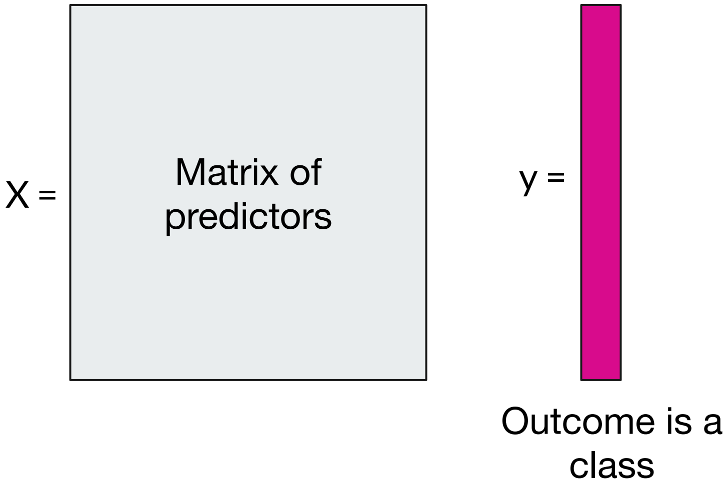
Chapter 14: Introduction to Boosting

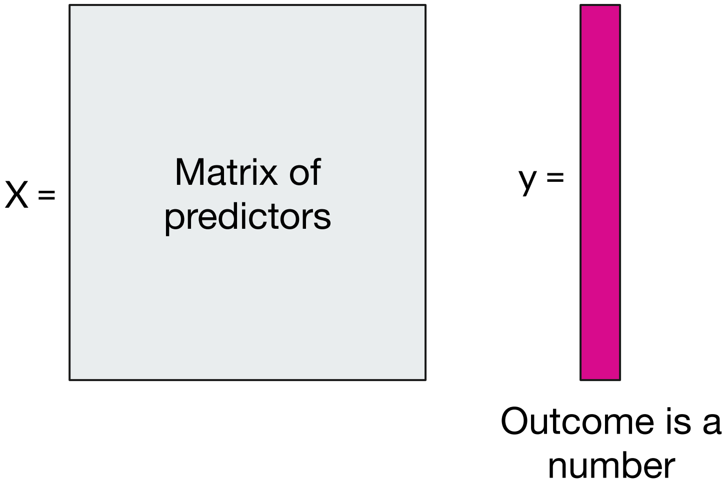
Modern Clinical Data Science
Chapter Guides
Bethany Percha, Instructor

How to Use this Guide

- Read the corresponding notes chapter first
- Try to answer the discussion questions on your own
- Listen to the chapter guide (should be 30 min, max) while following along in the notes



Classification

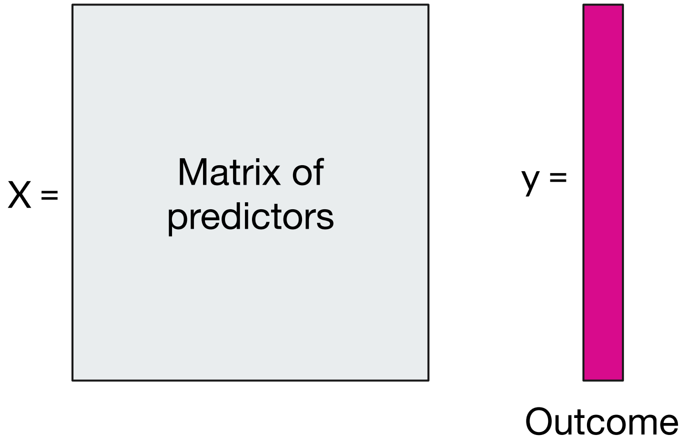


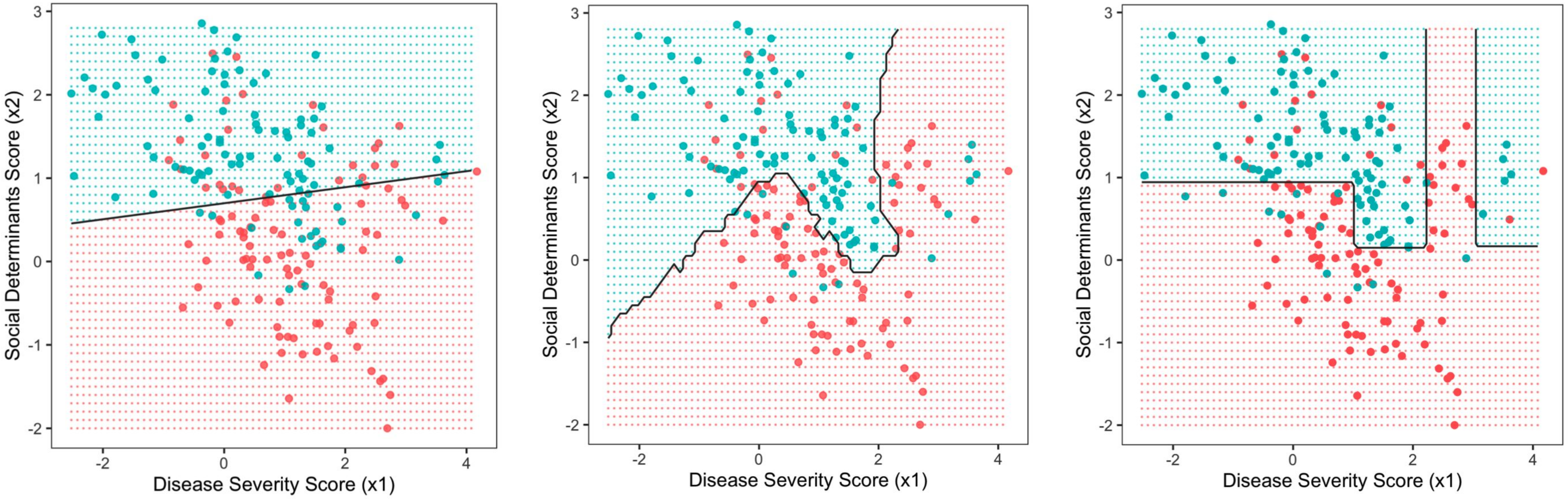
Regression

So far…

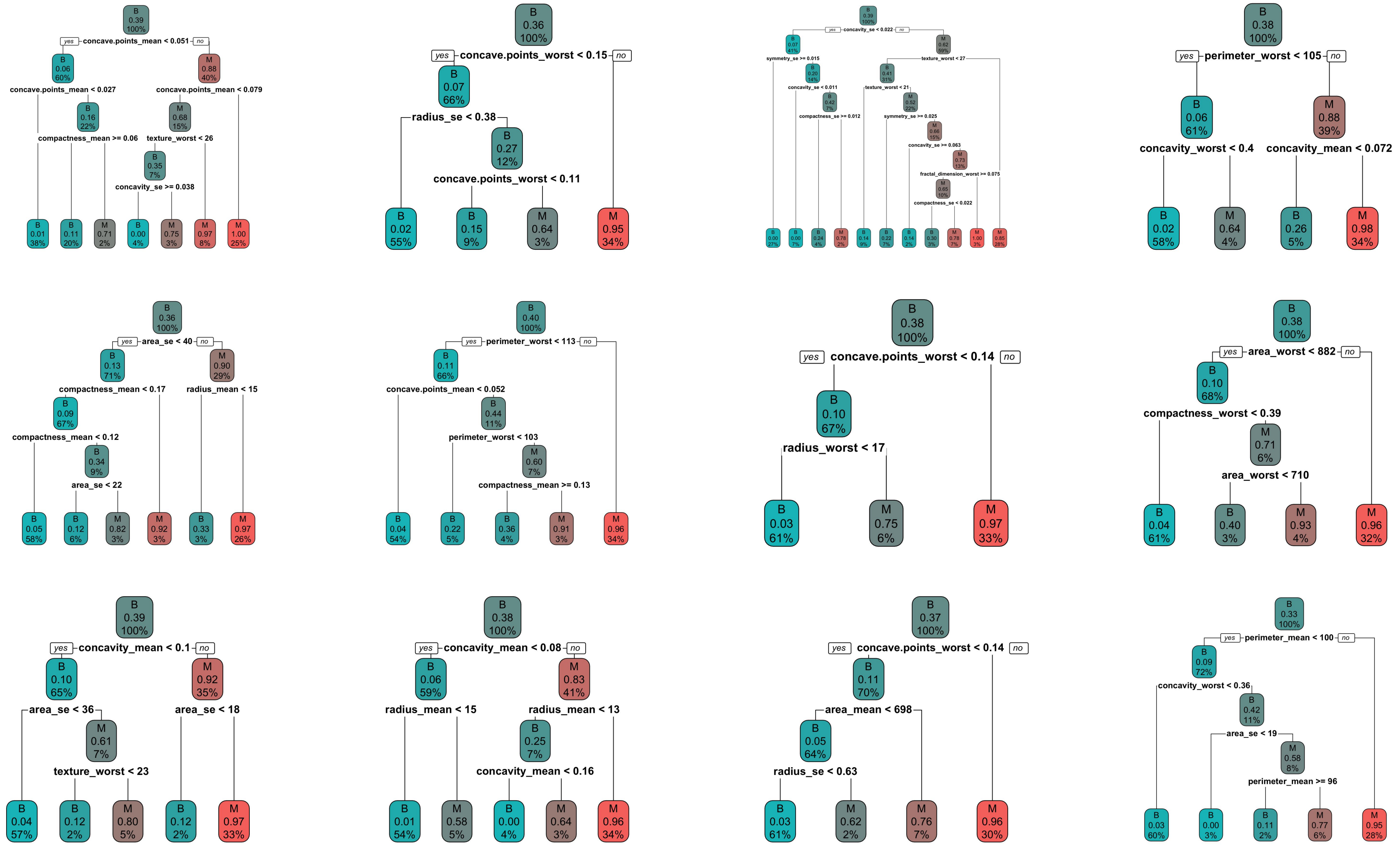
- Regression
- K-Nearest Neighbors (KNN)
- Decision trees
- Random forests

What are their stories?




Social Determinants Score (x2)
Disease Severity Score (x1)
Social Determinants Score (x2)
Disease Severity Score (x1)
Social Determinants Score (x2)
Disease Severity Score (x1)

Ensemble learning:
Combining multiple models that are only slightly better than random (weak learners) can produce a good model (strong learner).


B 0.39 100%
yes concave.points_mean < 0.051 no
B 0.06 60%
M 0.88 40%
concave.points_mean < 0.027
concave.points_mean < 0.079
B 0.16 22%
M 0.68 15%
compactness_mean >= 0.06
texture_worst < 26
B 0.35 7%
concavity_se >= 0.038
B 0.01 38%
B 0.11 20%
M 0.71 2%
B 0.00 4%
M 0.75 3%
M 0.97 8%
M 1.00 25%
B 0.36 100%
yes concave.points_worst < 0.15 no
B 0.07 66%
radius_se < 0.38
B 0.27 12%
concave.points_worst < 0.11
B 0.02 55%
B 0.15 9%
M 0.64 3%
M 0.95 34%
B 0.39 100%
yes concavity_se < 0.022 no
B 0.07 41%
M 0.62 59%
symmetry_se >= 0.015
texture_worst < 27
B 0.20 14%
B 0.41 31%
concavity_se < 0.011
texture_worst < 21
B 0.42 7%
M 0.52 22%
compactness_se >= 0.012
symmetry_se >= 0.025
M 0.66 15%
concavity_se >= 0.063
M 0.73 13%
fractal_dimension_worst >= 0.075
M 0.65 10%
compactness_se < 0.022
B 0.00 27%
B 0.00 7%
B 0.24 4%
M 0.78 2%
B 0.14 9%
B 0.22 7%
B 0.14 2%
B 0.30 3%
M 0.78 7%
M 1.00 3%
M 0.85 28%
B 0.38 100%
yes perimeter_worst < 105 no
B 0.06 61%
M 0.88 39%
concavity_worst < 0.4
concavity_mean < 0.072
B 0.02 58%
M 0.64 4%
B 0.26 5%
M 0.98 34%
B 0.36 100%
yes area_se < 40 no
B 0.13 71%
M 0.90 29%
compactness_mean < 0.17
radius_mean < 15
B 0.09 67%
compactness_mean < 0.12
B 0.34 9%
area_se < 22
B 0.05 58%
B 0.12 6%
M 0.82 3%
M 0.92 3%
B 0.33 3%
M 0.97 26%
B 0.40 100%
yes perimeter_worst < 113 no
B 0.11 66%
concave.points_mean < 0.052
B 0.44 11%
perimeter_worst < 103
M 0.60 7%
compactness_mean >= 0.13
B 0.04 54%
B 0.22 5%
B 0.36 4%
M 0.91 3%
M 0.96 34%
B 0.38 100%
yes concave.points_worst < 0.14 no
B 0.10 67%
radius_worst < 17
B 0.03 61%
M 0.75 6%
M 0.97 33%
B 0.38 100%
yes area_worst < 882 no
B 0.10 68%
compactness_worst < 0.39
M 0.71 6%
area_worst < 710
B 0.04 61%
B 0.40 3%
M 0.93 4%
M 0.96 32%
B 0.39 100%
yes concavity_mean < 0.1 no
B 0.10 65%
M 0.92 35%
area_se < 36
area_se < 18
M 0.61 7%
texture_worst < 23
B 0.04 57%
B 0.12 2%
M 0.80 5%
B 0.12 2%
M 0.97 33%
B 0.38 100%
yes concavity_mean < 0.08 no
B 0.06 59%
M 0.83 41%
radius_mean < 15
radius_mean < 13
B 0.25 7%
concavity_mean < 0.16
B 0.01 54%
M 0.58 5%
B 0.00 4%
M 0.64 3%
M 0.96 34%
B 0.37 100%
yes concave.points_worst < 0.14 no
B 0.11 70%
area_mean < 698
B 0.05 64%
radius_se < 0.63
B 0.03 61%
M 0.62 2%
M 0.76 7%
M 0.96 30%
B 0.33 100%
yes perimeter_mean < 100 no
B 0.09 72%
concavity_worst < 0.36
B 0.42 11%
area_se < 19
M 0.58 8%
perimeter_mean >= 96
B 0.03 60%
B 0.00 3%
B 0.11 2%
M 0.77 6%
M 0.95 28%

Boosting:
A principled way of creating weak learners whose combined predictions get better over time.

AdaBoost

First practical boosting algorithm. For a long time, no one knew why it worked.

1. Initialize the observation weights to $w_i^{(1)} = \frac{1}{N}$ for $i = 1, \ldots, N$.
2. For $m = 1, \ldots, M$:
   (a) Select a classifier, $G_m(x)$, that minimizes the weighted training error according to the current set of weights, $w_i^{(m)}$. Depending on the algorithm, it may be possible to train a single classifier on the weighted training set; in other cases, one may need to select the best-performing classifier from among a predefined set.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^N w_i^{(m)} \cdot \mathcal{I}(y^{(i)} \neq G_m(x^{(i)}))}{\sum_{i=1}^N w_i^{(m)}}$$
   (c) Compute voting weight for classifier m:
   $$\alpha_m = \log\left(\frac{1 - \text{err}_m}{\text{err}_m}\right)$$
   (d) Set
   $$w_i^{(m+1)} := w_i^{(m)} \cdot \exp\left[\alpha_m \cdot \mathcal{I}(y^{(i)} \neq G_m(x^{(i)}))\right]$$
   for $i = 1, \ldots, N$.
3. Output
   $$G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$$

Subject ID	friends (X_1)	money (X_2)	free time (X_3)	pet (X_4)	happy (Y)
1	1	1	0	0	-1
2	1	1	1	0	-1
3	0	1	1	0	-1
4	0	0	0	0	-1
5	1	0	0	0	-1
6	0	0	0	0	-1
7	1	2	1	0	1
8	1	0	1	0	1
9	0	0	1	1	1
10	1	0	0	1	1

$$X_1 = \begin{cases} 0 & \text{no friends} \\ 1 & \text{friends} \end{cases} \qquad X_2 = \begin{cases} 0 & \text{poor} \\ 1 & \text{enough money} \\ 2 & \text{rich} \end{cases}$$

$$X_3 = \begin{cases} 0 & \text{no free time} \\ 1 & \text{some free time} \end{cases} \qquad X_4 = \begin{cases} 0 & \text{no pet} \\ 1 & \text{has a pet} \end{cases}$$

Now, let's go through the process of applying AdaBoost to this dataset, step by step.

(a) Initialize the observation weights for the training data. (Make them uniform.)

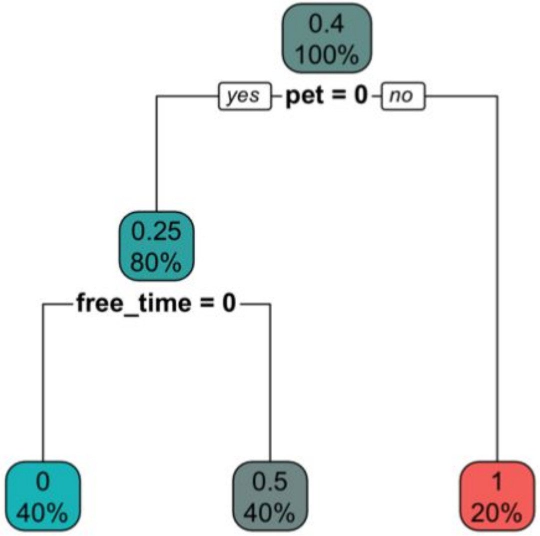
$$w_1^{(1)} = \qquad\qquad w_6^{(1)} =$$
$$w_2^{(1)} = \qquad\qquad w_7^{(1)} =$$
$$w_3^{(1)} = \qquad\qquad w_8^{(1)} =$$
$$w_4^{(1)} = \qquad\qquad w_9^{(1)} =$$
$$w_5^{(1)} = \qquad\qquad w_{10}^{(1)} =$$

Now, let's go through the process of applying AdaBoost to this dataset, step by step.

(a) Initialize the observation weights for the training data. (Make them uniform.)

$$
\begin{aligned}
w_1^{(1)} &= 0.1 & \quad w_6^{(1)} &= 0.1 \\
w_2^{(1)} &= 0.1 & \quad w_7^{(1)} &= 0.1 \\
w_3^{(1)} &= 0.1 & \quad w_8^{(1)} &= 0.1 \\
w_4^{(1)} &= 0.1 & \quad w_9^{(1)} &= 0.1 \\
w_5^{(1)} &= 0.1 & \quad w_{10}^{(1)} &= 0.1
\end{aligned}
$$

(b) We will now grow a decision tree on this dataset, $G_1(x)$, using these weights. We'll use the `rpart` package in R, just as in Section 7.3. Here is the first tree.



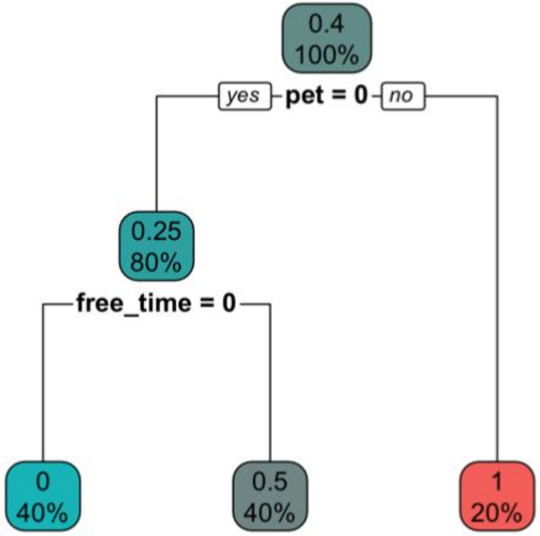
Datapoint ID	$w_i^{(1)}$	happy (Y)	$G_1(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	-1
6	0.1	-1	-1
7	0.1	1	$\boxed{-1}$
8	0.1	1	$\boxed{-1}$
9	0.1	1	1
10	0.1	1	1

Compute the misclassification error of this tree, err_1. Compare this tree to the one we constructed by hand in Chapter 7.

$$\mathrm{err}_1 =$$

(b) We will now grow a decision tree on this dataset, $G_1(x)$, using these weights. We'll use the `rpart` package in R, just as in Section 7.3. Here is the first tree.

Datapoint ID	$w_i^{(1)}$	happy (Y)	$G_1(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	-1
6	0.1	-1	-1
7	0.1	1	-1
8	0.1	1	-1
9	0.1	1	1
10	0.1	1	1



Compute the misclassification error of this tree, err_1. Compare this tree to the one we constructed by hand in Chapter 7.

$$\text{err}_1 = \frac{0.1(2)}{0.1(10)} = 0.2$$

(c) Based on how $G_1(x)$ performs, calculate α_1, its voting weight.

$$\alpha_1 =$$

(d) Re-weight the observation weights for the training examples.

$$w_1^{(2)} =$$
$$w_2^{(2)} =$$
$$w_3^{(2)} =$$
$$w_4^{(2)} =$$
$$w_5^{(2)} =$$

$$w_6^{(2)} =$$
$$w_7^{(2)} =$$
$$w_8^{(2)} =$$
$$w_9^{(2)} =$$
$$w_{10}^{(2)} =$$

(c) Based on how $G_1(x)$ performs, calculate α_1, its voting weight.

$$\alpha_1 = \log\left(\frac{1 - \text{err}_1}{\text{err}_1}\right) = 1.386$$

(d) Re-weight the observation weights for the training examples.

$w_1^{(2)} =$ 0.1

$w_2^{(2)} =$ 0.1

$w_3^{(2)} =$ 0.1

$w_4^{(2)} =$ 0.1

$w_5^{(2)} =$ 0.1

$w_6^{(2)} =$ 0.1

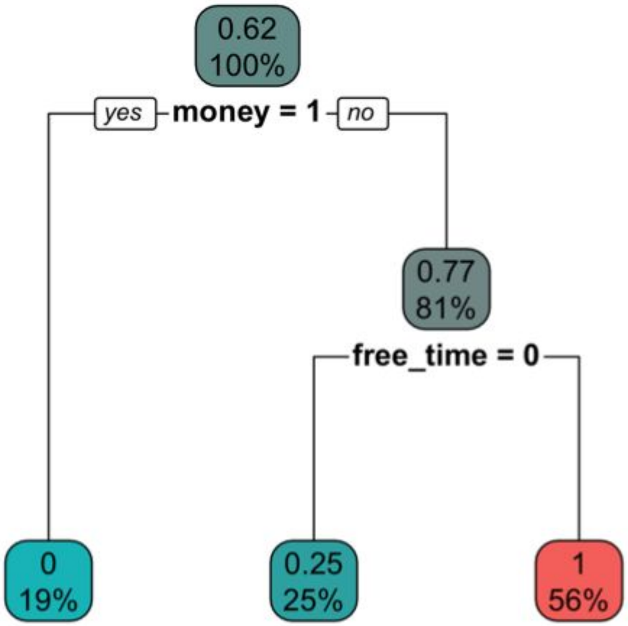
$w_7^{(2)} =$ 0.1 * exp(1.386) = 0.4

$w_8^{(2)} =$ 0.1 * exp(1.386) = 0.4

$w_9^{(2)} =$ 0.1

$w_{10}^{(2)} =$ 0.1

(e) Now we grow another decision tree, $G_2(x)$, using these new weights as inputs to the `rpart` package.

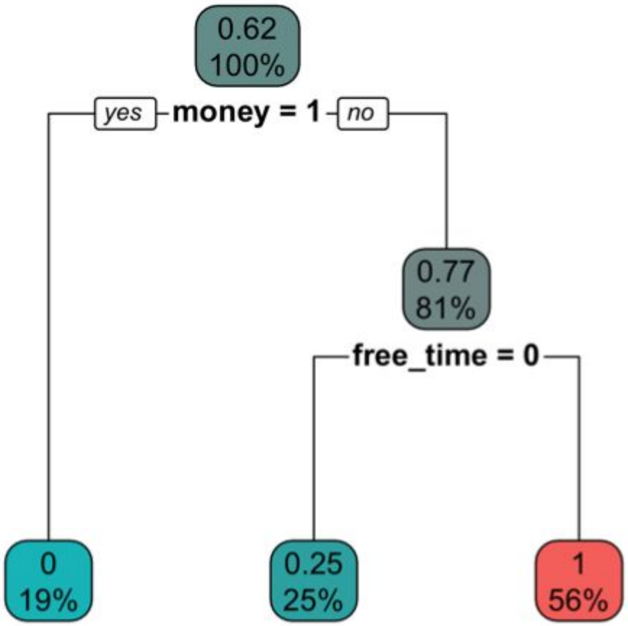


Datapoint ID	$w_i^{(2)}$	happy (Y)	$G_2(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	-1
6	0.1	-1	-1
7	0.4	1	1
8	0.4	1	1
9	0.1	1	1
10	0.1	1	$\boxed{-1}$

Compute the misclassification error of this tree, err_2.

$\text{err}_2 =$

(e) Now we grow another decision tree, $G_2(x)$, using these new weights as inputs to the `rpart` package.



Datapoint ID	$w_i^{(2)}$	happy (Y)	$G_2(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	-1
6	0.1	-1	-1
7	0.4	1	1
8	0.4	1	1
9	0.1	1	1
10	0.1	1	-1

Compute the misclassification error of this tree, err_2.

$$\text{err}_2 = \frac{0.1}{0.1(8) + 0.4(2)} = 0.0625$$

(f) Based on how $G_2(x)$ performs, calculate α_2, its voting weight.

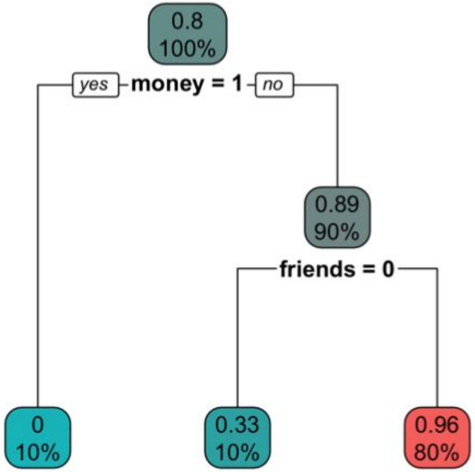
$$\alpha_2 = \log\left(\frac{1 - \text{err}_2}{\text{err}_2}\right) = 2.708$$

(g) Re-weight the observation weights for the training examples.

$$w_1^{(3)} = 0.1$$
$$w_2^{(3)} = 0.1$$
$$w_3^{(3)} = 0.1$$
$$w_4^{(3)} = 0.1$$
$$w_5^{(3)} = 0.1$$

$$w_6^{(3)} = 0.1$$
$$w_7^{(3)} = 0.4$$
$$w_8^{(3)} = 0.4$$
$$w_9^{(3)} = 0.1$$
$$w_{10}^{(3)} = 0.1*\exp(2.708) = 1.5$$

(h) Now we grow another decision tree, $G_3(x)$, using these new weights as inputs to the `rpart` package.



Datapoint ID	$w_i^{(3)}$	happy (Y)	$G_3(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	$\boxed{1}$
6	0.1	-1	-1
7	0.4	1	1
8	0.4	1	1
9	0.1	1	$\boxed{-1}$
10	1.5	1	1

(i) Compute the misclassification error of this tree, err_3.

$$\text{err}_3 =$$

$$\frac{0.2}{0.1(7) + 0.4(2) + 1.5} = 0.067$$

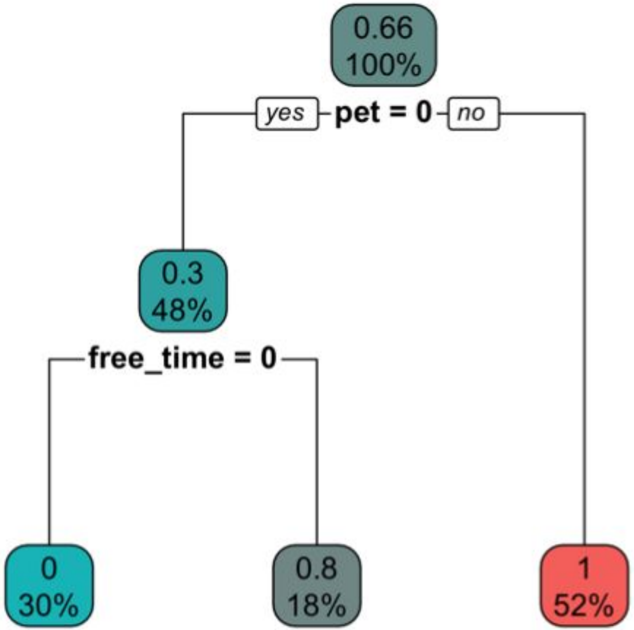
(j) Based on how $G_3(x)$ performs, calculate α_3, its voting weight.

$$\alpha_3 = \log\left(\frac{1 - \mathrm{err}_3}{\mathrm{err}_3}\right) = 2.639$$

(k) Re-weight the observation weights for the training examples.

$$w_1^{(4)} = 0.1$$
$$w_2^{(4)} = 0.1$$
$$w_3^{(4)} = 0.1$$
$$w_4^{(4)} = 0.1$$
$$w_5^{(4)} = 0.1*\exp(2.639) = 1.4$$
$$w_6^{(4)} = 0.1$$
$$w_7^{(4)} = 0.4$$
$$w_8^{(4)} = 0.4$$
$$w_9^{(4)} = 0.1*\exp(2.639) = 1.4$$
$$w_{10}^{(4)} = 1.5$$

(l) Now we grow another decision tree, $G_4(x)$, using these new weights as inputs to the `rpart` package.



Datapoint ID	$w_i^{(4)}$	happy (Y)	$G_4(x)$
1	0.1	-1	-1
2	0.1	-1	1
3	0.1	-1	1
4	0.1	-1	-1
5	1.4	-1	-1
6	0.1	-1	-1
7	0.4	1	1
8	0.4	1	1
9	1.4	1	1
10	1.5	1	1

(m) Compute the misclassification error of this tree, err_4.

$$\text{err}_4 = \frac{0.2}{0.1(5) + 0.4(2) + 1.4(2) + 1.5} = 0.036$$

(n) Based on how $G_4(x)$ performs, calculate α_4, its voting weight.

$$\alpha_4 = \log\left(\frac{1-\text{err}_4}{\text{err}_4}\right) = 3.296$$

(o) Output the weighted average of the four classifiers' votes for each training example:

$$G(x) = \text{sign}\left[\alpha_1 G_1(x) + \alpha_2 G_2(x) + \alpha_3 G_3(x) + \alpha_4 G_4(x)\right]$$

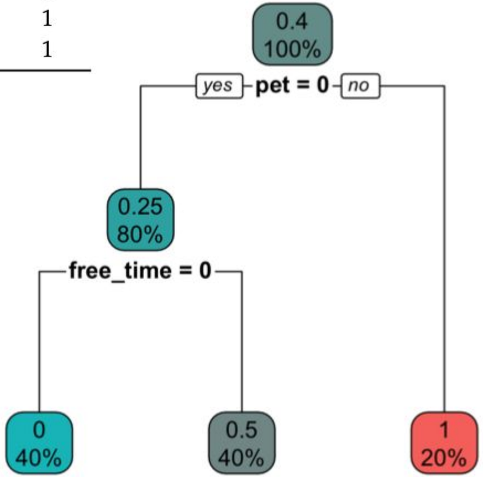
What is the final training error?

		1.386	2.708	2.639	3.296	
Datapoint ID	happy (Y)	$G_1(x)$	$G_2(x)$	$G_3(x)$	$G_4(x)$	$G(x)$
1	-1	-1	-1	-1	-1	
2	-1	-1	-1	-1	1	
3	-1	-1	-1	-1	1	
4	-1	-1	-1	-1	-1	
5	-1	-1	-1	1	-1	
6	-1	-1	-1	-1	-1	
7	1	-1	1	1	1	
8	1	-1	1	1	1	
9	1	1	1	-1	1	
10	1	1	-1	1	1	

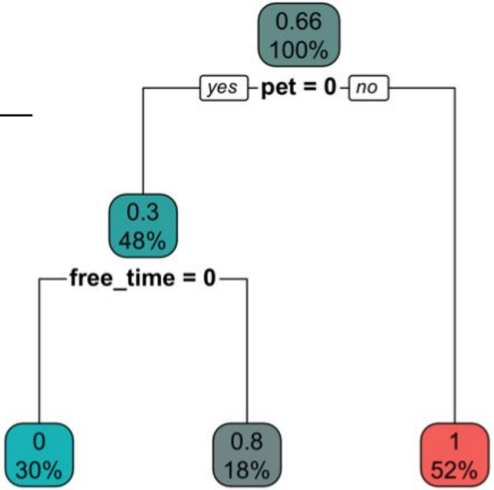
Question 14.1

The most difficult thing to understand in all of this is how the updated weights play into the construction of subsequent trees. A clue comes from the dataset percentages shown in the nodes of each tree. For example, trees 1 and 3 actually

Datapoint ID	$w_i^{(1)}$	happy (Y)	$G_1(x)$
1	0.1	-1	-1
2	0.1	-1	-1
3	0.1	-1	-1
4	0.1	-1	-1
5	0.1	-1	-1
6	0.1	-1	-1
7	0.1	1	-1
8	0.1	1	-1
9	0.1	1	1
10	0.1	1	1



Datapoint ID	$w_i^{(4)}$	happy (Y)	$G_4(x)$
1	0.1	-1	-1
2	0.1	-1	1
3	0.1	-1	1
4	0.1	-1	-1
5	1.4	-1	-1
6	0.1	-1	-1
7	0.4	1	1
8	0.4	1	1
9	1.4	1	1
10	1.5	1	1

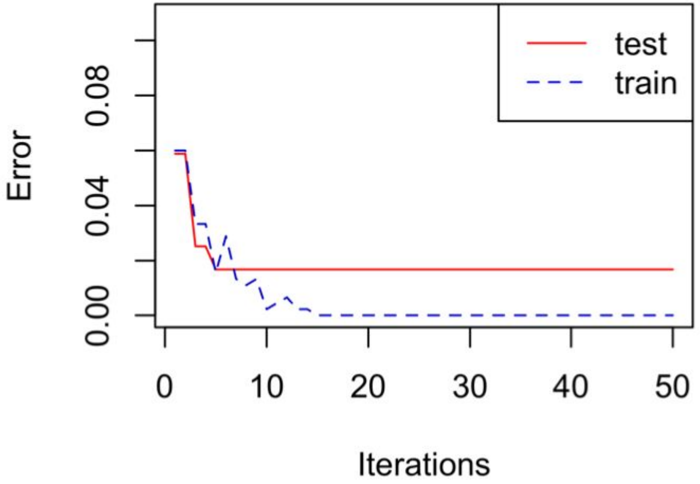


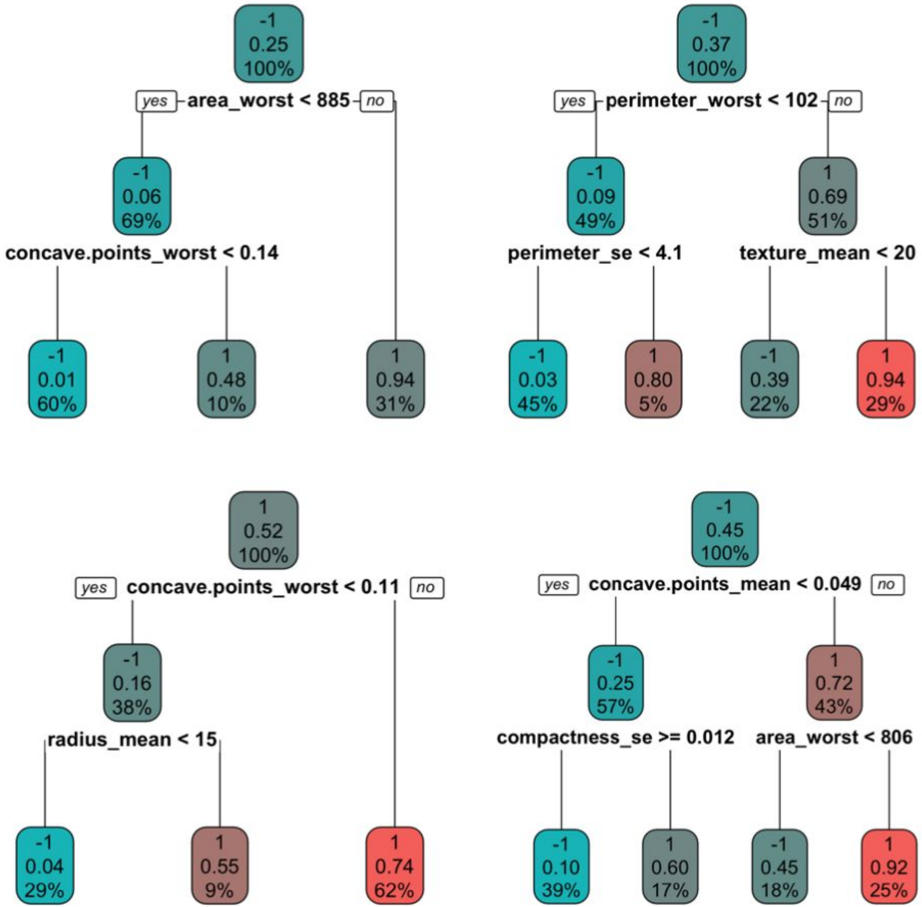
Gradient boosting:
A more general conception of boosting in which later models are fit to the errors (pseudo-residuals) of earlier models.

How does this guarantee diversity?

Wisconsin Breast Cancer dataset revisited

Ensemble error vs number of trees





Question 14.2

Compare and contrast boosting and random forests on the basis of:

- Whether each tree uses all or part of the dataset
- Whether they consider a subset of the predictors at each split or all the predictors
- Whether you can parallelize the construction of different trees (i.e., build them at the same time on different processors)
- Whether the votes of different classifiers are independent
- Ease of use and interpretability