DATASCI 347 Machine Learning

Lecture 14: Random forests and boosting

Ruoxuan Xiong

Suggested reading: ISL Chapter 8 and 10

Logistics for group project presentation

- We will have a project presentation on Wednesday, 3/18
- Each group has 5 minutes. Please sign up a slot
- Please prepare 3 slides for the presentation and send them to me before 11:59 PM on Tuesday, 3/17
- Ad hoc office hours from 4:00-5:00 PM on Tuesday, 3/17

Template for the presentation

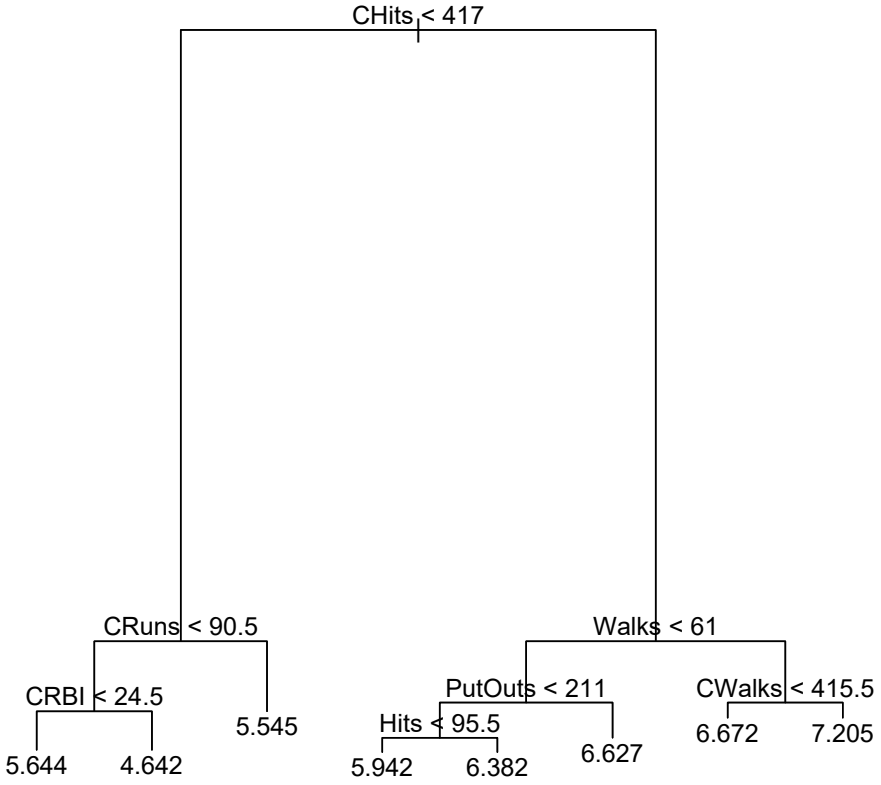
- **P1:** What is the problem? Why is it interesting?
- **P2:** Which dataset do you plan to use?
- **P3:** What are the approaches you plan to explore?
- **P4:** Why are these approaches reasonable to consider? Also, include any specific limitations.
- **P5:** What is the expected timeline for conducting the project? Also, including the division of work between every student in the team.

Lecture plan

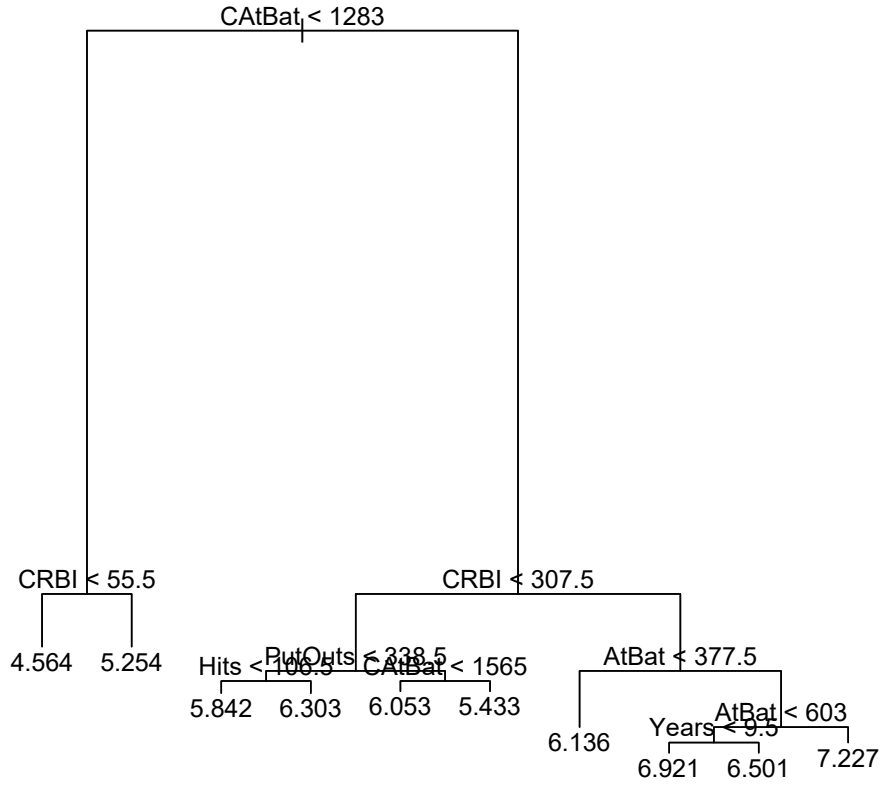
- Random forests
- Boosting

Decision tree has a high variance

- **Example:** Predicting a baseball player's salary
  - Split the training data into two equal-sized parts at random creates disparity



Subsample 1



Subsample 2

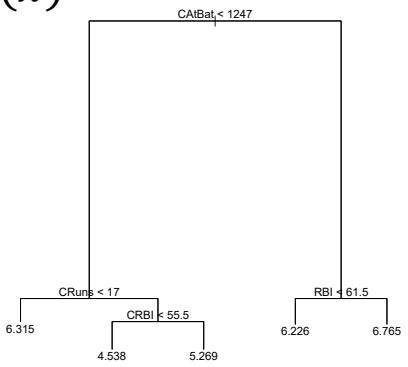
Bagging decision trees to reduce variance

- **Idea: Bootstrap aggregation** (mean / majority of predictions for regression / classification tasks)

Sample #1

X_1	Y_1
X_2	Y_2
X_1	Y_1
X_5	Y_5
X_4	Y_4

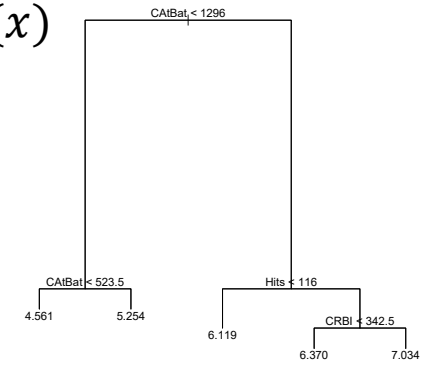
$\hat{f}^1(x)$



Sample #2

X_4	Y_4
X_1	Y_1
X_3	Y_3
X_2	Y_2
X_3	Y_3

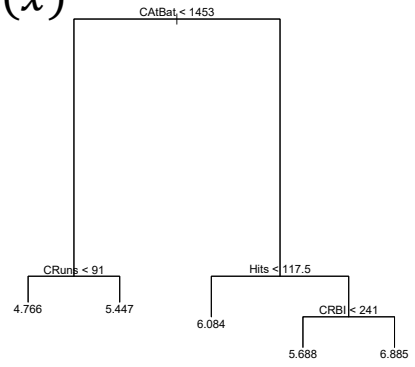
$\hat{f}^2(x)$



Sample #3

X_5	Y_5
X_2	Y_2
X_3	Y_3
X_2	Y_2
X_1	Y_1

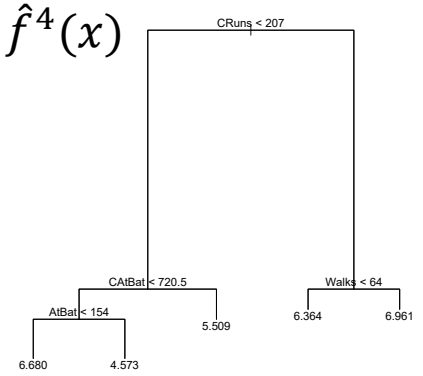
$\hat{f}^3(x)$



Sample #4

X_5	Y_5
X_3	Y_3
X_3	Y_3
X_1	Y_1
X_2	Y_2

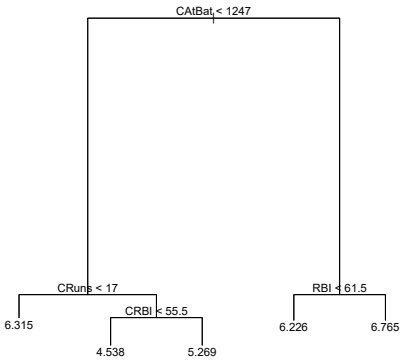
$\hat{f}^4(x)$



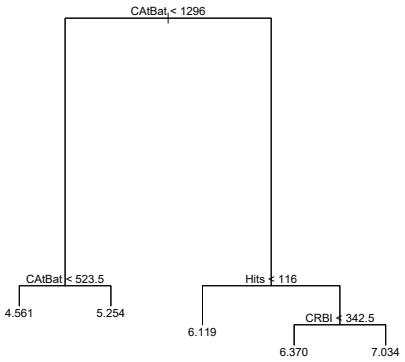
Bagging has a problem

- The trees produced by different Bootstrap samples can be very similar

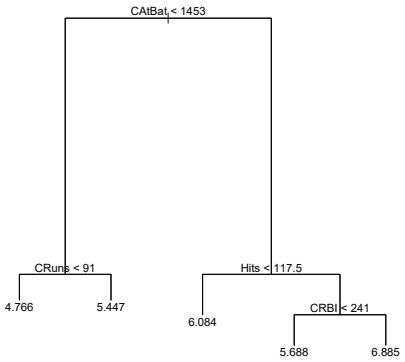
$\hat{f}^1(x)$



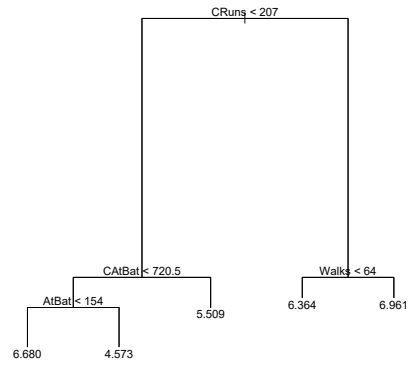
$\hat{f}^2(x)$



$\hat{f}^3(x)$



$\hat{f}^4(x)$



- Three decision trees first split by CAtBat

Random forests

- **Random forests: Bagging plus random sampling of predictors**
  - We fit a decision tree to each Bootstrap samples
  - When fitting the tree, we select a random subset of $m < p$ predictors to consider in each step
  - This will lead to very different trees from each sample
  - Finally, aggregate (mean or majority vote) the prediction of each tree

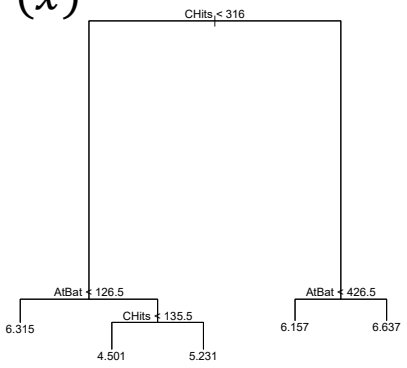
Random forests

- **Random forests** to predict a baseball player salary: $p = 19, m = 5$
  - $X_{i,j}$: jth predictor of observation i

Sample #1

$X_{1,4}$	$X_{1,17}$	$X_{1,9}$	$X_{1,6}$	$X_{1,1}$	Y_1
$X_{2,4}$	$X_{2,17}$	$X_{2,9}$	$X_{2,6}$	$X_{2,1}$	Y_2
$X_{1,4}$	$X_{1,17}$	$X_{1,9}$	$X_{1,6}$	$X_{1,1}$	Y_1
$X_{5,4}$	$X_{5,17}$	$X_{5,9}$	$X_{5,6}$	$X_{5,1}$	Y_5
$X_{4,4}$	$X_{4,17}$	$X_{4,9}$	$X_{4,6}$	$X_{4,1}$	Y_4

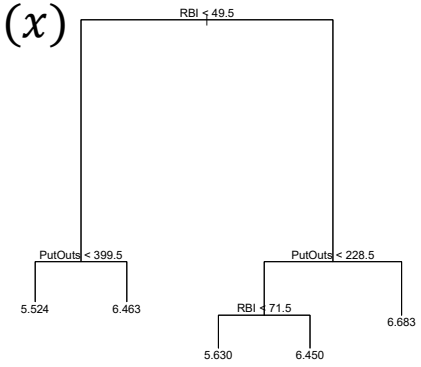
$\hat{f}^1(x)$



Sample #2

$X_{4,16}$	$X_{4,5}$	$X_{4,19}$	$X_{4,18}$	$X_{4,1}$	Y_4
$X_{1,16}$	$X_{1,5}$	$X_{1,19}$	$X_{1,18}$	$X_{1,1}$	Y_1
$X_{3,16}$	$X_{3,5}$	$X_{3,19}$	$X_{3,18}$	$X_{3,1}$	Y_3
$X_{2,16}$	$X_{2,5}$	$X_{2,19}$	$X_{2,18}$	$X_{2,1}$	Y_2
$X_{3,16}$	$X_{3,5}$	$X_{3,19}$	$X_{3,18}$	$X_{3,1}$	Y_3

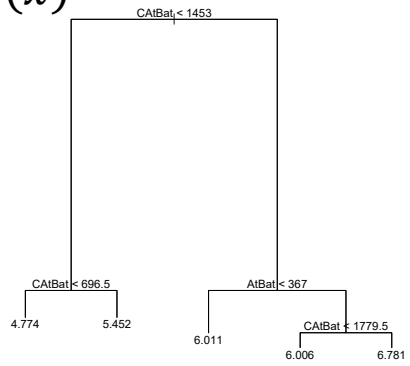
$\hat{f}^2(x)$



Sample #3

$X_{5,6}$	$X_{5,14}$	$X_{5,1}$	$X_{5,4}$	$X_{5,8}$	Y_5
$X_{2,6}$	$X_{2,14}$	$X_{2,1}$	$X_{2,4}$	$X_{2,8}$	Y_2
$X_{3,6}$	$X_{3,14}$	$X_{3,1}$	$X_{3,4}$	$X_{3,8}$	Y_3
$X_{2,6}$	$X_{2,14}$	$X_{2,1}$	$X_{2,4}$	$X_{2,8}$	Y_2
$X_{1,6}$	$X_{1,14}$	$X_{1,1}$	$X_{1,4}$	$X_{1,8}$	Y_1

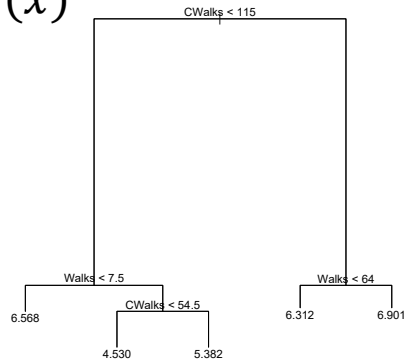
$\hat{f}^3(x)$



Sample #4

$X_{5,17}$	$X_{5,6}$	$X_{5,13}$	$X_{5,5}$	$X_{5,7}$	Y_5
$X_{3,17}$	$X_{3,6}$	$X_{3,13}$	$X_{3,5}$	$X_{3,7}$	Y_3
$X_{3,17}$	$X_{3,6}$	$X_{3,13}$	$X_{3,5}$	$X_{3,7}$	Y_3
$X_{1,17}$	$X_{1,6}$	$X_{1,13}$	$X_{1,5}$	$X_{1,7}$	Y_1
$X_{2,17}$	$X_{2,6}$	$X_{2,13}$	$X_{2,5}$	$X_{2,7}$	Y_2

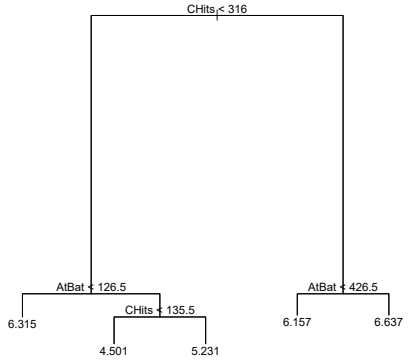
$\hat{f}^4(x)$



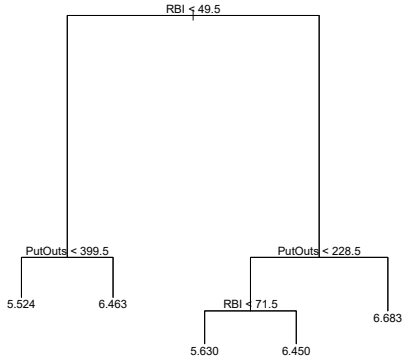
Random forests

	AtBat	Hits	HmRun	Runs	RBI	Walks	Years	CAtBat	CHits	CHmRun	CRuns	CRBI	CWalks	League	Division	PutOuts	Assists	Errors	Salary	NewLeague
-Andy Allanson	293	66	1	30	29	14	1	293	66	1	30	29	14	A	E	446	33	20	NA	A

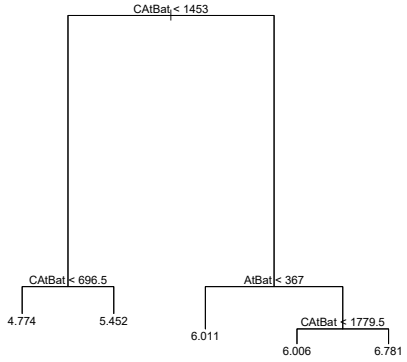
$\hat{f}^1(x)$



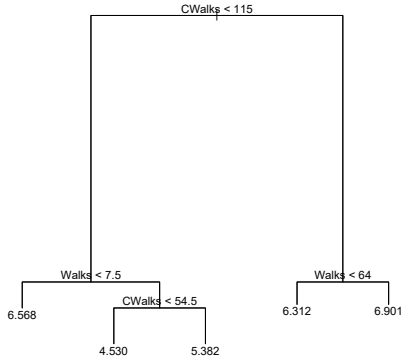
$\hat{f}^2(x)$



$\hat{f}^3(x)$



$\hat{f}^4(x)$



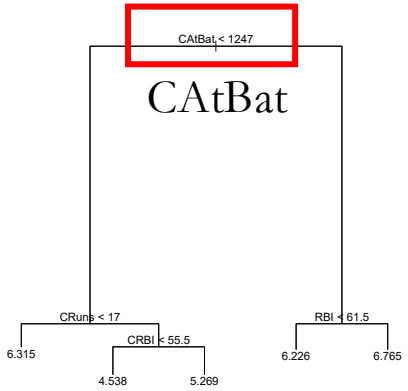
$$\hat{f}_{rf}(x) = \frac{1}{4}\{\hat{f}^1(x) + \hat{f}^2(x) + \hat{f}^3(x) + \hat{f}^4(x)\}$$

- More generally, if we have B bootstrapped training data sets, $\hat{f}_{rf}(x) = \frac{1}{B}\{\hat{f}^1(x) + \hat{f}^2(x) + \cdots + \hat{f}^B(x)\}$

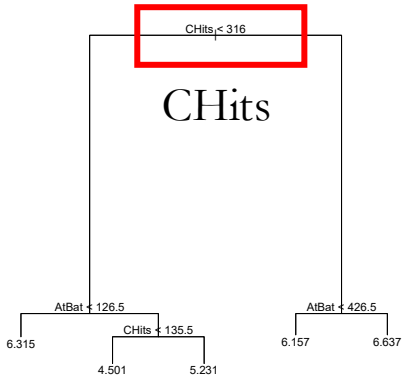
Bagging vs. random forests

$\hat{f}^1(x)$

Bagging

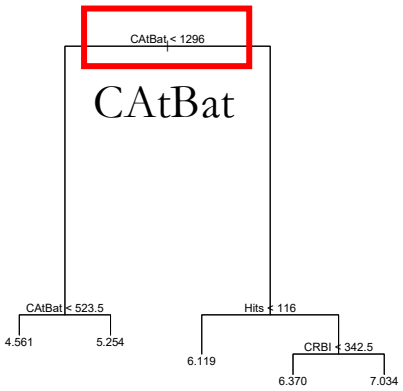


Random forests

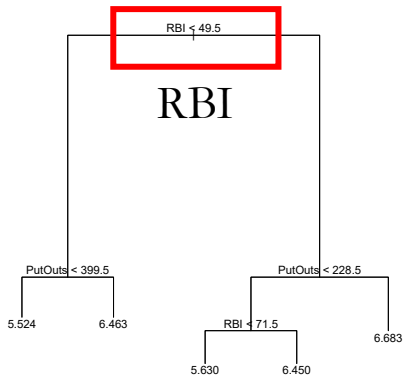


$\hat{f}^2(x)$

Bagging

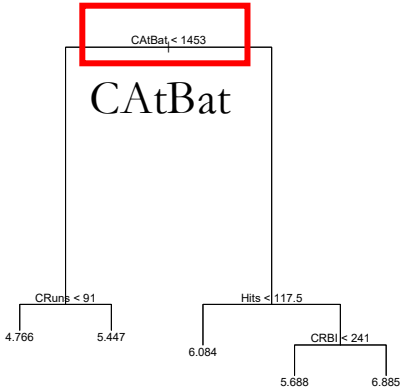


Random forests

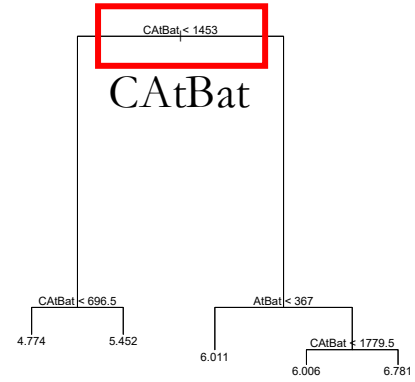


$\hat{f}^3(x)$

Bagging

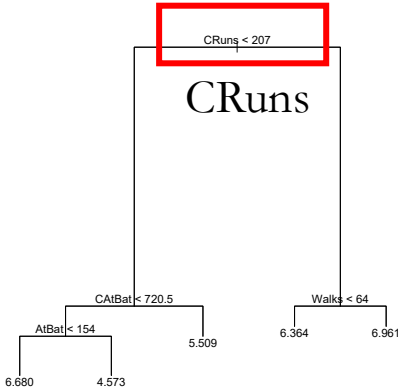


Random forests

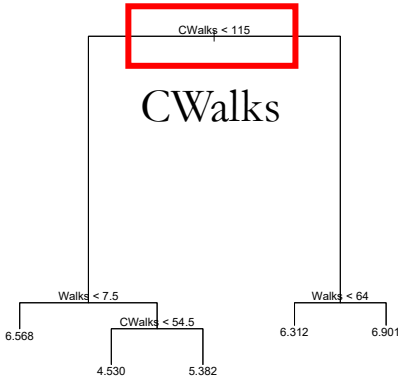


$\hat{f}^4(x)$

Bagging

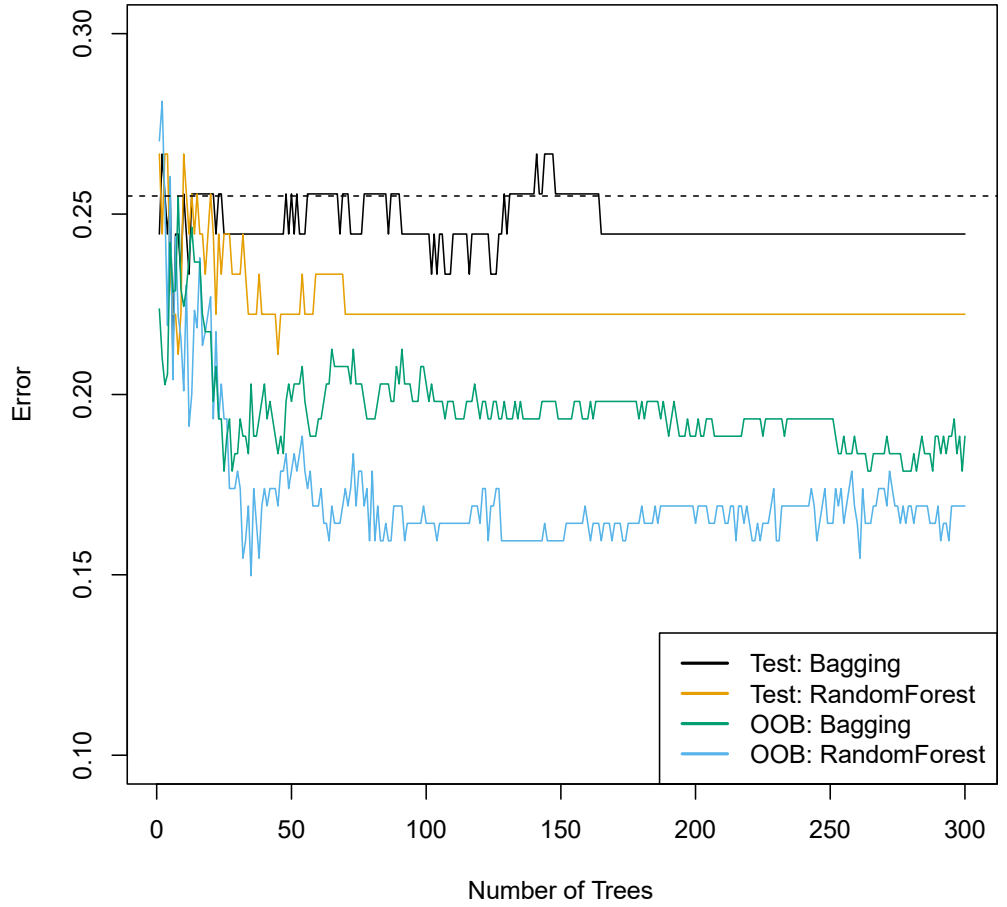


Random forests



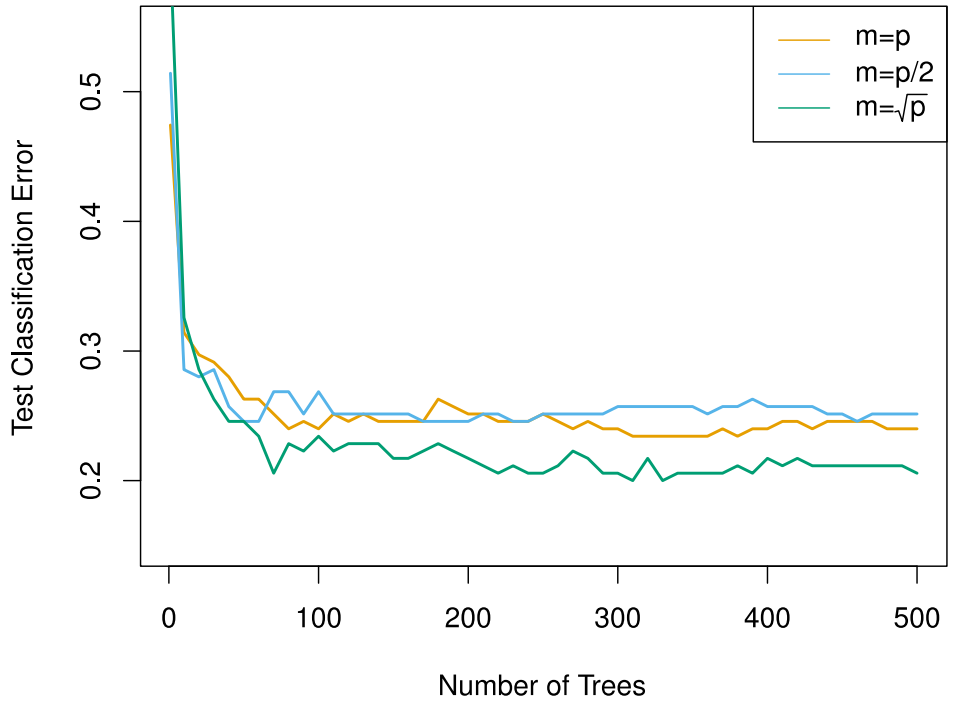
Bagging vs. random forests

- **Example:** Predict whether a patient with chest pain has heart disease
  - Random forests outperform bagging
  - $m = \sqrt{p}$



Random forests, choosing m

- **Example:** Predict cancer type (either normal or 1 of 14 different types of cancer) based on 500 genes
  - Error rate of a single tree: 45.6%
  - Using 400 trees is sufficient



- The optimal m is usually around $\sqrt{p}$, but this can be used as a tuning parameter

Lecture plan

- Random forests

- Boosting

Boosted trees

- **Boosted trees**
  - Trees are grown sequentially using the information left from previously grown trees
  - Each tree is fit on a **modified version** of the original data

- Random forests involve a lot of randomness, while boosting has less randomness

- Random forests are easy to parallelize, but boosting is often more scalable (through smaller tree size)

Boosting (combine weak learners)

- **Step 1**: Set $\hat{f}(x) = 0$, and $r_i = y_i$ for $i = 1, \cdots, n$.
- **Step 2**: For $b = 1, \cdots, B$, iterate:
  - Fit a decision tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the response $r_1, \cdots, r_n$
  - Update the prediction to

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

  - Update the residuals

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

- **Step 3**: Output the final model

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x)$$

Boosting

X_1	Y_1
X_2	Y_2
X_3	Y_3
X_4	Y_4
X_5	Y_5

$r_i^1 \leftarrow Y_i - \lambda \hat{f}^1(X_i)$

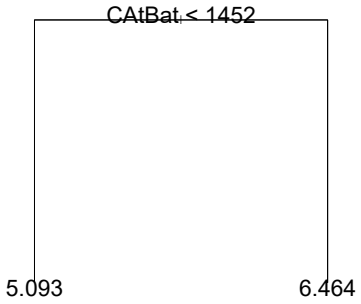
X_1	r_1^1
X_2	r_2^1
X_3	r_3^1
X_4	r_4^1
X_5	r_5^1

$r_i^2 \leftarrow r_i^1 - \lambda \hat{f}^2(X_i)$

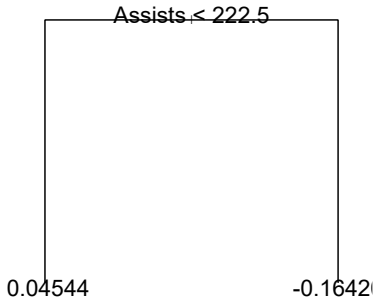
X_1	r_1^2
X_2	r_2^2
X_3	r_3^2
X_4	r_4^2
X_5	r_5^2

$\cdots$

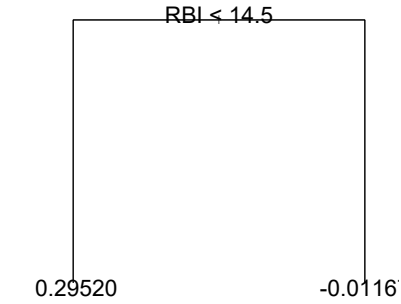
$\hat{f}^1(x)$



$\hat{f}^2(x)$



$\hat{f}^3(x)$



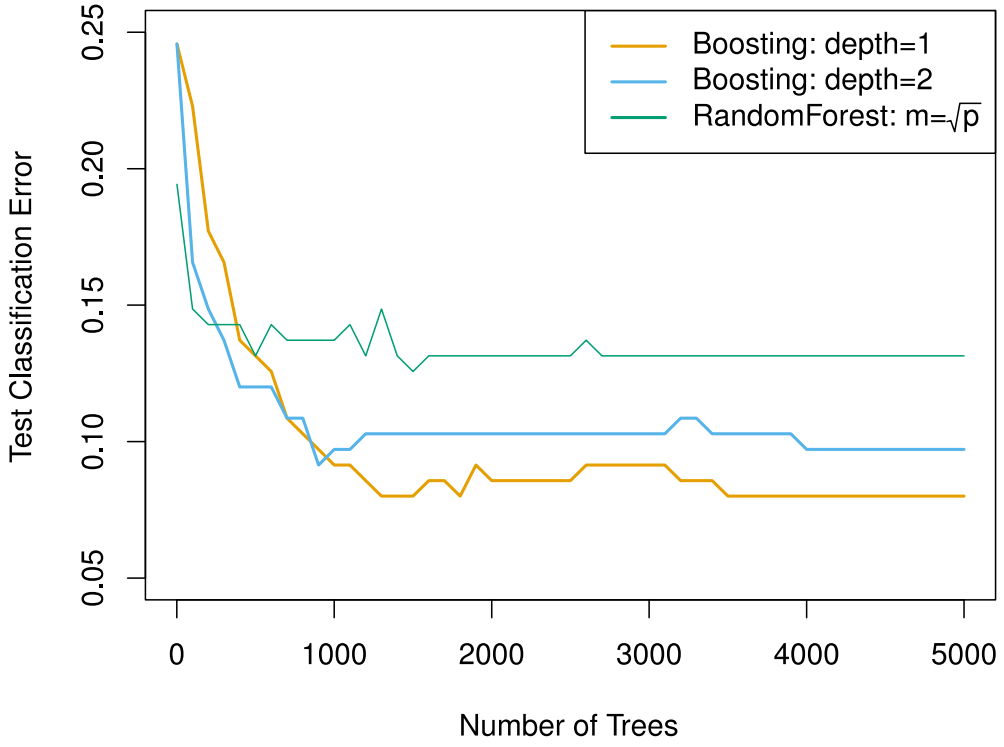
$$\hat{f}(x) = \lambda \hat{f}^1(x) + \lambda \hat{f}^2(x) + \lambda \hat{f}^3(x) + \cdots + \lambda \hat{f}^B(x)$$

Tuning parameters in boosting

- The number of trees B
  - Boosting can overfit if B is too large (a.k.a. early stopping)
  - Use cross-validation to select B
- The shrinkage parameter λ
  - Typical values are 0.01 or 0.001
  - Very small λ requires a large B to achieve good performance
- The number of splits/depth d in each tree
  - $d = 1$ works well
- Remark:
  - Also called **gradient boosting**
  - λ is learning rate

Boosting vs. random forests

- **Example:** Predict cancer type (either normal or 1 of 14 different types of cancer) based on 500 genes
  - $\lambda = 0.01$
  - Depth-1 trees outperform depth-2 trees
  - Both outperform random forests



AdaBoost

- A particular method of training a boosted **classifier**
- For example, $Y \in \{-1,1\}$ is binary

Initial weight

X_1	Y_1	1/5
X_2	Y_2	1/5
X_3	Y_3	1/5
X_4	Y_4	1/5
X_5	Y_5	1/5

Fitted tree $\hat{f}^1(x)$
Correctly predict all samples besides Y_3 and Y_5

$$Total\ Error = \frac{1}{n}\sum_i I(\hat{f}(X_i) \neq Y_i) = \frac{2}{5}$$

$$Amount\ of\ stay = \frac{1}{2}\log\frac{1 - Total\ Error}{Total\ Error} = \frac{1}{2}\log\frac{1-2/5}{2/5} = 0.088$$

Next we *increase* the sample weight for the sample that was incorrectly classified. We *decrease* the sample weight for the sample that was correctly classified.

AdaBoost

- A particular method of training a boosted **classifier**
- For example, $Y \in \{-1,1\}$ is binary

Initial weight

X_1	Y_1	1/5
X_2	Y_2	1/5
X_3	Y_3	1/5
X_4	Y_4	1/5
X_5	Y_5	1/5

Fitted tree $\hat{f}^1(x)$
Correctly predict all samples besides Y_3 and Y_5

$$Amount\ of\ stay = \frac{1}{2}\log\frac{1 - Total\ Error}{Total\ Error} = \frac{1}{2}\log\frac{1-2/5}{2/5} = 0.088$$

Next we *increase* the sample weight for the sample that was incorrectly classified

$$New\ sample\ weight = sample\ weight \times \exp(Amount\ of\ stay)$$

$$New\ sample\ weight = \frac{1}{5} \times \exp(Amount\ of\ stay) = 0.2184$$

We *decrease* the sample weight for the sample that was correctly classified

$$New\ sample\ weight = sample\ weight \times \exp(-Amount\ of\ stay)$$

$$New\ sample\ weight = \frac{1}{5} \times \exp(-Amount\ of\ stay) = 0.1831$$

EMORY

AdaBoost

- A particular method of training a boosted classifier
- For example, $Y \in \{-1,1\}$ is binary

Initial weight

X_1	Y_1	1/5
X_2	Y_2	1/5
X_3	Y_3	1/5
X_4	Y_4	1/5
X_5	Y_5	1/5

Update weight

New weight

X_1	Y_1	0.1831
X_2	Y_2	0.1831
X_3	Y_3	0.2184
X_4	Y_4	0.1831
X_5	Y_5	0.2184

Fitted tree $\hat{f}^1(x)$
Correctly predict all samples besides Y_3 and Y_5

$$Sum\ of\ the\ weights\ =\ 0.9862 \neq 1$$

AdaBoost

- A particular method of training a boosted classifier
- For example, $Y \in \{-1,1\}$ is binary

Initial weight

X_1	Y_1	1/5
X_2	Y_2	1/5
X_3	Y_3	1/5
X_4	Y_4	1/5
X_5	Y_5	1/5

Update weight

New weight

X_1	Y_1	0.1831/0.9862
X_2	Y_2	0.1831/0.9862
X_3	Y_3	0.2184/0.9862
X_4	Y_4	0.1831/0.9862
X_5	Y_5	0.2184/0.9862

…

Fitted tree $\hat{f}^1(x)$
Correctly predict all samples besides Y_3 and Y_5

Fitted tree $\hat{f}^2(x)$

$$\hat{f}(x) = Sign\left(\sum_{b=1}^{B} \lambda_b \hat{f}^b(x)\right)$$

Predict the most likely class

XGBoost

- **XGBoost** (eXtreme Gradient Boosting) is an open-source software library that provides a ***regularized* gradient-boosting** framework
- Objective function is

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

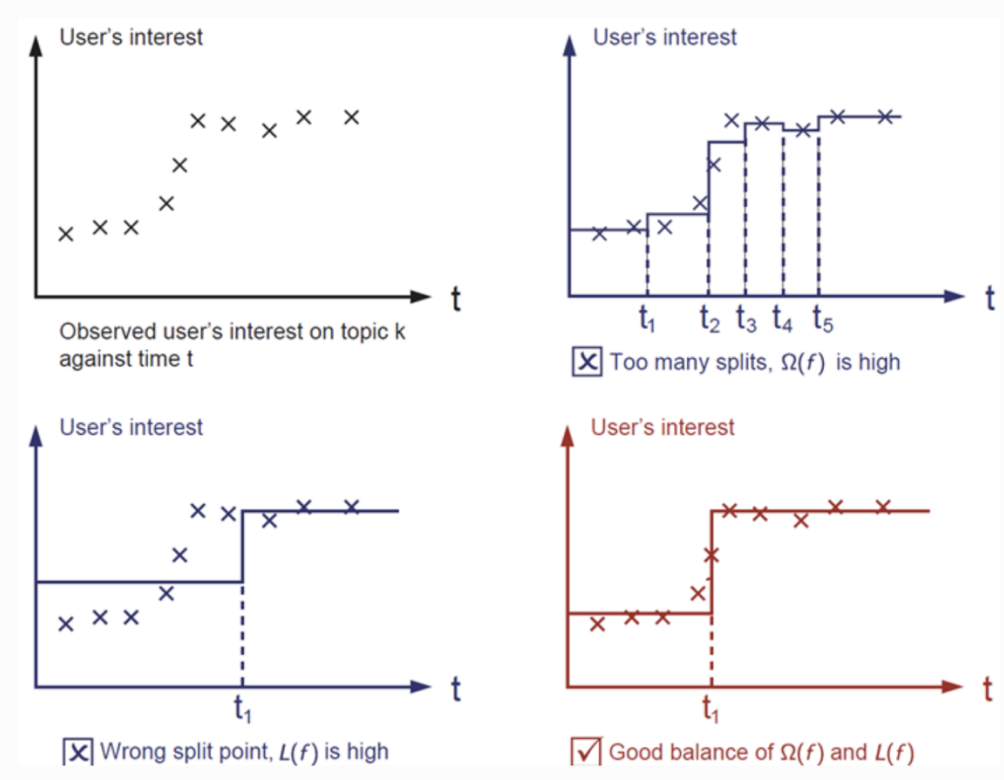
  - $L = \sum_i l(Y_i, \hat{Y}_i)$ is the training loss function
    - Regression problem: $l(Y_i, \hat{Y}_i) = (Y_i - \hat{Y}_i)^2$
    - Classification problem: L can be the logistic loss

XGBoost

- **XGBoost** (eXtreme Gradient Boosting) is an open-source software library that provides a ***regularized* gradient-boosting** framework
- Objective function is

$$obj(\theta) = L(\theta) + \Omega(\theta)$$

  - $\Omega = \sum_b \omega(f^b)$ is the regularization term
    - $\omega(f) = \gamma T + \frac{1}{2}\zeta \sum_{j=1}^{T} w_j^2$ where $f^b(x) = w_{q(x)}$, and q is a function assigning each data point to the corresponding leaf



Additive training

- Parameters θ of trees: structure of the tree and leaf predicted values
- Let the prediction value at step t be $\hat{Y}_i^{(t)}$. Then we have (ignore λ)
  - $\hat{Y}_i^{(0)} = 0$
  - $\hat{Y}_i^{(1)} = \hat{Y}_i^{(0)} + f^1(X_i) = f^1(X_i)$
  - $\hat{Y}_i^{(2)} = \hat{Y}_i^{(1)} + f^2(X_i) = f^1(X_i) + f^2(X_i)$
  - $\cdots$
  - $\hat{Y}_i^{(t)} = \hat{Y}_i^{(t-1)} + f^t(X_i) = \sum_{b=1}^{t} f^b(X_i)$
- XGBoost provides an approach to obtain $f^t(X_i)$ that can reduce $obj(\theta)$

Taylor expansion of the objective function (optional)

- Objective function at step t

$$obj^{(t)} = \sum_{i=1}^{n} l\left(Y_i, \hat{Y}_i^{(t)}\right) + \sum_{b=1}^{t} \omega\left(f^b\right)$$
$$= \sum_{i=1}^{n} l\left(Y_i, \hat{Y}_i^{(t-1)} + f^t(X_i)\right) + \sum_{b=1}^{t} \omega\left(f^b\right)$$

- We take the Taylor expansion of the loss function to the second order

$$l\left(Y_i, \hat{Y}_i^{(t-1)} + f^t(X_i)\right) = l\left(Y_i, \hat{Y}_i^{(t-1)}\right) + g_i f^t(X_i) + \frac{1}{2} h_i [f^t(X_i)]^2$$

  - g_i and h_i are the first-order and second-order derivatives of $l\left(Y_i, \hat{Y}_i^{(t-1)}\right)$ w.r.t. $\hat{Y}_i^{(t-1)}$
  - Treat $l\left(Y_i, \hat{Y}_i^{(t-1)}\right)$ as a constant term
  - Example (MSE loss)

$$\left(Y_i - (\hat{Y}_i^{(t-1)} + f^t(X_i))\right)^2 = \left(Y_i - \hat{Y}_i^{(t-1)}\right)^2 + 2\left(\hat{Y}_i^{(t-1)} - Y_i\right) f^t(X_i) + [f^t(X_i)]^2$$

    - $g_i = 2\left(\hat{Y}_i^{(t-1)} - Y_i\right)$ and $h_i = 2$

The structure score (optional)

- Objective function at step t

$$obj^{(t)} = \sum_{i=1}^{n} l\left(Y_i, \hat{Y}_i^{(t)}\right) + \sum_{b=1}^{t} \omega\left(f^b\right)$$

$$= \sum_{i=1}^{n} l\left(Y_i, \hat{Y}_i^{(t-1)}\right) + \sum_{i=1}^{n} \left[g_i f^t(X_i) + \frac{1}{2} h_i [f^t(X_i)]^2\right] + \gamma T + \frac{1}{2}\zeta \sum_{j=1}^{T} w_j^2$$

$$= \sum_{i=1}^{n} \left[g_i w_{q(X_i)} + \frac{1}{2} h_i \left[w_{q(X_i)}\right]^2\right] + \gamma T + \frac{1}{2}\zeta \sum_{j=1}^{T} w_j^2 + C$$ Replace $f^t(X_i)$ by $w_{q(X_i)}$

$$= \sum_{j=1}^{T} \left[\left(\underbrace{\sum_{i \in I_j} g_i}_{G_j}\right) w_j + \frac{1}{2}\left(\underbrace{\sum_{i \in I_j} h_i}_{H_j} + \zeta\right)\left[w_j\right]^2\right] + \gamma T + C$$ Change the sum by leaves

$$= \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2}(H_j + \zeta)\left[w_j\right]^2\right] + \gamma T + C$$

$$C = \sum_{i=1}^{n} l\left(Y_i, \hat{Y}_i^{(t-1)}\right)$$

- The best w_j to minimize $obj^{(t)}$ is given by $w_j^* = -\frac{G_j}{H_j + \zeta}$