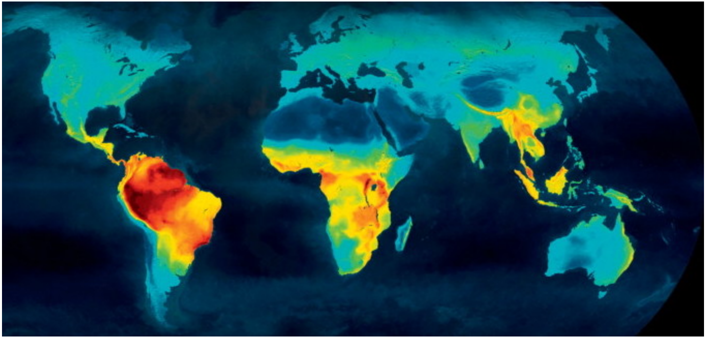
Ensemble Methods (Part I)
Bagging and Random Forests

Schwartz

August 1, 2017

Trees!

A recent study published in the Proceedings of the National Academy of Sciences estimates that there are somewhere between 40,000 to 50,000 unique tree species in the American neotropics and Indo-Pacific tropics, respectively, and that tropical Africa includes an additional approximately 10,000 species. This amount of tree diversity far outpaces tree diversity in temperate regions. For example, temperate forests in Europe have only 124 unique tree species and there are only approximately 1,000 tree species in North America. Another recent study published in Nature estimates that there are 3.04 trillion trees on earth – 422 per person – and that 45% of the trees are in the tropics. Tropical forests make up 2% of the earth's landmass. By area then, tropical jungles produce tree diversity at a rate of 5000:1 and tree density at a rate of 40:1 compared the rest of the earth's dry landmasses.



Objectives

- What is *Bagging*?

Objectives

- What is *Bagging*?
  - How do you do it?

Objectives

- What is *Bagging*?
  - How do you do it?
    - Why is it good?

Objectives

- What is *Bagging*?
  - How do you do it?
    - Why is it good?
- What are *Random Forests*?

Objectives

- What is *Bagging*?
  - How do you do it?
    - Why is it good?
- What are *Random Forests*?
  - How do you make them?

Objectives

- What is *Bagging*?
  - How do you do it?
    - Why is it good?
- What are *Random Forests*?
  - How do you make them?
    - Why might they be an improvement over bagging?

Objectives

- What is *Bagging*?
  - How do you do it?
    - Why is it good?
- What are *Random Forests*?
  - How do you make them?
    - Why might they be an improvement over bagging?
- What is "out of bag error" (OOB) relative to K-folds?

Objectives

- What is *Bagging*?
  - How do you do it?
    - Why is it good?
- What are *Random Forests*?
  - How do you make them?
    - Why might they be an improvement over bagging?
- What is "out of bag error" (OOB) relative to K-folds?
- How can one interpret variables in tree ensembles?

Single Predictors/Estimators

- A single tree is pretty great!

Single Predictors/Estimators

- A single tree is pretty great!
- Well, it's not that good, actually

Single Predictors/Estimators

- A single tree is pretty great!
- Well, it's not that good, actually
  What could be wrong with it?

Single Predictors/Estimators

- A single tree is pretty great!
- Well, it's not that good, actually

  What could be wrong with it?

    Biased?

    High variance?

Single Predictors/Estimators

- A single tree is pretty great!
- Well, it's not that good, actually

  What could be wrong with it?

  Biased?
  High variance?
  How so with trees?

Single Predictors/Estimators

- A single tree is pretty great!
- Well, it's not that good, actually

  What could be wrong with it?

    Biased?
    High variance?
    How so with trees?

  Do we think a prediction from a single tree is good?

Ensemble Challenge

Suppose you are trying to predict the election...

You have 5 expert opinions, each with a 70% chance of being right, and each expert pick is *independent* of the other expert picks.

How could you leverage expert picks to improve accuracy and how often would you be right?

Ensemble Challenge

Suppose you are trying to predict the election...

You have 5 expert opinions, each with a 70% chance of being right, and each expert pick is *independent* of the other expert picks.

How could you leverage expert picks to improve accuracy and how often would you be right?

How many independent predictors would you need to have a 99.9% of being correct?

Ensemble Challenge

Suppose you are trying to predict the election...

You have 5 expert opinions, each with a 70% chance of being right, and each expert pick is *independent* of the other expert picks.

How could you leverage expert picks to improve accuracy and how often would you be right?

How many independent predictors would you need to have a 99.9% of being correct?

`1 - stats.binom.cdf(26,n=53,p=.7)` $< .5$

$<$ `1 - stats.binom.cdf(27,n=55,p=.7)`

Averaging Multiple Predictors/Estimators

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be the j^{th} predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$
- If each $\hat{f}^{(j)}(\boldsymbol{x}_0)$ is an unbiased predictor of $f(\boldsymbol{x}_0)$ then

$$\mathrm{E}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = f(\boldsymbol{x}_0)$$

Averaging Multiple Predictors/Estimators

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be the j^{th} predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- If each $\hat{f}^{(j)}(\boldsymbol{x}_0)$ is an unbiased predictor of $f(\boldsymbol{x}_0)$ then

$$\mathrm{E}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = f(\boldsymbol{x}_0)$$

$\hat{f}^{(j)}(\boldsymbol{x}_0)$ can be an unbiased predictor by being highly complex

Averaging Multiple Predictors/Estimators

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be the j^{th} predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- If each $\hat{f}^{(j)}(\boldsymbol{x}_0)$ is an unbiased predictor of $f(\boldsymbol{x}_0)$ then

$$\mathrm{E}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = f(\boldsymbol{x}_0)$$

$\hat{f}^{(j)}(\boldsymbol{x}_0)$ can be an unbiased predictor by being highly complex

E.g., like an overfit tree!

Averaging Multiple Predictors/Estimators

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be the j^{th} predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- If each $\hat{f}^{(j)}(\boldsymbol{x}_0)$ is an unbiased predictor of $f(\boldsymbol{x}_0)$ then

$$\mathrm{E}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = f(\boldsymbol{x}_0)$$

$\hat{f}^{(j)}(\boldsymbol{x}_0)$ can be an unbiased predictor by being highly complex

E.g., like an overfit tree!

- And if $\mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \sigma^2$, then $\mathrm{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = ?$

Averaging Multiple Predictors/Estimators

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be the j^{th} predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- If each $\hat{f}^{(j)}(\boldsymbol{x}_0)$ is an unbiased predictor of $f(\boldsymbol{x}_0)$ then

$$\mathrm{E}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = f(\boldsymbol{x}_0)$$

$\hat{f}^{(j)}(\boldsymbol{x}_0)$ can be an unbiased predictor by being highly complex

E.g., like an overfit tree!

- And if $\mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \sigma^2$, then $\mathrm{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \frac{\sigma^2}{m}$?

Averaging Multiple Predictors/Estimators

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be the j^{th} predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- If each $\hat{f}^{(j)}(\boldsymbol{x}_0)$ is an unbiased predictor of $f(\boldsymbol{x}_0)$ then

$$\mathrm{E}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = f(\boldsymbol{x}_0)$$

$\hat{f}^{(j)}(\boldsymbol{x}_0)$ can be an unbiased predictor by being highly complex

E.g., like an overfit tree!

- And if $\mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \sigma^2$, then $\mathrm{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \frac{\sigma^2}{m}$?

So even if $\hat{f}^{(j)}$ is a high variance predictor

Averaging Multiple Predictors/Estimators

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be the j^{th} predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- If each $\hat{f}^{(j)}(\boldsymbol{x}_0)$ is an unbiased predictor of $f(\boldsymbol{x}_0)$ then

$$\mathrm{E}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = f(\boldsymbol{x}_0)$$

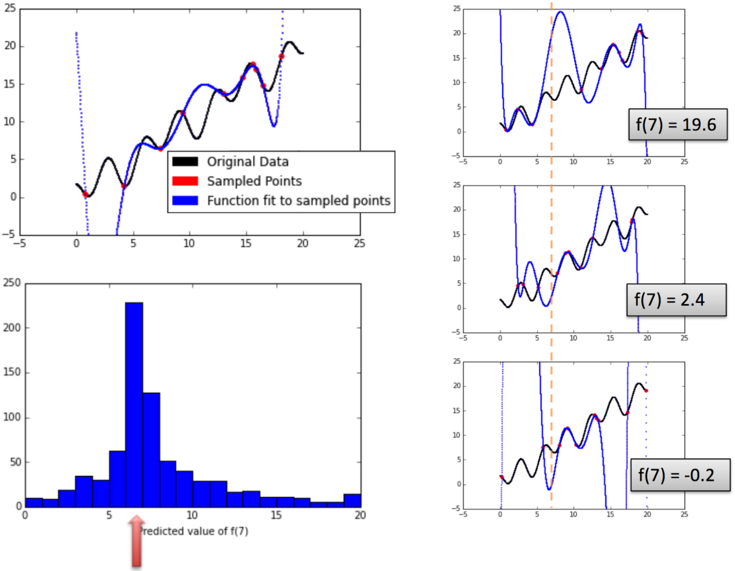
$\hat{f}^{(j)}(\boldsymbol{x}_0)$ can be an unbiased predictor by being highly complex

E.g., like an overfit tree!

- And if $\mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \sigma^2$, then $\mathrm{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \frac{\sigma^2}{m}$?

So even if $\hat{f}^{(j)}$ is a high variance predictor the variance of the *averaged* predictor $\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}$ decreases with m

What is happening?



Actual value: f(7) = 6.8

Bootstrapping Aggregation (Bagging)

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be a predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

Bootstrapping Aggregation (Bagging)

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be a predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- *Bagging* is simply predicting $f(\boldsymbol{x}_0)$ with $\displaystyle\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)$

  an average of predictors made from bootstraped samples

Bootstrapping Aggregation (Bagging)

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be a predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- *Bagging* is simply predicting $f(\boldsymbol{x}_0)$ with $\displaystyle\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)$

  an average of predictors made from bootstraped samples

- $\boldsymbol{x}^{(j)}$ and $\boldsymbol{Y}^{(j)}$ is a *bootstrapped* sample from $\boldsymbol{X}$ and $\boldsymbol{Y}$
- $\hat{f}^{(j)}$ is the model fit on that bootstrapped training set

Bootstrapping Aggregation (Bagging)

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be a predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- *Bagging* is simply predicting $f(\boldsymbol{x}_0)$ with $\dfrac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)$

  an average of predictors made from bootstraped samples
- $\boldsymbol{x}^{(j)}$ and $\boldsymbol{Y}^{(j)}$ is a *bootstrapped* sample from $\boldsymbol{X}$ and $\boldsymbol{Y}$
- $\hat{f}^{(j)}$ is the model fit on that bootstrapped training set

1. Bootrapped trees provides unbiased, high variance predictors*

Bootstrapping Aggregation (Bagging)

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be a predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- *Bagging* is simply predicting $f(\boldsymbol{x}_0)$ with $\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)$

  an average of predictors made from bootstraped samples

- $\boldsymbol{x}^{(j)}$ and $\boldsymbol{Y}^{(j)}$ is a *bootstrapped* sample from $\boldsymbol{X}$ and $\boldsymbol{Y}$
- $\hat{f}^{(j)}$ is the model fit on that bootstrapped training set

1. Bootrapped trees provides unbiased, high variance predictors*
2. Averaged estimators are lower variance than single predictors

Bootstrapping Aggregation (Bagging)

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be a predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- *Bagging* is simply predicting $f(\boldsymbol{x}_0)$ with $\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)$

  an average of predictors made from bootstraped samples

- $\boldsymbol{x}^{(j)}$ and $\boldsymbol{Y}^{(j)}$ is a *bootstrapped* sample from $\boldsymbol{X}$ and $\boldsymbol{Y}$
- $\hat{f}^{(j)}$ is the model fit on that bootstrapped training set

1. Bootrapped trees provides unbiased, high variance predictors*
2. Averaged estimators are lower variance than single predictors
3. Averaged predictors are still unbiased

Bootstrapping Aggregation (Bagging)

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be a predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$

- *Bagging* is simply predicting $f(\boldsymbol{x}_0)$ with $\displaystyle\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)$

  an average of predictors made from bootstraped samples

- $\boldsymbol{x}^{(j)}$ and $\boldsymbol{Y}^{(j)}$ is a *bootstrapped* sample from $\boldsymbol{X}$ and $\boldsymbol{Y}$
- $\hat{f}^{(j)}$ is the model fit on that bootstrapped training set

1. Bootrapped trees provides unbiased, high variance predictors*
2. Averaged estimators are lower variance than single predictors
3. Averaged predictors are still unbiased
4. Bootrapping explores the "data set" / "predictors" space: then uses the average of predictors we might see under sampling

Bootstrapping Aggregation (Bagging)

- Let $\hat{f}^{(j)}(\boldsymbol{x}_0)$ be a predictor of outcome $f(\boldsymbol{x}_0)$ constructed from a training set with features $\boldsymbol{x}^{(j)}$ and outcomes $\boldsymbol{Y}^{(j)}$
- *Bagging* is simply predicting $f(\boldsymbol{x}_0)$ with $\displaystyle\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)$

  $\underline{\text{an average of predictors made from bootstraped samples}}$
- $\boldsymbol{x}^{(j)}$ and $\boldsymbol{Y}^{(j)}$ is a *bootstrapped* sample from $\boldsymbol{X}$ and $\boldsymbol{Y}$
- $\hat{f}^{(j)}$ is the model fit on that bootstrapped training set

1. Bootrapped trees provides unbiased, high variance predictors*
2. Averaged estimators are lower variance than single predictors
3. Averaged predictors are still unbiased
4. Bootrapping explores the "data set" / "predictors" space: then uses the average of predictors we might see under sampling

* *Bagging is a general ensemble procedure often used with trees*

“Out of Bag” (OOB) testing error

- Let $\boldsymbol{x}_0^{(j)}$ and $\boldsymbol{Y}_0^{(j)}$ be the features and outcomes not in $\boldsymbol{x}^{(j)}$

"Out of Bag" (OOB) testing error

- Let $\boldsymbol{x}_0^{(j)}$ and $\boldsymbol{Y}_0^{(j)}$ be the features and outcomes not in $\boldsymbol{x}^{(j)}$
- OOB prediction error is calculated from

$$\boldsymbol{Y}_0^{(j)} \;\text{ vs }\; \hat{f}^{(j)}\left(\boldsymbol{x}_0^{(j)}\right)$$

i.e., actual $\boldsymbol{Y}_0^{(j)}$ versus prediction $\hat{\boldsymbol{Y}}_0^{(j)}$ from $\hat{f}^{(j)}\left(\boldsymbol{x}_0^{(j)}\right)$

“Out of Bag” (OOB) testing error

- Let $\boldsymbol{x}_0^{(j)}$ and $\boldsymbol{Y}_0^{(j)}$ be the features and outcomes not in $\boldsymbol{x}^{(j)}$
- OOB prediction error is calculated from

$$\boldsymbol{Y}_0^{(j)} \ \text{ vs } \ \hat{f}^{(j)}\left(\boldsymbol{x}_0^{(j)}\right)$$

i.e., actual $\boldsymbol{Y}_0^{(j)}$ versus prediction $\hat{\boldsymbol{Y}}_0^{(j)}$ from $\hat{f}^{(j)}\left(\boldsymbol{x}_0^{(j)}\right)$

(j)	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	$\cdots$
(1)	■	■	■	□	□	■	■	□	■	
(2)	□	□	■	■	■	□	■	■	■	
(3)	■	□	■	■	■	□	■	□	■	
$\vdots$										

Bootstrapped samples of size n leave out $\sim \frac{1}{3}$ of the data

"Out of Bag" (OOB) testing error

- Let $\boldsymbol{x}_0^{(j)}$ and $\boldsymbol{Y}_0^{(j)}$ be the features and outcomes not in $\boldsymbol{x}^{(j)}$
- OOB prediction error is calculated from

$$\boldsymbol{Y}_0^{(j)} \ \text{ vs } \ \hat{f}^{(j)}\left(\boldsymbol{x}_0^{(j)}\right)$$

i.e., actual $\boldsymbol{Y}_0^{(j)}$ versus prediction $\hat{\boldsymbol{Y}}_0^{(j)}$ from $\hat{f}^{(j)}\left(\boldsymbol{x}_0^{(j)}\right)$

(j)	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	$\cdots$
(1)	■	■	■	□	□	■	■	□	■	
(2)	□	□	■	■	■	□	■	■	■	
(3)	■	□	■	■	■	□	■	□	■	
$\vdots$										

Bootstrapped samples of size n leave out $\sim \frac{1}{3}$ of the data

- When the number of bootstrapped samples $j = 1, \cdots m$ is large, this closely approximates LOO test error estimation.

“Out of Bag” (OOB) testing error

- Let $\boldsymbol{x}_0^{(j)}$ and $\boldsymbol{Y}_0^{(j)}$ be the features and outcomes not in $\boldsymbol{x}^{(j)}$
- OOB prediction error is calculated from

$$\boldsymbol{Y}_0^{(j)} \text{ vs } \hat{f}^{(j)}\left(\boldsymbol{x}_0^{(j)}\right)$$

i.e., actual $\boldsymbol{Y}_0^{(j)}$ versus prediction $\hat{\boldsymbol{Y}}_0^{(j)}$ from $\hat{f}^{(j)}\left(\boldsymbol{x}_0^{(j)}\right)$

(j)	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7	Y_8	Y_9	$\cdots$
(1)	■	■	■	□	□	■	■	□	■	
(2)	□	□	■	■	■	□	■	■	■	
(3)	■	□	■	■	■	□	■	□	■	
$\vdots$										

Bootstrapped samples of size n leave out $\sim \frac{1}{3}$ of the data

- When the number of bootstrapped samples $j = 1, \cdots m$ is large, this closely approximates LOO test error estimation.
- Parameter tuning can be done “for free” when model fitting

But wait...

$$\mathrm{Var}\left[\sum_{j=1}^{m} \frac{1}{m} \hat{f}^{(j)}(\boldsymbol{x}_0)\right] \neq \frac{\sigma^2}{m}$$

But wait...

$$\mathrm{Var}\left[\sum_{j=1}^{m}\frac{1}{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] \neq \frac{\sigma^2}{m}$$

- Why?

But wait...

$$\mathrm{Var}\left[\sum_{j=1}^{m} \frac{1}{m} \hat{f}^{(j)}(\boldsymbol{x}_0)\right] \neq \frac{\sigma^2}{m}$$

- Why?

$$\mathrm{Var}[X+Y] = \mathrm{Var}[X] + \mathrm{Var}[Y] + 2\mathrm{Cov}[X, Y]$$

I.e.,

$$\begin{aligned}\mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0) + \hat{f}^{(k)}(\boldsymbol{x}_0)\right] = \mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0)\right] + \mathrm{Var}\left[\hat{f}^{(k)}(\boldsymbol{x}_0)\right] \\ + 2\mathrm{Cov}\left[\hat{f}^{(j)}(\boldsymbol{x}_0), \hat{f}^{(k)}(\boldsymbol{x}_0)\right]\end{aligned}$$

But wait...

$$\mathrm{Var}\left[\sum_{j=1}^{m}\frac{1}{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] \neq \frac{\sigma^2}{m}$$

- Why?

$$\mathrm{Var}\left[X+Y\right] = \mathrm{Var}\left[X\right] + \mathrm{Var}\left[Y\right] + 2\mathrm{Cov}\left[X,Y\right]$$

I.e.,

$$\mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0)+\hat{f}^{(k)}(\boldsymbol{x}_0)\right] = \mathrm{Var}\left[\hat{f}^{(j)}(\boldsymbol{x}_0)\right] + \mathrm{Var}\left[\hat{f}^{(k)}(\boldsymbol{x}_0)\right] + 2\mathrm{Cov}\left[\hat{f}^{(j)}(\boldsymbol{x}_0), \hat{f}^{(k)}(\boldsymbol{x}_0)\right]$$

$\hat{f}^{(j)}$ and $\hat{f}^{(k)}$ are correlated: they're predicting the same thing from bootstrapped samples $\boldsymbol{x}^{(j)}/\boldsymbol{Y}^{(j)}$ & $\boldsymbol{x}^{(k)}/\boldsymbol{Y}^{(k)}$ from $\boldsymbol{x}/\boldsymbol{Y}$!

How good is bagging?

- If $\mathrm{Cor}\left[\hat{f}^{(j)}, \hat{f}^{(k)}\right] = \rho$ for all j and k

How good is bagging?

- If $\operatorname{Cor}\left[\hat{f}^{(j)}, \hat{f}^{(k)}\right] = \rho$ for all j and k

  then

$$\operatorname{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \rho\sigma^2 + (1-\rho)\frac{\sigma^2}{m}$$

How good is bagging?

- If $\mathrm{Cor}\left[\hat{f}^{(j)}, \hat{f}^{(k)}\right] = \rho$ for all j and k

  then
  $$\mathrm{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \rho\sigma^2 + (1-\rho)\frac{\sigma^2}{m}$$
- Only "uncorrelated parts" of $\hat{f}^{(j)}$ and $\hat{f}^{(k)}$ get "CLT effect"

How good is bagging?

- If $\text{Cor}\left[\hat{f}^{(j)}, \hat{f}^{(k)}\right] = \rho$ for all j and k

  then
  $$\text{Var}\left[\frac{1}{m}\sum_{j=1}^{m}\hat{f}^{(j)}(\boldsymbol{x}_0)\right] = \rho\sigma^2 + (1-\rho)\frac{\sigma^2}{m}$$

- Only "uncorrelated parts" of $\hat{f}^{(j)}$ and $\hat{f}^{(k)}$ get "CLT effect"

- So is there any way to get ρ close to 0?

Decorrelating trees

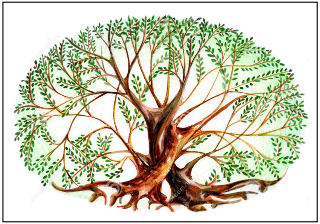
We might try to decorrelate any class of predictors $\hat{f}^{(j)}$

But let's focus on 

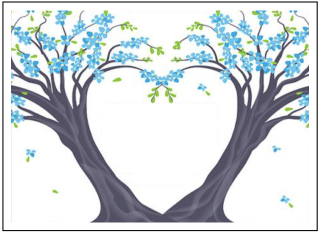
Decorrelating trees

We might try to decorrelate any class of predictors $\hat{f}^{(j)}$

But let's focus on 



$$\Downarrow$$

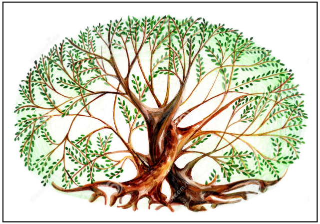


Decorrelating trees

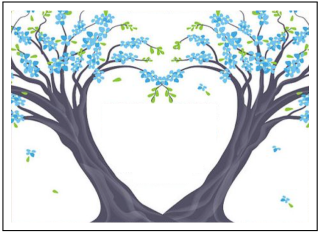
We might try to decorrelate any class of predictors $\hat{f}^{(j)}$

But let's focus on 

Trees are "correlated" because



$\Downarrow$



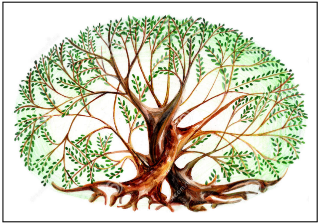
Decorrelating trees

We might try to decorrelate any class of predictors $\hat{f}^{(j)}$

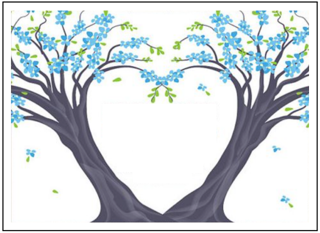
But let's focus on 

Trees are "correlated" because

1. bootstrap samples are about the same [nothing to do there]



$\Downarrow$



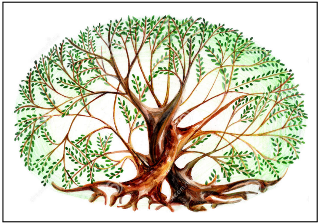
Decorrelating trees

We might try to decorrelate any class of predictors $\hat{f}^{(j)}$

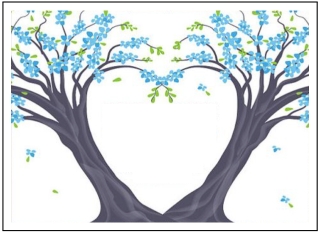
But let's focus on 

Trees are "correlated" because

1. bootstrap samples are about the same [nothing to do there]
2. influential features tend to be same [this we could influence...]



$\Downarrow$



Decorrelating trees

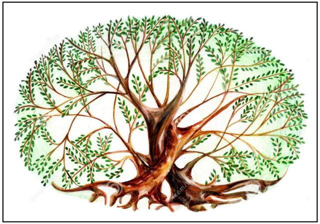
We might try to decorrelate any class of predictors $\hat{f}^{(j)}$

But let's focus on 

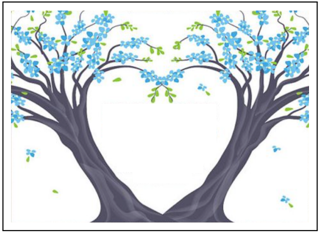
Trees are "correlated" because

1. bootstrap samples are about the same [nothing to do there]
2. influential features tend to be same [this we could influence...]

How can we make constructed tree not exactly the same?

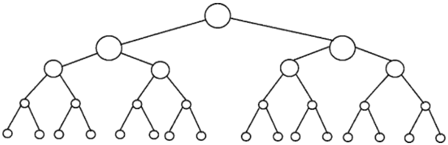


$\Downarrow$

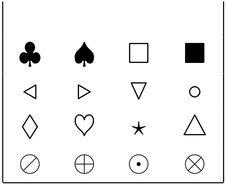


Random Forests

- Tree is constructed by recursive best splits



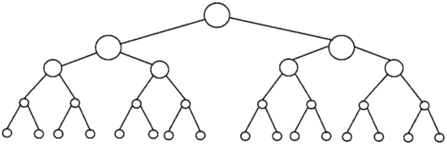
FEATURES

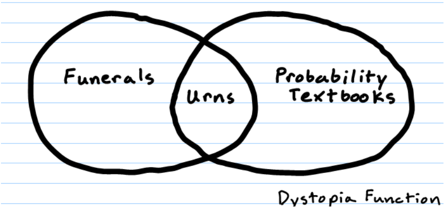


Features sampled with replacement for possible use **at each node**

Random Forests

- Tree is constructed by recursive best splits





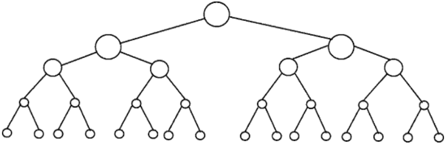
FEATURES

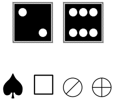
$\clubsuit$	$\spadesuit$	$\square$	$\blacksquare$
$\triangleleft$	$\triangleright$	$\triangledown$	$\circ$
$\diamondsuit$	$\heartsuit$	$\star$	$\triangle$
$\oslash$	$\oplus$	$\odot$	$\otimes$

Features sampled with replacement for possible use **at each node**

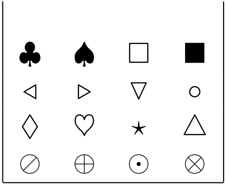
Random Forests

- Tree is constructed by recursive best splits
- We only use a random subset of features **at each split**





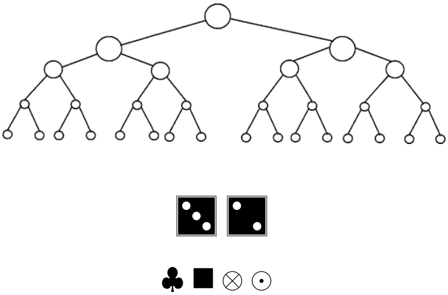
FEATURES



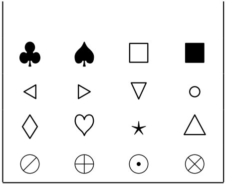
Features sampled with replacement for possible use **at each node**

Random Forests

- Tree is constructed by recursive best splits
- We only use a random subset of features **at each split**



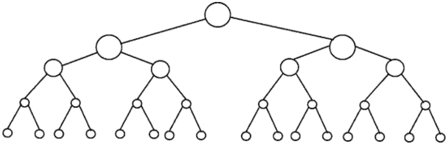
FEATURES

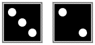


Features sampled with replacement for possible use **at each node**

Random Forests

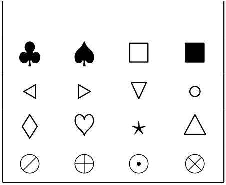
- Tree is constructed by recursive best splits
- We only use a random subset of features **at each split**





♣ ■ ⊗ ⊙

FEATURES

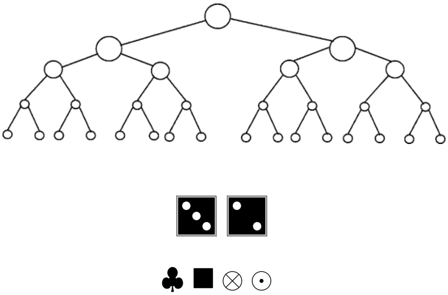


Features sampled with replacement for possible use **at each node**

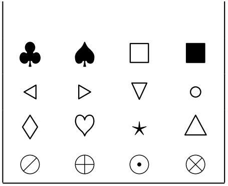
- *Temporary exclusion* – features includable again at each split

Random Forests

- Tree is constructed by recursive best splits
- We only use a random subset of features **at each split**



FEATURES

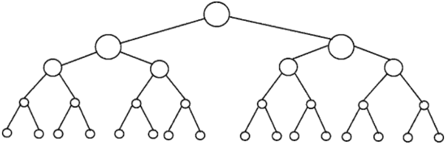


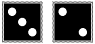
Features sampled with replacement for possible use **at each node**

- <u>*Temporary exclusion*</u> – features includable again at each split
- Typically select $\sqrt{p}$ (classification) & $\frac{p}{3}$ (regression) features

Random Forests

- Tree is constructed by recursive best splits
- We only use a random subset of features **at each split**





♣ ■ ⊗ ⊙

FEATURES

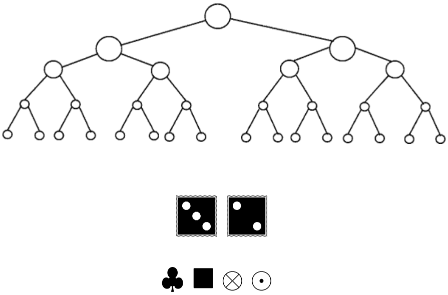
♣	♠	□	■
◁	▷	▽	○
◇	♡	⋆	△
⊘	⊕	⊙	⊗

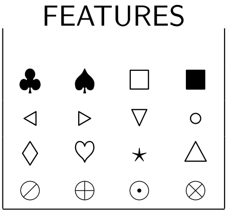
Features sampled with replacement for possible use **at each node**

- *Temporary exclusion* – features includable again at each split
- Typically select $\sqrt{p}$ (classification) & $\frac{p}{3}$ (regression) features
  Number features considered is a tuning parameter for what?

Random Forests

- Tree is constructed by recursive best splits
- We only use a random subset of features **at each split**





Features sampled with replacement for possible use **at each node**

- <u>*Temporary exclusion*</u> – features includable again at each split
- Typically select $\sqrt{p}$ (classification) & $\frac{p}{3}$ (regression) features
  Number features considered is a tuning parameter for what?
- More features means stronger but also more correlated trees...

Random Forest *Tuning*

$g(p)$: Number of features considered at each split

Random Forest *Tuning*

$g(p)$: Number of features considered at each split

m: Number of trees (i.e., number of bootstrapped samples)

Random Forest *Tuning*

$g(p)$: Number of features considered at each split

m: Number of trees (i.e., number of bootstrapped samples)

n_b: Sample size of each bootstrapped sample

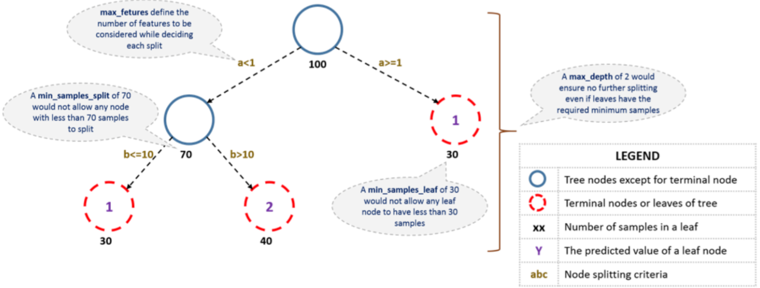
Random Forest *Tuning*

$g(p)$: Number of features considered at each split

m: Number of trees (i.e., number of bootstrapped samples)

n_b: Sample size of each bootstrapped sample

χ: Tree characteristics



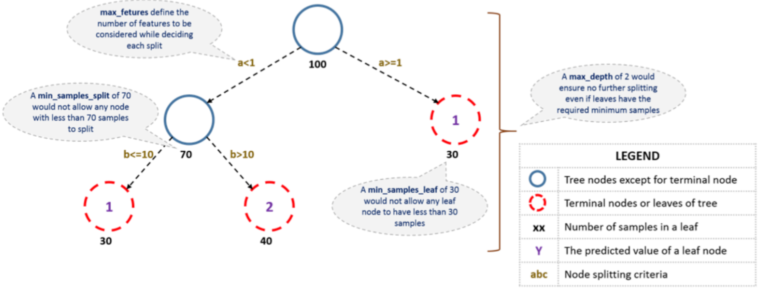
Random Forest *Tuning*

$g(p)$: Number of features considered at each split

m: Number of trees (i.e., number of bootstrapped samples)

n_b: Sample size of each bootstrapped sample

χ: Tree characteristics $\Longrightarrow \sum(\boldsymbol{Y}_i - \hat{\boldsymbol{Y}}_i)^2 + \alpha|T|$



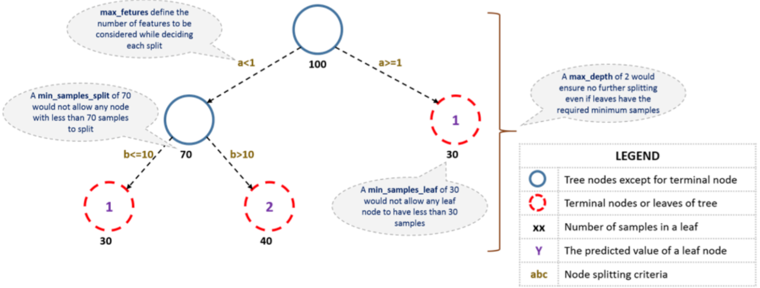
Random Forest *Tuning*

$g(p)$: Number of features considered at each split

m: Number of trees (i.e., number of bootstrapped samples)

n_b: Sample size of each bootstrapped sample

χ: Tree characteristics $\Longrightarrow \sum(\boldsymbol{Y}_i - \hat{\boldsymbol{Y}}_i)^2 + \alpha|T|$



- Random forests are often robust to tree characteristics:

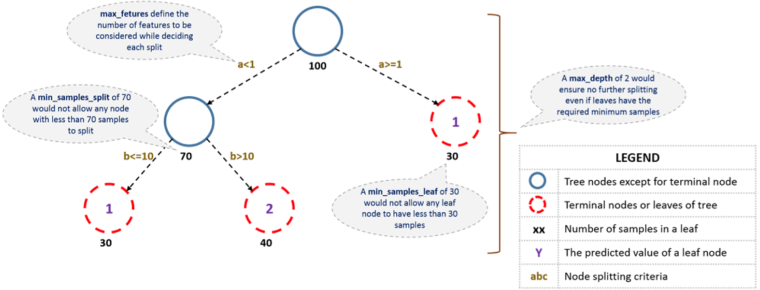
Random Forest *Tuning*

$g(p)$: Number of features considered at each split

m: Number of trees (i.e., number of bootstrapped samples)

n_b: Sample size of each bootstrapped sample

χ: Tree characteristics $\Longrightarrow \sum(\boldsymbol{Y}_i - \hat{\boldsymbol{Y}}_i)^2 + \alpha|T|$



- Random forests are often robust to tree characteristics: **very large numbers of "bushy" trees typically used** &

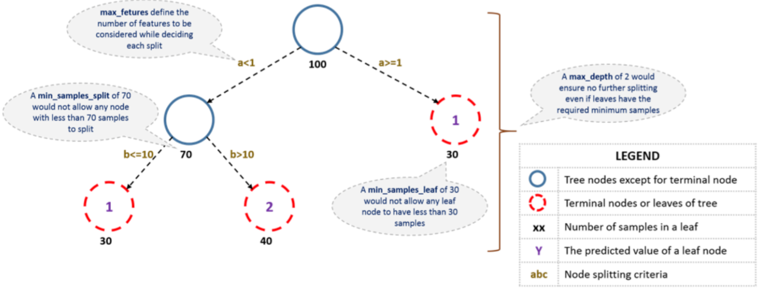
Random Forest *Tuning*

$g(p)$: Number of features considered at each split

m: Number of trees (i.e., number of bootstrapped samples)

n_b: Sample size of each bootstrapped sample

χ: Tree characteristics $\Longrightarrow \sum(\boldsymbol{Y}_i - \hat{\boldsymbol{Y}}_i)^2 + \alpha|T|$



- Random forests are often robust to tree characteristics: **very large numbers of "bushy" trees typically used** & approach **state of the art performance out of the box**

Random Forest *Computational Notes*

- Trees can be computationally expensive to fit

Random Forest *Computational Notes*

- Trees can be computationally expensive to fit
  - But they can be fit in parallel

Random Forest *Computational Notes*

- Trees can be computationally expensive to fit
  - But they can be fit in parallel
  - At some point $\rho\sigma^2 + (1-\rho)\frac{\sigma^2}{m}$ reaches an asymptote

Random Forest *Computational Notes*

- Trees can be computationally expensive to fit
  - But they can be fit in parallel
  - At some point $\rho\sigma^2 + (1-\rho)\frac{\sigma^2}{m}$ reaches an asymptote
- Cross-validation can be computationally demanding

Random Forest *Computational Notes*

- Trees can be computationally expensive to fit
  - But they can be fit in parallel
  - At some point $\rho\sigma^2 + (1-\rho)\frac{\sigma^2}{m}$ reaches an asymptote
- Cross-validation can be computationally demanding
  - Parameters can be tuned using OOB

    You have built-in test samples so you don't need K-folds – you score a model right when it's fit at a specific tuning parameter!

Random Forest *Computational Notes*

- Trees can be computationally expensive to fit
  - But they can be fit in parallel
  - At some point $\rho\sigma^2 + (1-\rho)\frac{\sigma^2}{m}$ reaches an asymptote
- Cross-validation can be computationally demanding
  - Parameters can be tuned using OOB

    You have built-in test samples so you don't need K-folds – you score a model right when it's fit at a specific tuning parameter!

  - Cross-validation is still needed for methodology comparisons...

    OOB samples don't appear in other model fitting contexts – so you still need a common "out of sample" set to compare on...

Interpreting Tree Ensembles

"Plentie is nodeintie, ye see not your owne ease. I see, ye can not see the wood for trees."

– J. Heywood (1546)

"You can't see the forest for the treeeeeeees!"

– M. Manson (1996)

Interpreting Tree Ensembles

"Plentie is nodeintie, ye see not your owne ease. I see, ye can not see the wood for trees."

– J. Heywood (1546)

"You can't see the forest for the treeeeeeees!"

– M. Manson (1996)

"I find that people be all like,

'Machine learning models are "Black Boxes" and you can't interpret them...',

and I be like, 'Damn you people have some back-ass wackwards ideas...'"

– S. Schwartz (Yesterday)

Interpreting Models

Change X_i and see what happens

It's just really not that hard...

Change X_i and see what happens

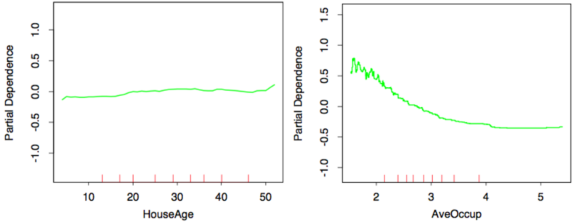
It's just really not that hard...

That's all your doing in a linear model:

$$\beta_0 + \beta_1 X_1 + \beta_1 X_1 + \beta_2 X_2 + \cdots$$

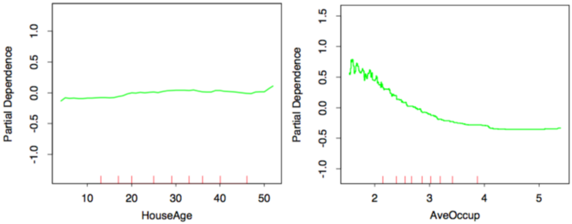
Partial Dependency Plots

1. Set feature $V = v$ for a single values v



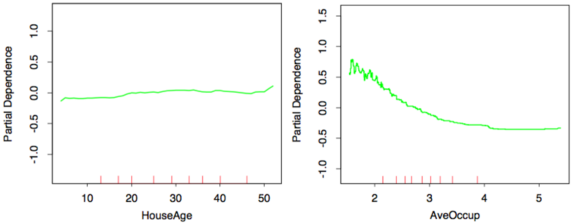
Partial Dependency Plots

1. Set feature $V = v$ for a single values v
2. Record average predictions $\frac{1}{n}\sum_{i=1}^{n} \hat{Y}_i^{(v)}$ over "synthetic" data



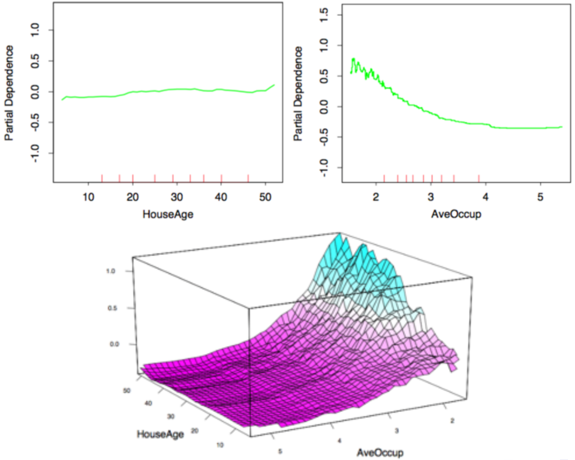
Partial Dependency Plots

1. Set feature $V = v$ for a single values v
2. Record average predictions $\frac{1}{n}\sum_{i=1}^{n}\hat{Y}_i^{(v)}$ over "synthetic" data
3. Plot $\frac{1}{n}\sum_{i=1}^{n}\hat{Y}_i^{(v)}$ at v



Partial Dependency Plots

1. Set feature $V = v$ for a single values v
2. Record average predictions $\frac{1}{n}\sum_{i=1}^{n} \hat{Y}_i^{(v)}$ over "synthetic" data
3. Plot $\frac{1}{n}\sum_{i=1}^{n} \hat{Y}_i^{(v)}$ at v



But there are some other *Very Cool* model interpretation options available in tree based contexts as well...

Variable Importance (v1.0): “share of information gain”

1. Split are made on the feature minimizing RSS or Gini/Entropy

Variable Importance (v1.0): "share of information gain"

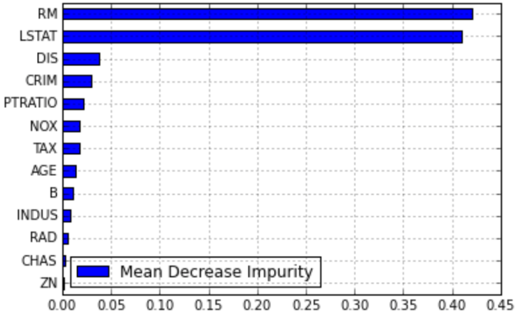
1. Split are made on the feature minimizing RSS or Gini/Entropy
2. For split s, cumulate the explanatory contribution $\xi_V^{(s)}$ for V

Variable Importance (v1.0): "share of information gain"

1. Split are made on the feature minimizing RSS or Gini/Entropy
2. For split s, cumulate the explanatory contribution $\xi_V^{(s)}$ for V
3. Average the scores $\xi_V^{(s)}$ for each feature V **across all trees**

Variable Importance (v1.0): "share of information gain"

1. Split are made on the feature minimizing RSS or Gini/Entropy
2. For split s, cumulate the explanatory contribution $\xi_V^{(s)}$ for V
3. Average the scores $\xi_V^{(s)}$ for each feature V **across all trees**



In Boston, the most relevant associations with neighborhood home values are (1) number of rooms and (2) proportion of low income households in the neighborhood

Variable Importance (v2.0): "error caused by losing V"

1. Get OOB accuracy using all features, including original V

Variable Importance (v2.0): "error caused by losing V"

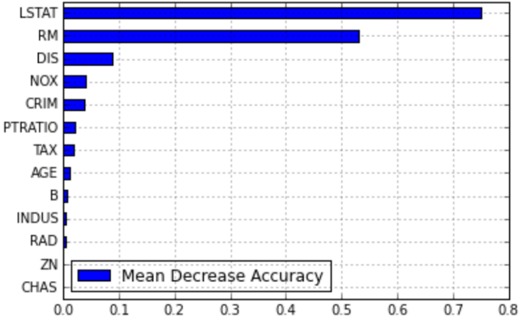
1. Get OOB accuracy using all features, including original V
2. Get OOB accuracy using all features & V^*, a permuted V

Variable Importance (v2.0): "error caused by losing V"

1. Get OOB accuracy using all features, including original V
2. Get OOB accuracy using all features & V^*, a permuted V
3. Compare the differences between 1 and 2 for all features

Variable Importance (v2.0): "error caused by losing V"

1. Get OOB accuracy using all features, including original V
2. Get OOB accuracy using all features & V^*, a permuted V
3. Compare the differences between 1 and 2 for all features



This approach suggests prediction is more sensitive to (1) low income proportion rather than (2) room number

Variable Importance (v3.0): “what influences predictions”

1. Follow test samples down the tree through each split s on V

Variable Importance (v3.0): "what influences predictions"

1. Follow test samples down the tree through each split s on V
2. Record the change in prediction $\epsilon_V^{(s)}$ due to V split at s

Variable Importance (v3.0): "what influences predictions"

1. Follow test samples down the tree through each split s on V
2. Record the change in prediction $\epsilon_V^{(s)}$ due to V split at s
3. Sum up all the changes $\epsilon_V^{(s)}$ due to feature V

Variable Importance (v3.0): "what influences predictions"

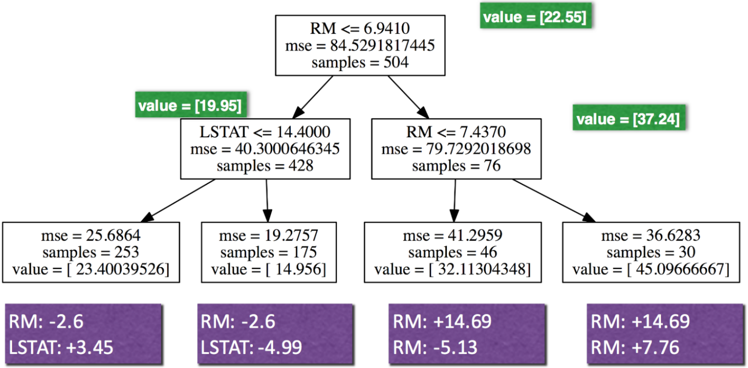
1. Follow test samples down the tree through each split s on V
2. Record the change in prediction $\epsilon_V^{(s)}$ due to V split at s
3. Sum up all the changes $\epsilon_V^{(s)}$ due to feature V

   These values are then averaged across trees

Variable Importance (v3.0): "what influences predictions"

1. Follow test samples down the tree through each split s on V
2. Record the change in prediction $\epsilon_V^{(s)}$ due to V split at s
3. Sum up all the changes $\epsilon_V^{(s)}$ due to feature V

   These values are then averaged across trees



Variable Importance (v3.0): "what influences predictions"

1. Follow test samples down the tree through each split s on V
2. Record the change in prediction $\epsilon_V^{(s)}$ due to V split at s
3. Sum up all the changes $\epsilon_V^{(s)}$ due to feature V

   These values are then averaged by tree

Distance from city-center

Fraction of lower-class residents

Average number of rooms

	DIS	INDUS	LSTAT	NOX	PTRATIO	RAD	RM	TAX	ZN
Prediction 1	6.11	0.10	2.67	-0.02	-0.19	0.06	-2.63	-0.20	0.03
Prediction 2	6.22	0.16	2.56	-0.01	-0.19	0.06	-3.15	-0.16	0.03
Prediction 3	-0.70	0.04	7.42	-0.11	0.42	-0.02	1.10	-0.14	-0.05
Prediction 4	-0.53	0.25	3.50	0.16	1.46	0.13	2.21	-0.24	0.11
Prediction 5	-0.68	0.15	7.86	0.03	0.85	0.01	-1.14	-0.16	0.18
Prediction 6	0.18	-0.26	8.62	-0.19	-0.02	-0.07	-1.83	-0.34	-0.05

Features effects can be characterized on individual samples (!)

Variable Importance (v4.0): “what the tree predicts with”

1. Calculate proportion of samples visiting feature V in each tree

Variable Importance (v4.0): "what the tree predicts with"

1. Calculate proportion of samples visiting feature V in each tree (conditions higher in the tree typically visited by more data)

Variable Importance (v4.0): “what the tree predicts with”

1. Calculate proportion of samples visiting feature V in each tree (conditions higher in the tree typically visited by more data)
2. Average the “proportion of samples visited” across all trees

Fine

So don't tell me machine learning models, *particularly tree-based methods* are uninterpretable "Black-Boxes". I'll kick your butt.

Fine

So don't tell me machine learning models, *particularly tree-based methods* are uninterpretable "Black-Boxes". I'll kick your butt.

Now:

Tree $<$ **Bagging** $<$ **Random Forests** $<$ **________?**