcome of survival (0 = No; 1 = Yes)
- **Pclass:** Socio-economic class (1 = Upper class; 2 = Middle class; 3 = Lower class)
- **Name:** Name of passenger
- **Sex:** Sex of the passenger
- **Age:** Age of the passenger (Some entries contain NaN)
- **SibSp:** Number of siblings and spouses of the passenger aboard
- **Parch:** Number of parents and children of the passenger aboard
- **Ticket:** Ticket number of the passenger
- **Fare:** Fare paid by the passenger
- **Cabin** Cabin number of the passenger (Some entries contain NaN)
- **Embarked:** Port of embarkation of the passenger (C = Cherbourg; Q = Queenstown; S = Southampton)



# Trees



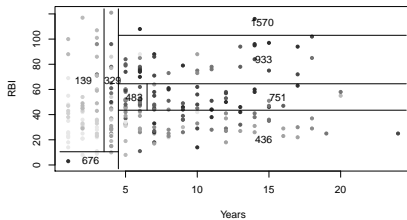
# Trees



# Trees

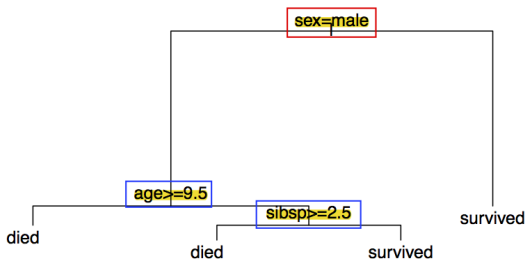


## Modeling Titanic survival:



# Tree terminology

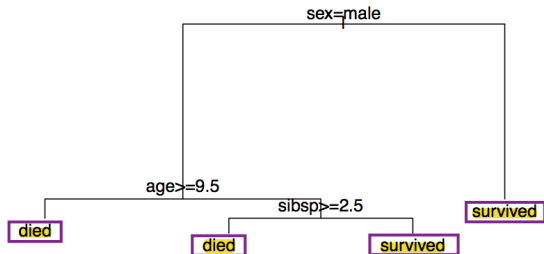
**Internal nodes** are points where the predictor space is split.



The internal node at the top is the **root** of the tree.

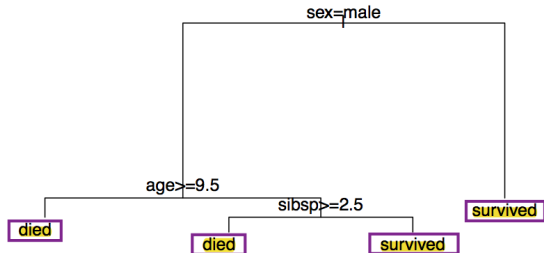
# Tree terminology

**Terminal nodes** (or **leaves**) are the ends of the tree where no further splitting occurs.



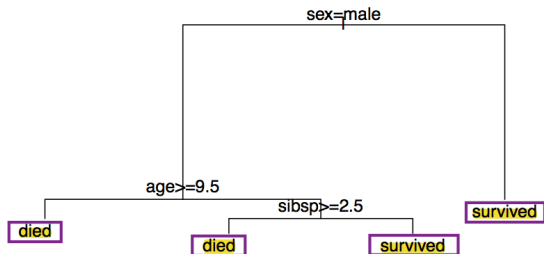
# Tree terminology

**Terminal nodes** (or **leaves**) are the ends of the tree where no further splitting occurs.



Denote these  $J$  regions as  $R_1, \dots, R_J$ .

# Tree terminology



- $R_1 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i \geq 9.5\}$
- $R_2 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i \geq 2.5\}$
- $R_3 = \{i : \text{sex}_i = \text{male} \cap \text{age}_i < 9.5 \cap \text{sibsp}_i < 2.5\}$
- $R_4 = \{i : \text{sex}_i \neq \text{male}\}$

Let's go to the Titanic demo



# Bias-variance

- Nodes are split by greedily choosing the best question (greatest reduction in error)
- As tree is grown deeper, bias decreases
- But the variance increases
- How to choose the right size of tree?

# Stopping criterion

Once we stop, we relabel the terminal nodes to be  $R_1, \dots, R_J$  and compute  $\bar{y}_{R_j}$  (means within each region) to serve as  $\hat{y}$  values.

But when do we stop?

# Stopping criterion

Once we stop, we relabel the terminal nodes to be  $R_1, \dots, R_J$  and compute  $\bar{y}_{R_j}$  (means within each region) to serve as  $\hat{y}$  values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum

# Stopping criterion

Once we stop, we relabel the terminal nodes to be  $R_1, \dots, R_J$  and compute  $\bar{y}_{R_j}$  (means within each region) to serve as  $\hat{y}$  values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum

# Stopping criterion

Once we stop, we relabel the terminal nodes to be  $R_1, \dots, R_J$  and compute  $\bar{y}_{R_j}$  (means within each region) to serve as  $\hat{y}$  values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum
- grow until no further splits can reduce RSS by some amount

# Stopping criterion

Once we stop, we relabel the terminal nodes to be  $R_1, \dots, R_J$  and compute  $\bar{y}_{R_j}$  (means within each region) to serve as  $\hat{y}$  values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum
- grow until no further splits can reduce RSS by some amount

# Stopping criterion

Once we stop, we relabel the terminal nodes to be  $R_1, \dots, R_J$  and compute  $\bar{y}_{R_j}$  (means within each region) to serve as  $\hat{y}$  values.

But when do we stop?

Some possibilities:

- number of observations in a node has reached a minimum
- depth of tree has reached a maximum
- grow until no further splits can reduce RSS by some amount

Many options – resulting in tuning parameters that are hard to deal with.

# Tree pruning

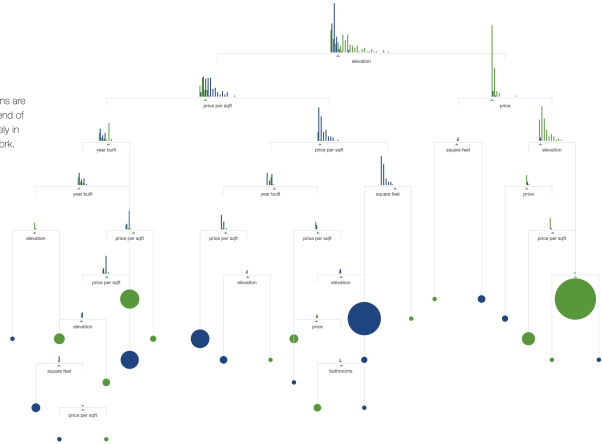
Another way to get around the overfitting problem is to grow a large tree and then **prune** it back.

Typically, pruning involves looking at subtrees of the fully-grown tree, and comparing how well the subtrees perform.



## Tree pruning

You could even continue to add branches until the tree's predictions are **100% accurate**, so that at the end of every branch, the homes are purely in San Francisco or purely in New York.



# Tree pruning

How do we prune?

# Tree pruning

How do we prune?

- cross validation

# Tree pruning

How do we prune?

- cross validation
- cost-complexity pruning

# Cost-complexity pruning

$$\begin{aligned}\text{Minimize:} \quad & \text{Loss}(T) + \lambda \{\# \text{ of nodes in } T\} \\ & = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \lambda |T|\end{aligned}$$

$\lambda$  is a tuning parameter that controls for the complexity of the model.

# Cost-complexity pruning

$$\begin{aligned}\text{Minimize: } & \text{Loss}(T) + \lambda \{\# \text{ of nodes in } T\} \\ & = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \lambda |T|\end{aligned}$$

$\lambda$  is a tuning parameter that controls for the complexity of the model.

- $\lambda = 0$  implies the full tree

# Cost-complexity pruning

$$\begin{aligned}\text{Minimize: } & \text{Loss}(T) + \lambda \{\# \text{ of nodes in } T\} \\ & = \sum_{m=1}^{|T|} \sum_{i \in R_m} (y_i - \hat{y}_{R_m})^2 + \lambda |T|\end{aligned}$$

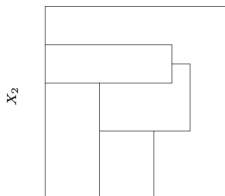
$\lambda$  is a tuning parameter that controls for the complexity of the model.

- $\lambda = 0$  implies the full tree
- Larger  $\lambda$  implies higher penalty for complexity of model

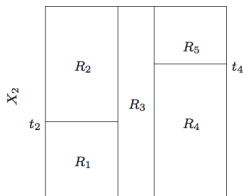
# Tree pruning

- 1 Grow a big tree on a *training set*.
- 2 Obtain a nested set of subtrees  $T_L \subset \dots \subset T_2 \subset T_1 \subset T_0$  corresponding to a sequence of  $\lambda$  values.
- 3 Use (leave-one-out or  $k$ -fold) cross-validation to identify the subtree/ $\lambda$  that does best.

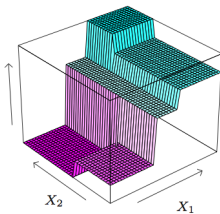
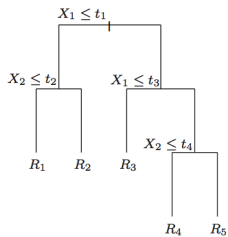




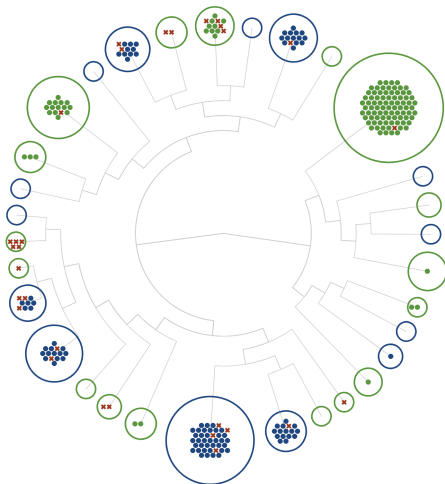
$X_1$



$X_1$



# Beautiful demo <http://www.r2d3.us/>



# Trees vs. other methods

Decision trees are similar in spirit to  $k$ -nearest neighbors.

# Trees vs. other methods

Decision trees are similar in spirit to  $k$ -nearest neighbors.

- Both produce simple predictions (averages/maximally occurring) based on “neighborhoods” in the predictor space.

# Trees vs. other methods

Decision trees are similar in spirit to  $k$ -nearest neighbors.

- Both produce simple predictions (averages/maximally occurring) based on “neighborhoods” in the predictor space.
- Neighborhoods chosen very differently

# Trees vs. other methods

Recall that linear regression fits models of the form

$$f(X) = \beta_0 + \sum_{j=1}^p X_j \beta_j$$

# Trees vs. other methods

Recall that linear regression fits models of the form

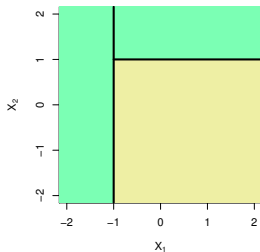
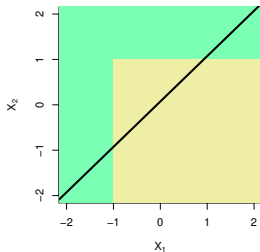
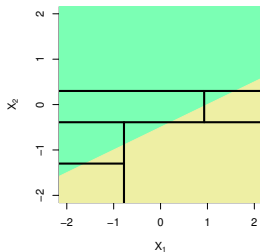
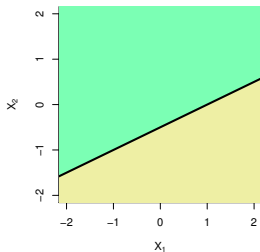
$$f(X) = \beta_0 + \sum_{j=1}^p X_j \beta_j$$

Regression trees are like fitting linear regression models with a bunch of indicators

$$f(X) = \sum_{j=1}^J \beta_j \mathbb{1} \{X \in R_j\}$$

# Trees vs. other methods

Are trees always better than linear methods?





# Summary so far

- Trees give interpretable, nonlinear prediction rules
- Deep trees have low bias, high variance
- Shallow trees have high bias, low variance
- Deep trees are pruned back using cross-validation to find best bias/variance tradeoff.

# Random Forests

- Grow many trees and average their predictions
- Trees are grown deep, to have low bias, but high variance
- To “decorrelate” the predictors, each tree is
  - ▶ grown on a bootstrap sample of the data
  - ▶ grown with random subsets of the predictors at each split
- Tree growing can be done in parallel

# Leo Breiman—"Keep it simple"



# Random Forests Algorithm

- ① For  $b = 1$  to  $B$ :
  - (a) Draw a bootstrap sample  $Z^*$  of size  $n$  from the training data
  - (b) Grow a random-forest tree  $T_b$  to the bootstrapped data, recursively repeating following steps, until minimum node size reached:
    - i. Select  $m$  variables at random from the  $p$  variables
    - ii. Pick the best variable/split-point among the  $m$
    - iii. Split the node into two children nodes
- ② Output the ensemble of trees  $\{T_b\}_{b=1}^B$ .

# Random Forests Algorithm

To make a prediction at a new point  $x$ :

*Regression:* Average  $\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^B T_b(x)$

*Classification:* Majority vote of the individual trees

# Out of bag (OOB) prediction

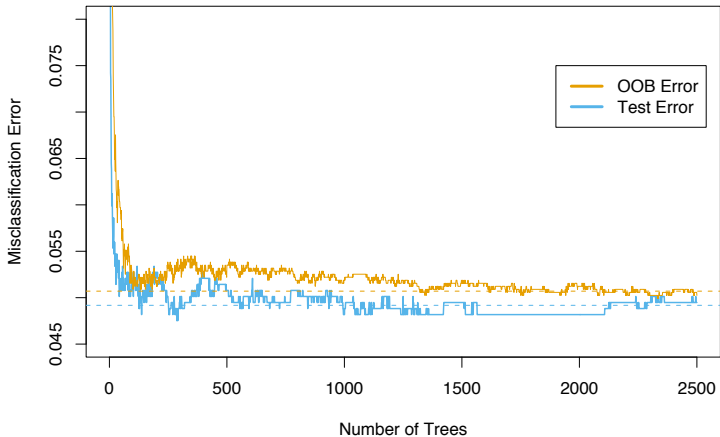
Each tree is grown on a random sample containing about  $\frac{2}{3}$  of the original data. The remaining  $\frac{1}{3}$  can be used as validation data

- For each observation  $z_i = (x_i, y_i)$ , construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which  $z_i$  did not appear
- Thus, cross-validation can be performed “along the way”

---

Chance a sample  $x_i$  does not appear in a bootstrap sample is  $\left(1 - \frac{1}{n}\right)^n \longrightarrow \frac{1}{e} \approx 0.37$

# Out of bag (OOB) prediction



# Performance on email spam task

Single tree: 8.7%

Random forests: 5.1%

(standard error of the estimates is  $\approx 0.6\%$ )



# Variable importance

Random forests improve upon the predictive ability of trees, but kills off the interpretability of the model.

# Variable importance

Random forests improve upon the predictive ability of trees, but kills off the interpretability of the model.

A good tool for interpreting a forest the **variable importance measure**.

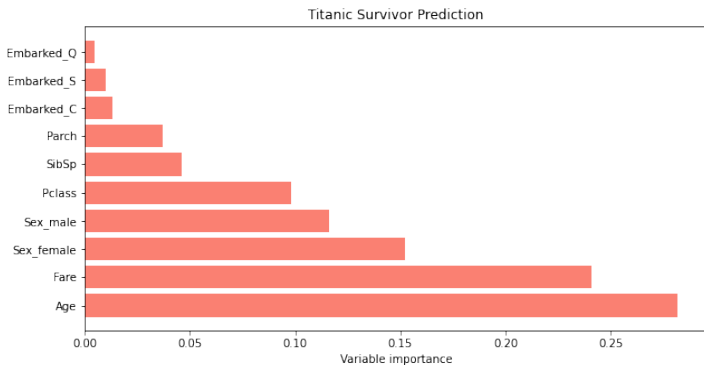
# Variable importance

Random forests improve upon the predictive ability of trees, but kills off the interpretability of the model.

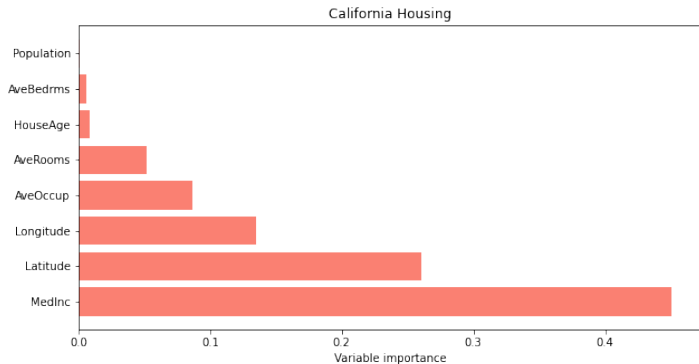
A good tool for interpreting a forest the **variable importance measure**.

Variable importance can be measured by the amount that the RSS (or other loss) is reduced due to splits over a given predictor, averaged over all trees in the forest

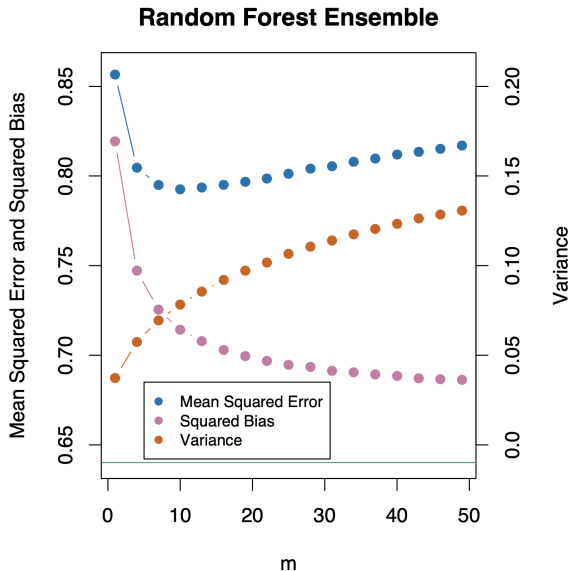
# Variable importance



# Variable importance



# Random forest MSE



# Let's go to the notebook

```
In [1]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```
In [2]: titanic_train = pd.read_csv('https://raw.githubusercontent.com/minsuk-heo/kaggle-titanic/master/input/train.csv')
titanic_test = pd.read_csv('https://raw.githubusercontent.com/minsuk-heo/kaggle-titanic/master/input/test.csv')
titanic_train
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cummings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows x 12 columns

# Summary

- Trees give interpretable, nonlinear prediction rules
- Deep trees have low bias, high variance
- Random forests are a way of combining trees
- Want: Different trees should capture different aspects of the data
- How: Grow each tree on a random (bootstrap) sample of the data and choosing from a random set of questions at each split