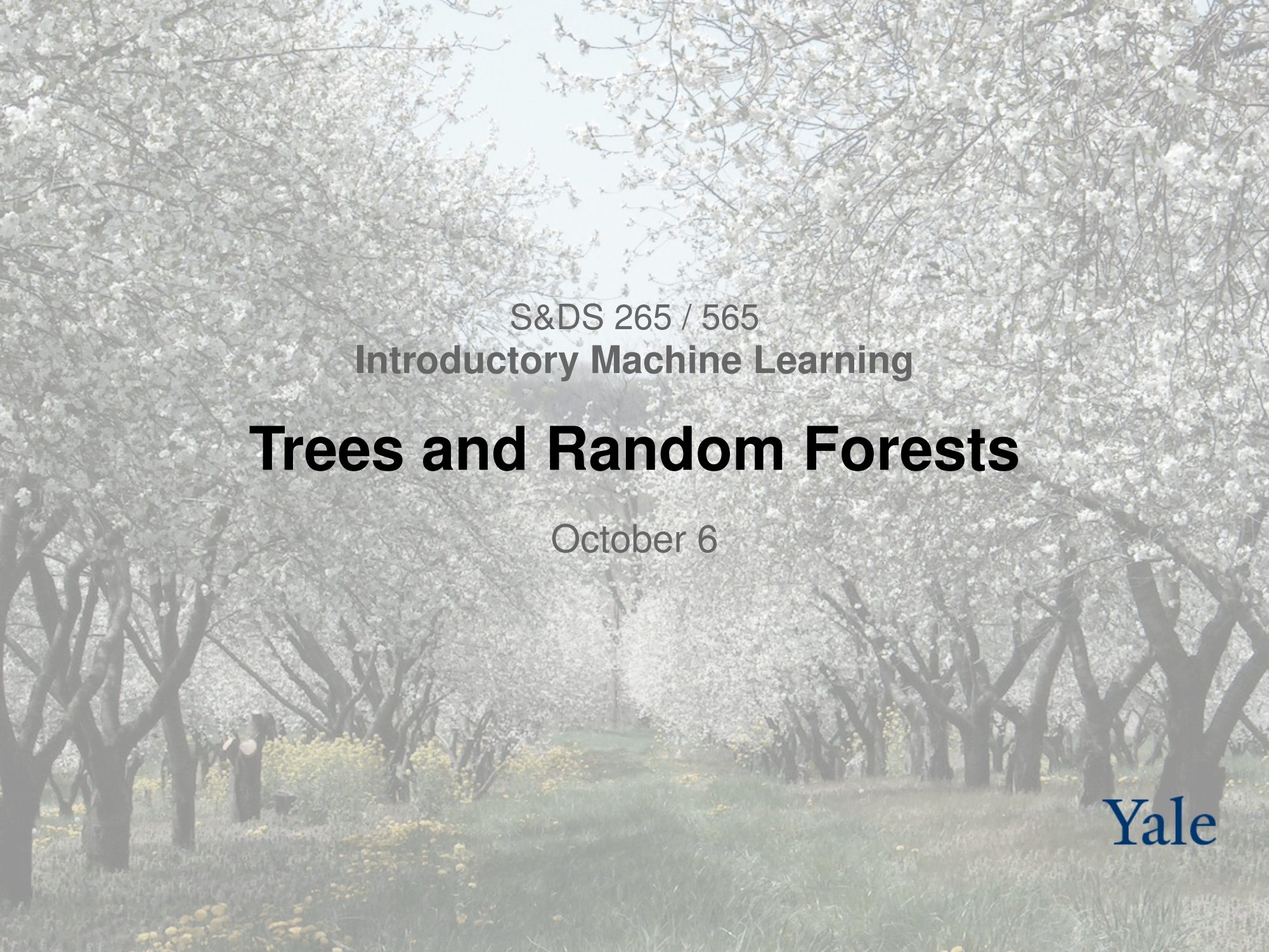


S&DS 265 / 565

Introductory Machine Learning

Trees and Random Forests

October 6

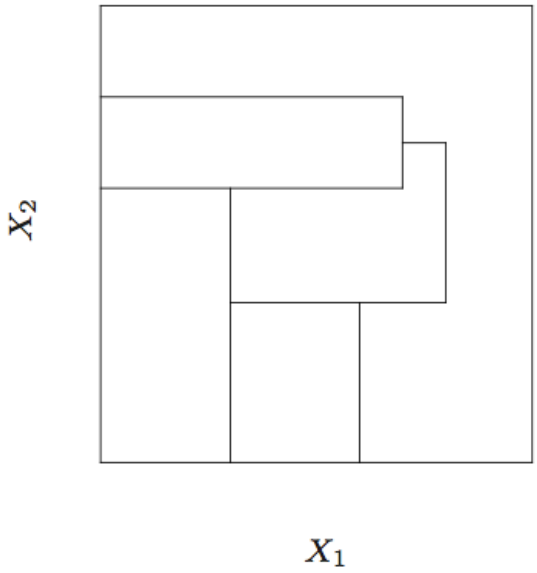
Yale

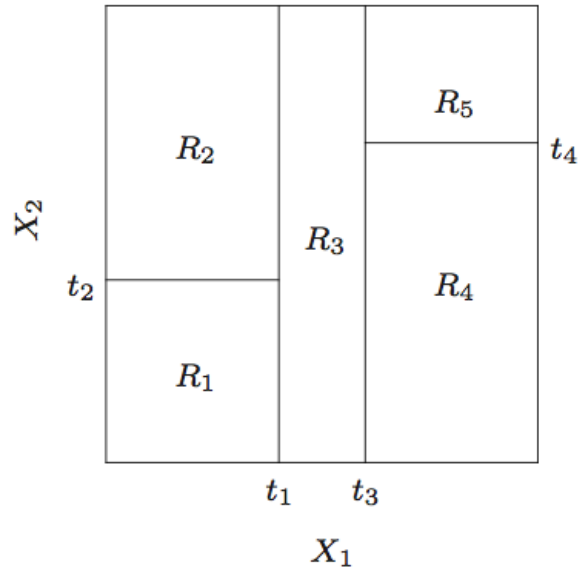
Reminders

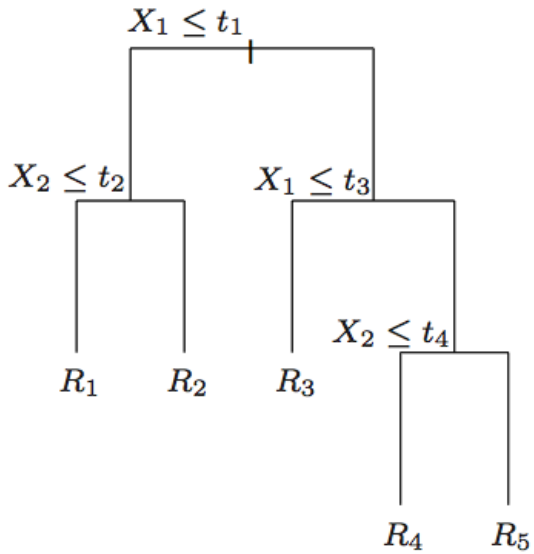
- Assn 2 out; due week from today
- Quiz 3 available 10:30am today (48 hours, 20 minutes)
- Midterm in class on October 18
- Practice midterms have been posted
- Review sessions TBA
- Questions?

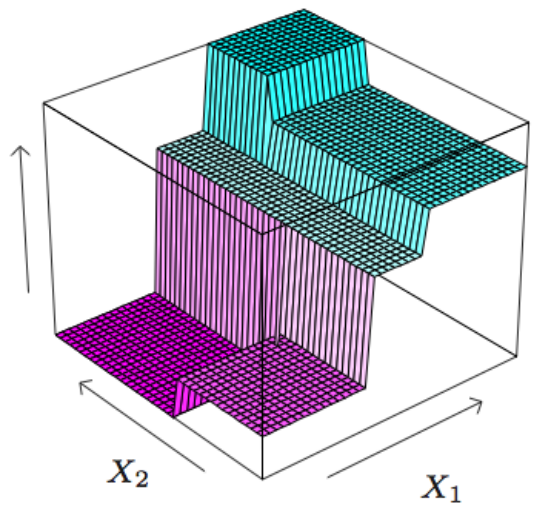
Last time: trees

- Give interpretable decision rules
- Deep trees have low bias, high variance
- Trees can be grown greedily to be full, then pruned back
- Predictive power is ... meh

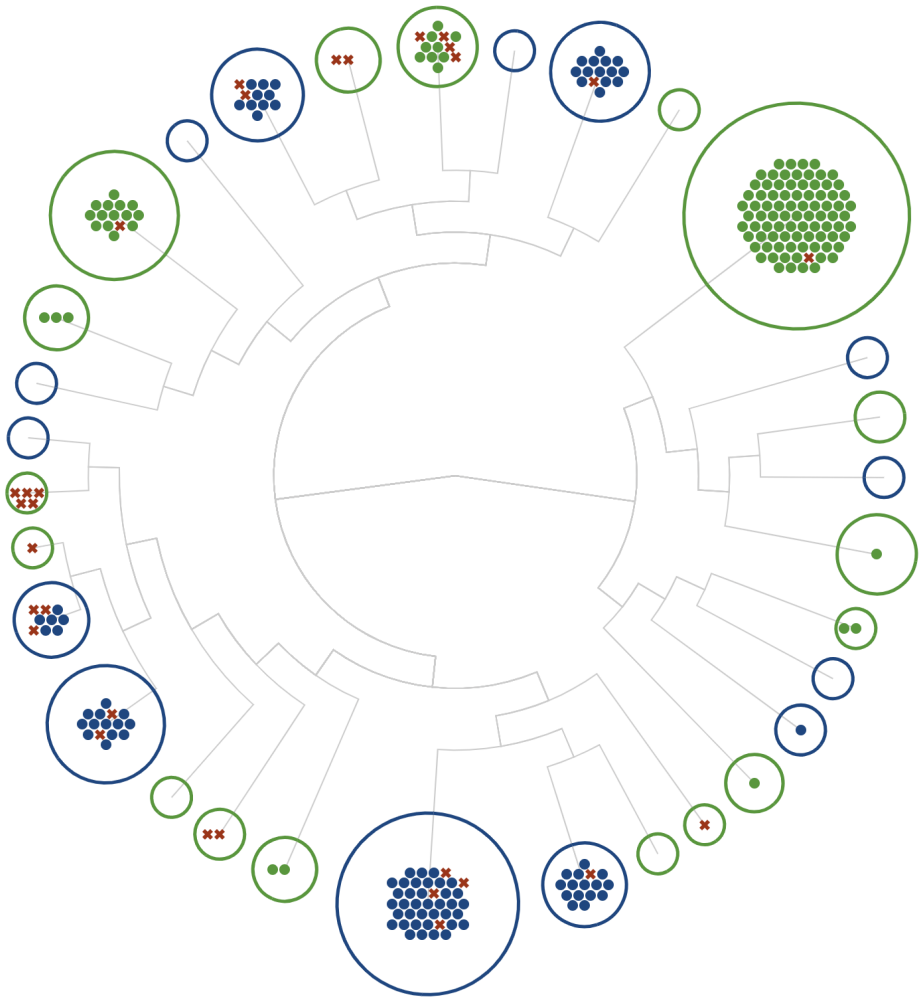

X_2
X_1


R_1
R_2
R_3
R_4
R_5
t_1
t_2
t_3
t_4
X_2
X_1


$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
R_1
R_2
R_3
R_4
R_5


X_2
X_1

Beautiful demo http://www.r2d3.us/visual-intro-to-machine-learning-part-2



Classification Trees

- $\widehat{y}_i$ for all $i \in R_j$ is *most commonly occurring class* of training observations in R_j.

$$\widehat{y}_{R_j} = \arg\max_k \widehat{p}_{jk},$$

  where $\widehat{p}_{jk}$ is the proportion of training observations with label k in rectangle (leaf) R_j.
- No longer want to minimize RSS, but instead minimize...
  - classification error rate

$$E = \sum_{j=1}^{J} |R_j|(1 - \max_k(\widehat{p}_{jk}))$$

Classification Trees

- $\widehat{y}_i$ for all $i \in R_j$ is *most commonly occurring class* of training observations in R_j.

$$\widehat{y}_{R_j} = \arg\max_k \widehat{p}_{jk},$$

  where $\widehat{p}_{jk}$ is the proportion of training observations with label k in rectangle (leaf) R_j.
- No longer want to minimize RSS, but instead minimize...
  - classification error rate

$$E = \sum_{j=1}^{J} |R_j|(1 - \max_k(\widehat{p}_{jk}))$$

  - Gini index

$$G = \sum_{j=1}^{J} |R_j| \sum_{k=1}^{K} \widehat{p}_{jk}(1 - \widehat{p}_{jk})$$

    Encourages higher **node purity**.

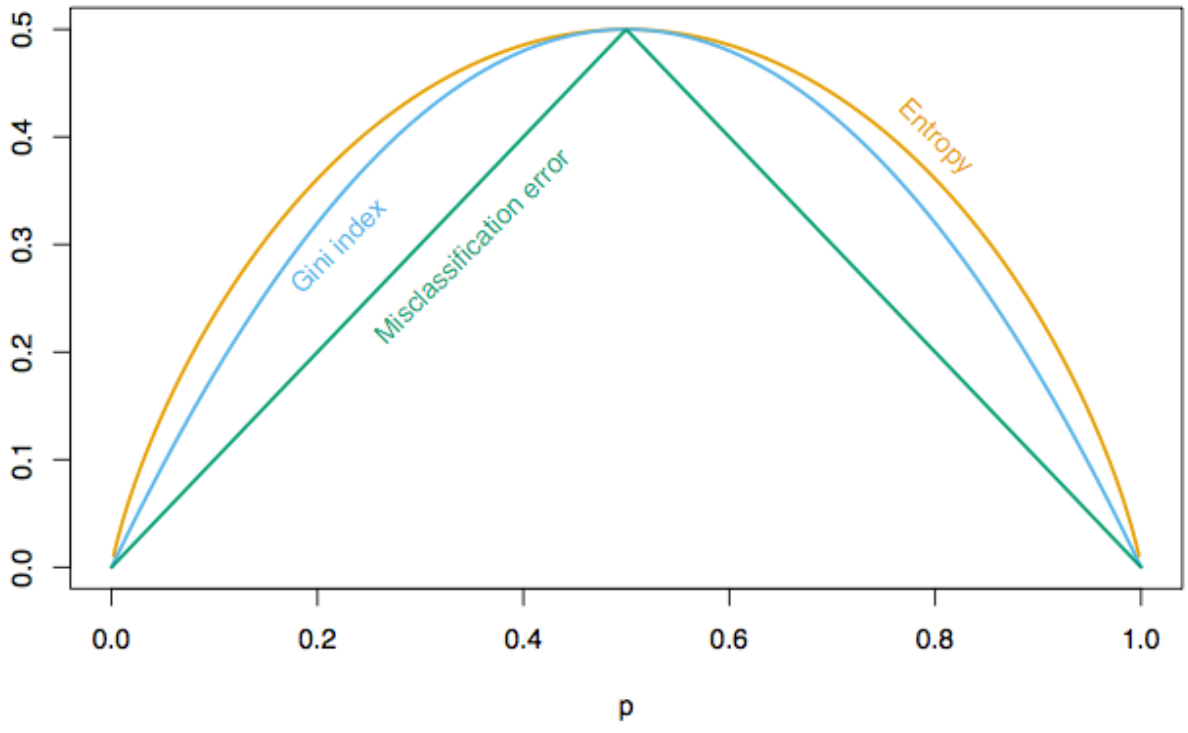
Impurity measures

Define node proportion of class k

$$\begin{aligned} \widehat{p}_{mk} &= \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k) \\ k(m) &= \arg\max_k \widehat{p}_{mk} \end{aligned}$$

- Misclassification error: $1 - \widehat{p}_{mk(m)}$
- Gini index: $\sum_{k=1}^{K} \widehat{p}_{mk}(1 - \widehat{p}_{mk})$
- Entropy: $-\sum_{k=1}^{K} \widehat{p}_{mk} \log \widehat{p}_{mk}$

Impurity measures



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

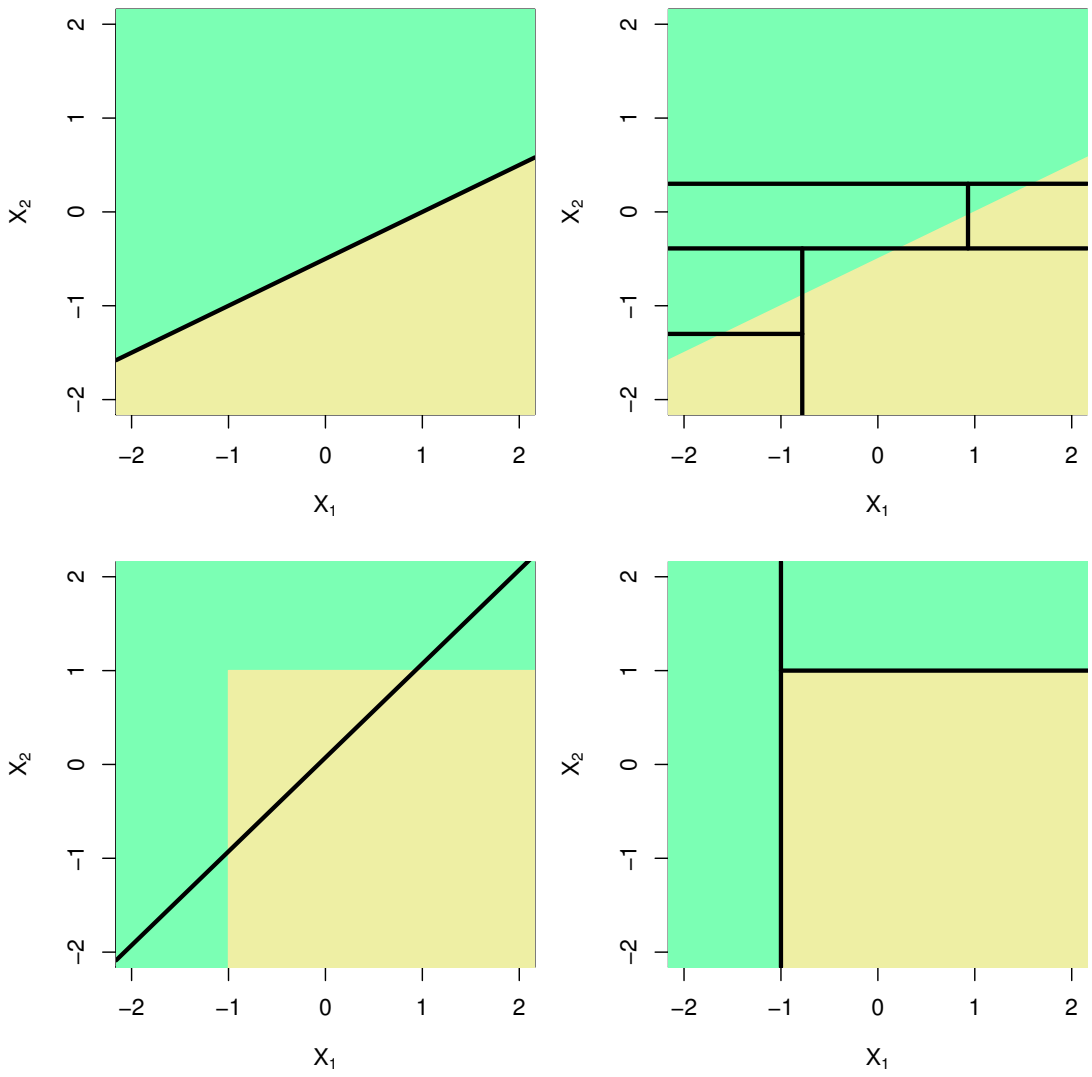
$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j$$

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

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

Trees vs. other methods

Are trees always better than linear methods?



Summary

- trees are intuitive
- prediction rules easy to explain, interpret
- trees are sensitive to underlying data
- trees produce non-smooth prediction surfaces
- prediction accuracy can be so-so

Ensemble methods

Ensemble methods pool together multiple models to arrive at more reliable predictions.

Ensemble methods

Ensemble methods pool together multiple models to arrive at more reliable predictions.

- bagging
- random forests
- boosting

Bootstrap samples

- We have n data points
- Randomly sample exactly n of them *with replacement*
- Contains about $\frac{2}{3}$ of the data, with duplicates

Chance a particular example does not appear in sample:

$$\left(1 - \frac{1}{n}\right)^n \longrightarrow \frac{1}{e} \approx 0.37$$

https://inferentialthinking.com/chapters/13/2/Bootstrap.html

Bagging

Regression trees: If we had multiple training sets, could grow multiple trees, then take an average.

Bagging

Regression trees: If we had multiple training sets, could grow multiple trees, then take an average.

Create B bootstrap samples, grow tree (without pruning) using each $\widehat{f}^{*1}, \widehat{f}^{*2}, \ldots \widehat{f}^{*B}$. For prediction at x, we take an average:

$$\widehat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}^{*b}(x)$$

Bagging

Regression trees: If we had multiple training sets, could grow multiple trees, then take an average.

Create B bootstrap samples, grow tree (without pruning) using each $\widehat{f}^{*1}, \widehat{f}^{*2}, \ldots \widehat{f}^{*B}$. For prediction at x, we take an average:

$$\widehat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}^{*b}(x)$$

Classification trees: For prediction at x, $\widehat{f}_{bag}(x)$ is decided by *majority vote*.

Leo Brieman – “Keep it Simple”

Out-of-Bag error estimation

Each bagged tree uses about 2/3 of all observations (with repeats).

The remaining data—*out-of-bag* (OOB) observations—can be put to good use.

Out-of-Bag error estimation

	Bagging iteration					OOB Est.
Obs	1	2	3	...	B	
1	OOB	train	train	...	train	$\widehat{y}_1$
2	train	OOB	train	...	train	$\widehat{y}_2$
3	train	train	OOB	...	train	$\widehat{y}_3$
4	OOB	train	train	...	OOB	$\widehat{y}_4$
...	...	...	...	...	...	...
n	train	train	OOB	...	train	$\widehat{y}_n$

Out-of-Bag error estimation

	Bagging iteration					OOB Est.
Obs	1	2	3	...	B	
1	OOB	train	train	...	train	$\widehat{y}_1$
2	train	OOB	train	...	train	$\widehat{y}_2$
3	train	train	OOB	...	train	$\widehat{y}_3$
4	OOB	train	train	...	OOB	$\widehat{y}_4$
...	...	...	...	...	...	...
n	train	train	OOB	...	train	$\widehat{y}_n$

- For each bagged tree, we can make predictions for the OOB observations.
- At the end, we can aggregate over all predictions for the i-th observation to arrive at a OOB prediction $\widehat{y}_i$.
- We can compute prediction error based on these OOB predictions $\widehat{y}_1, \ldots, \widehat{y}_n$.

Out-of-Bag error estimation

	Bagging iteration					OOB Est.
Obs	1	2	3	...	B	
1	OOB	train	train	...	train	$\widehat{y}_1$
2	train	OOB	train	...	train	$\widehat{y}_2$
3	train	train	OOB	...	train	$\widehat{y}_3$
4	OOB	train	train	...	OOB	$\widehat{y}_4$
...	...	...	...	...	...	...
n	train	train	OOB	...	train	$\widehat{y}_n$

- For each bagged tree, we can make predictions for the OOB observations.
- At the end, we can aggregate over all predictions for the i-th observation to arrive at a OOB prediction $\widehat{y}_i$.
- We can compute prediction error based on these OOB predictions $\widehat{y}_1, \ldots, \widehat{y}_n$.

Variable importance

While bagging improves upon the predictive ability of trees, it kills off the interpretability of the model.

Variable importance

While bagging improves upon the predictive ability of trees, it kills off the interpretability of the model.

A good tool for interpreting a bagged tree model (and other tree-based ensemble methods like forests) is the **variable importance measure**.

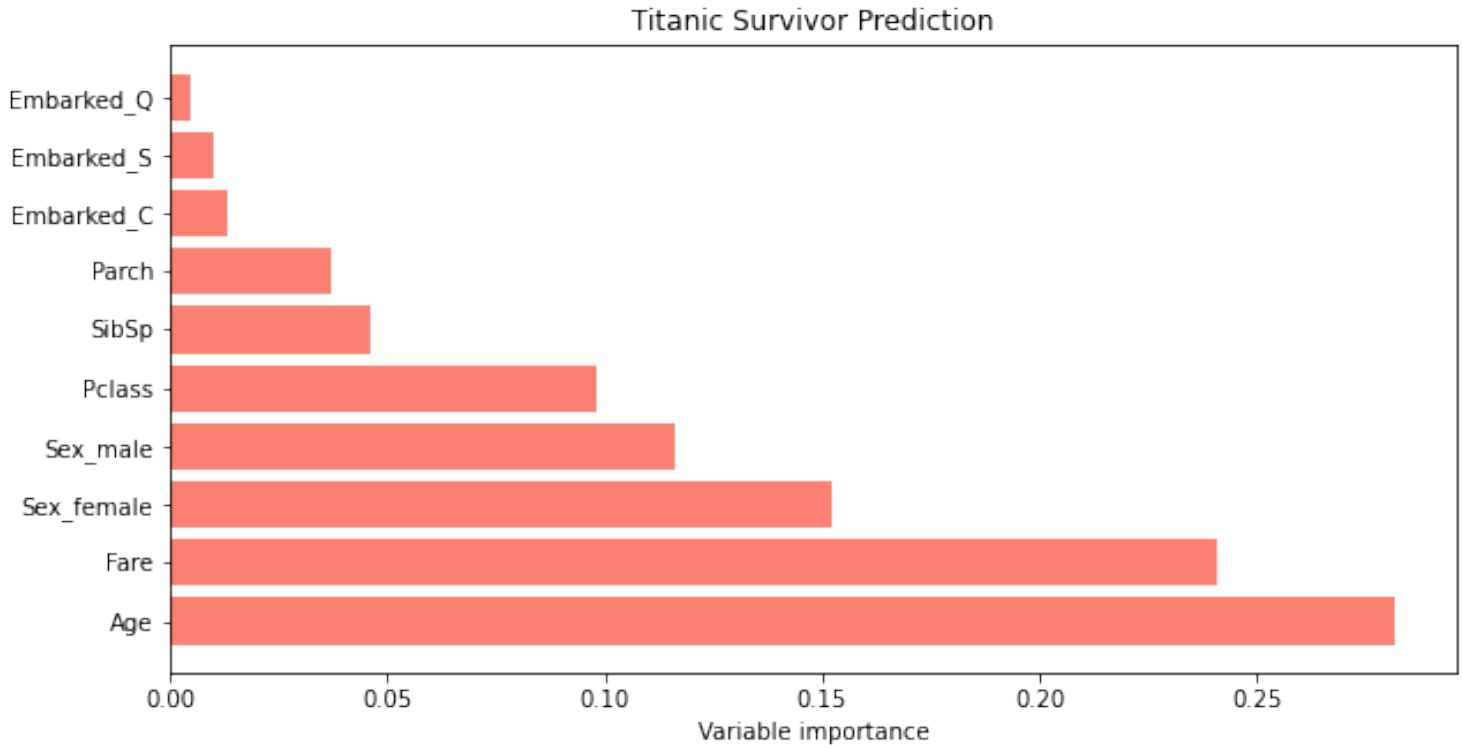
Variable importance

While bagging improves upon the predictive ability of trees, it kills off the interpretability of the model.

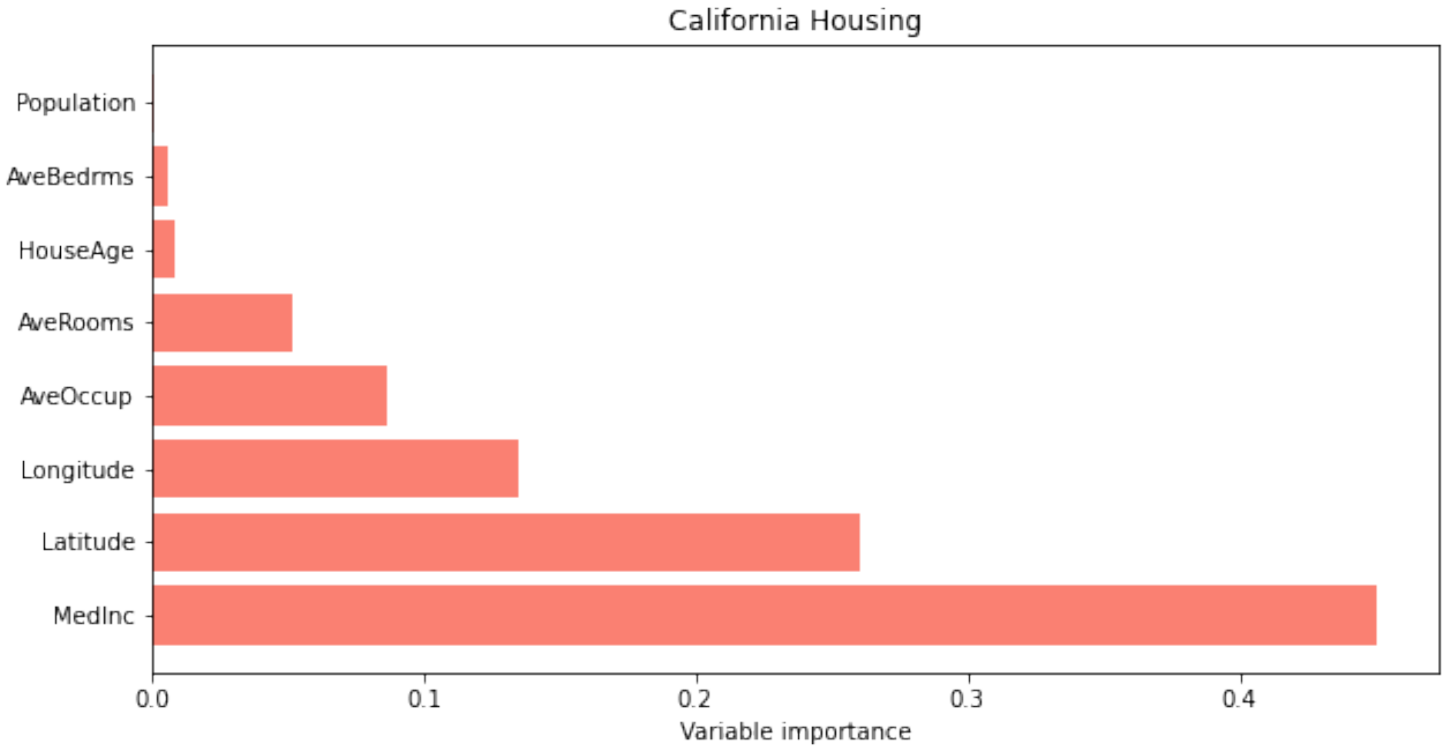
A good tool for interpreting a bagged tree model (and other tree-based ensemble methods like forests) is the **variable importance measure**.

Variable importance can be measured by the amount that the RSS (or Gini index) is reduced due to splits over a given predictor, averaged over all B trees.

Variable importance



Variable importance



Random Forests

Similar to bagging, but takes the averaging idea even further – averaging uncorrelated things decreases the error!

Random Forests

Similar to bagging, but takes the averaging idea even further – averaging uncorrelated things decreases the error!

Still use bootstrap samples, but only use m out of p predictors at each split, *chosen randomly*

Bagging and 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

Guess who?

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 fo trees $\{T_b\}_{b=1}^{B}$.

To make a prediction at a new point x:

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

Classification: Majority vote of the individual trees

Random forests–recommended parameters

- For classification, default value of m is $\lfloor \sqrt{p} \rfloor$ and the minimum node size is one.
- For regression, the default values of m is $\lfloor p/3 \rfloor$ and the minimum node size is five.

(you don't need to remember this)

MSRP: Manufacturer's Suggested Randomization Parameters

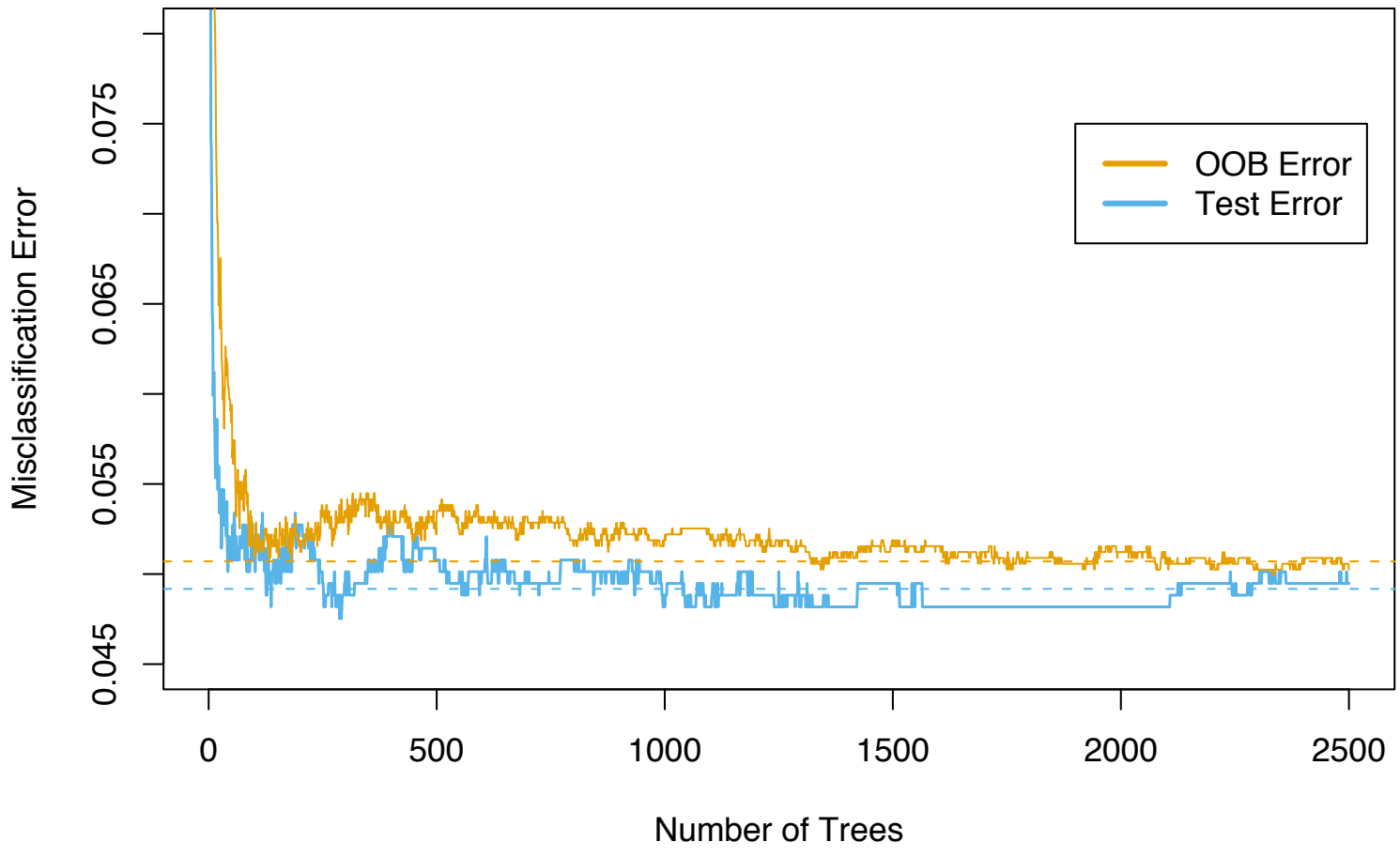
Out of bag (OOB) prediction

As before, can use out-of-bag (OOB) samples:

- 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



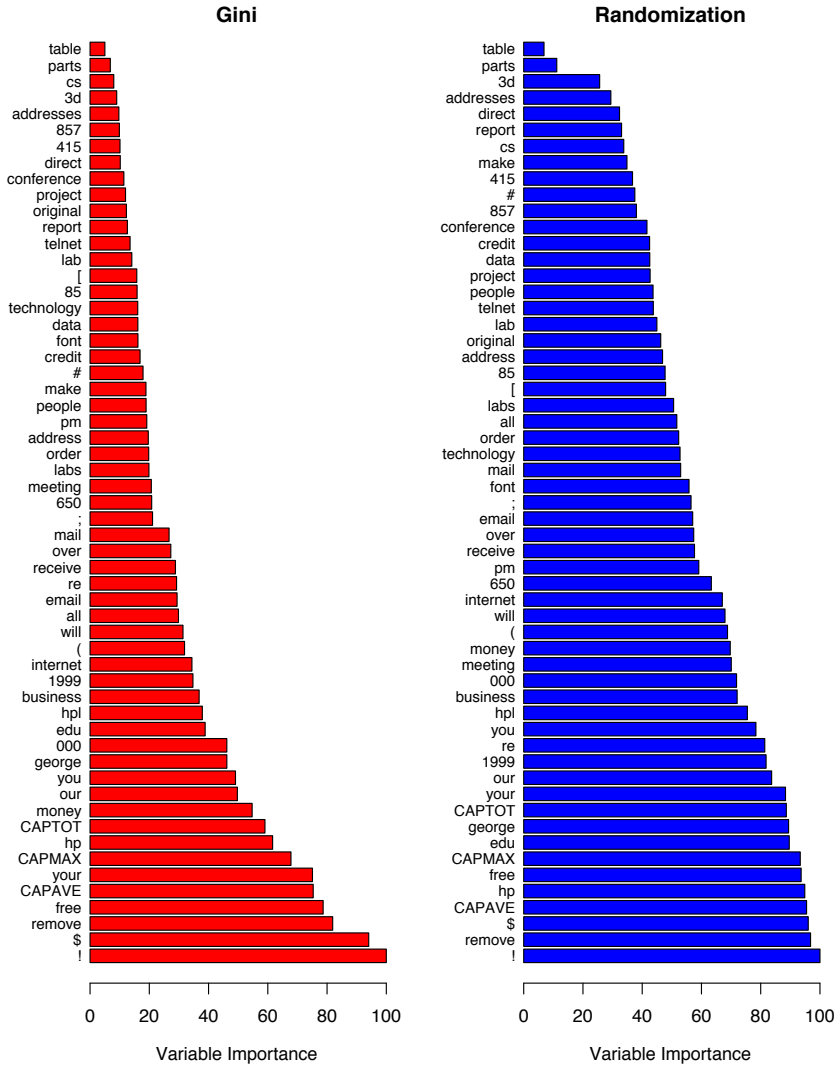
Performance on email spam task

Single tree: 8.7%

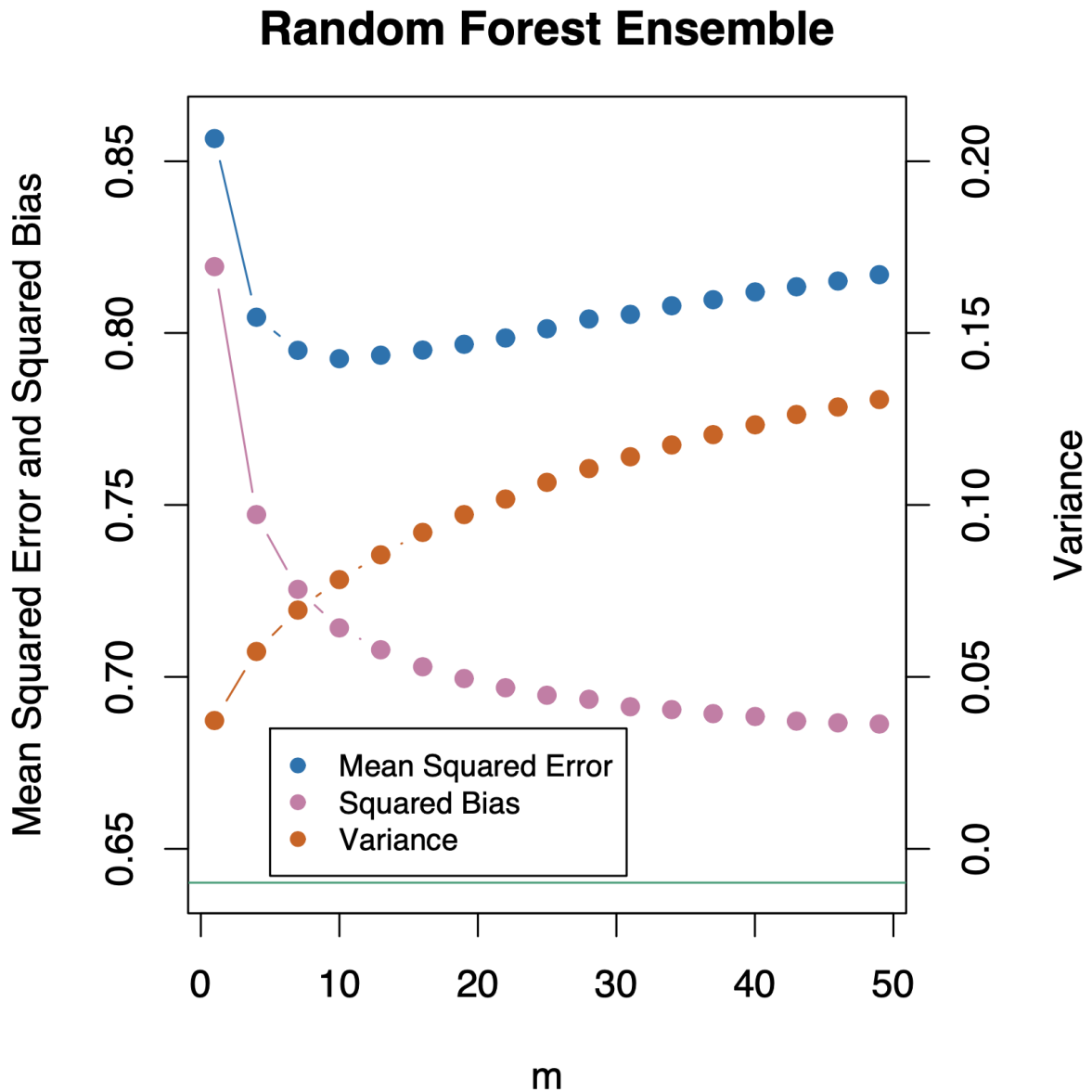
Random forests: 5.1%

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

Important features: Spam



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	Cumings, 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	Futrelle, 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 × 12 columns

What did we learn today?

- Bagging tries to reduce variance by averaging many trees
- Random forests decorrelate by random sampling of predictors
- Manage bias-variance tradeoff with randomization