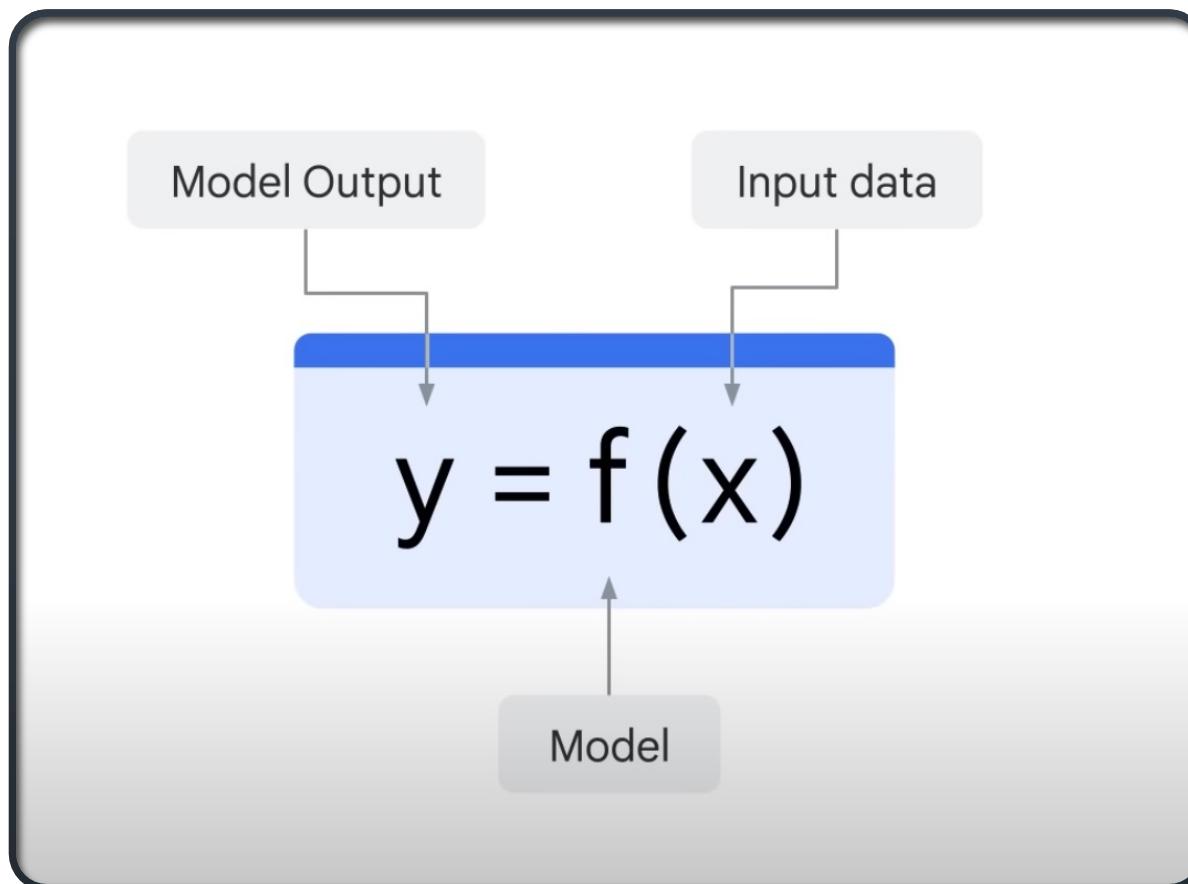


# Introduction to Supervised Machine Learning

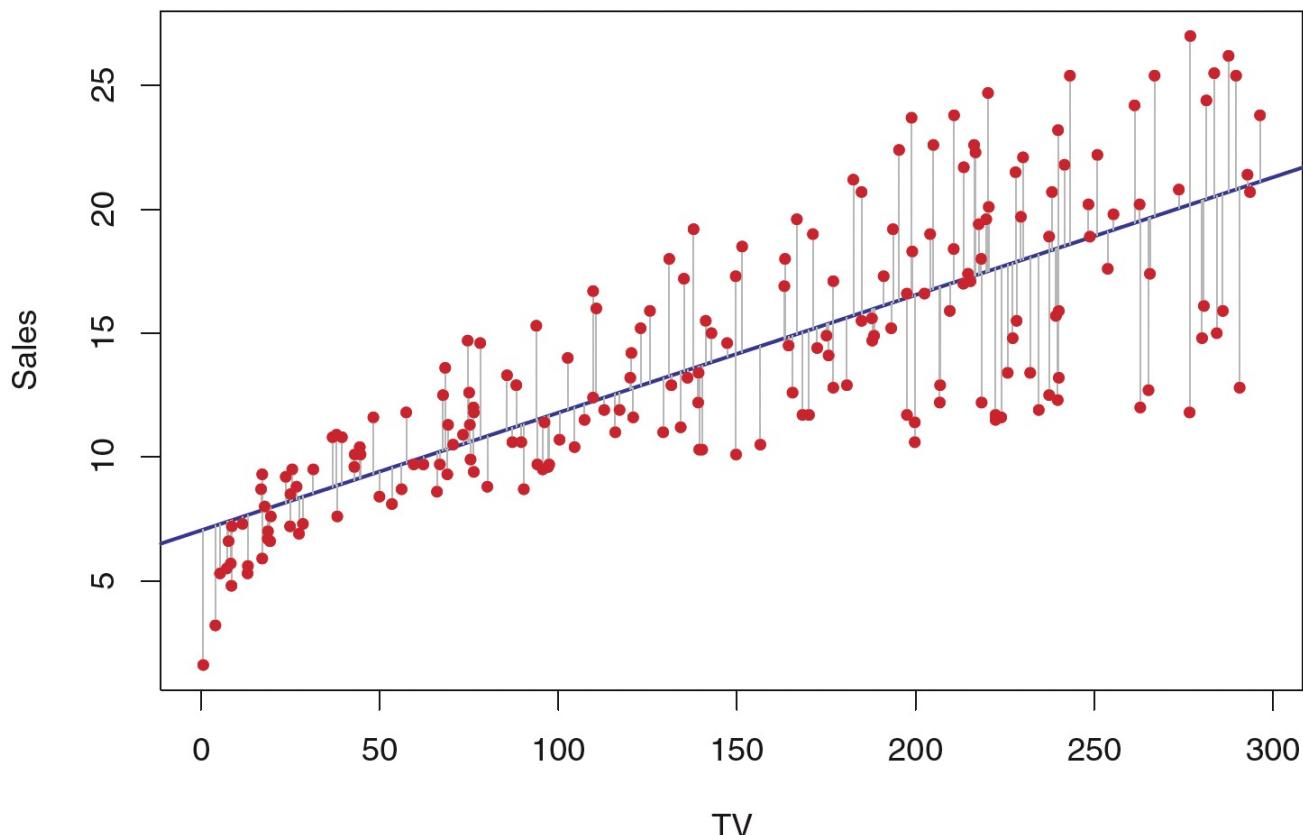
Soumya Banerjee

# Supervised learning



# Linear regression

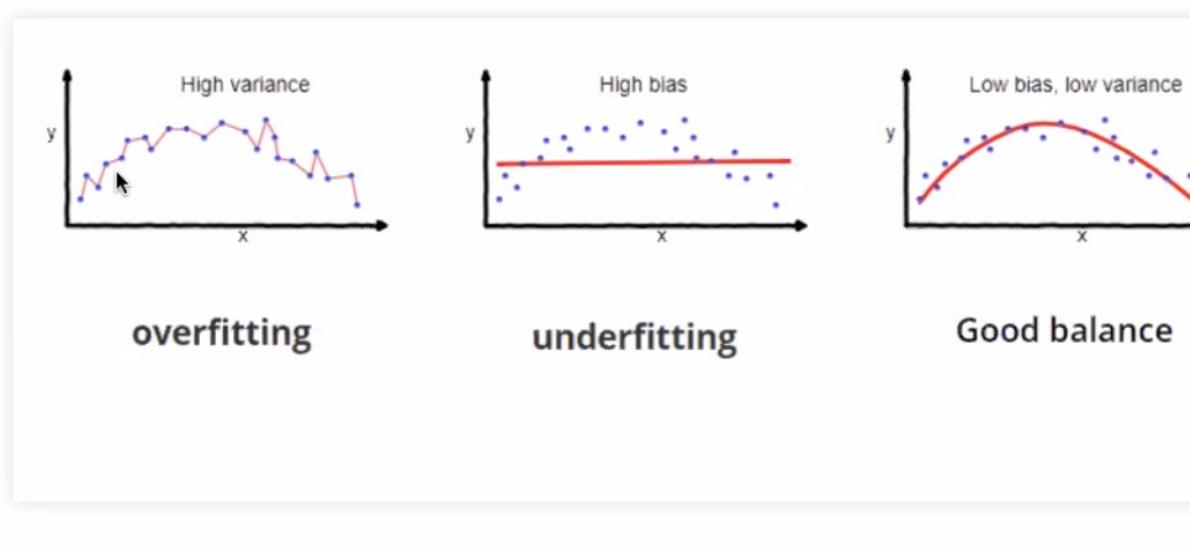
Limma analysis of PBMCs in microarray (LPS vs. LPS + anti-IL10R) across all donors



**FIGURE 3.1.** For the [Advertising](#) data, the least squares fit for the regression of **sales** onto **TV** is shown. The fit is found by minimizing the sum of squared errors. Each grey line segment represents an error, and the fit makes a compromise by averaging their squares. In this case a linear fit captures the essence of the relationship, although it is somewhat deficient in the left of the plot.

# A fundamental concept in machine learning

## Bias-variance tradeoff



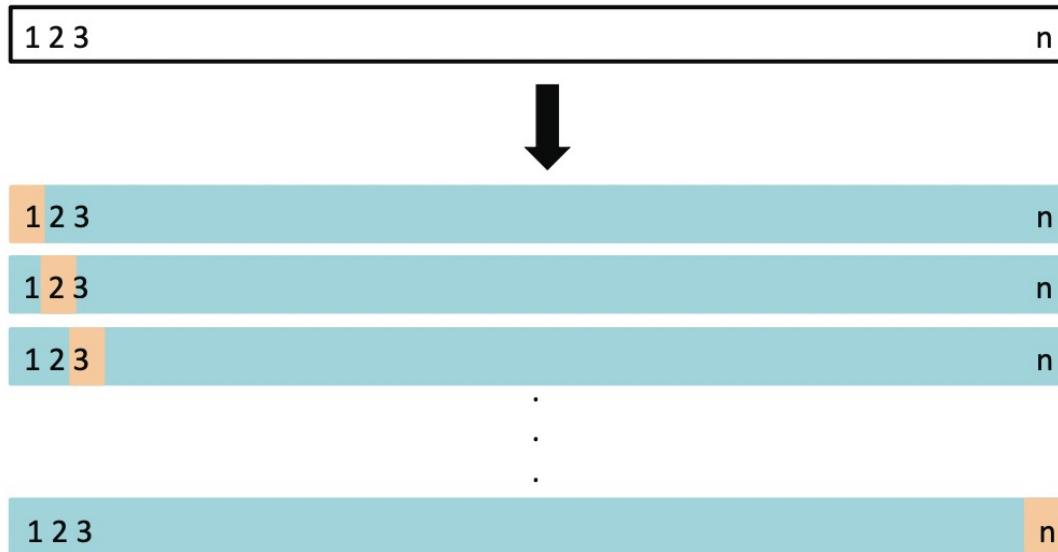
- Bias is residual error from fitting the Training data
- Variance is generalization error when applying the model fit to

# A fundamental concept in machine learning



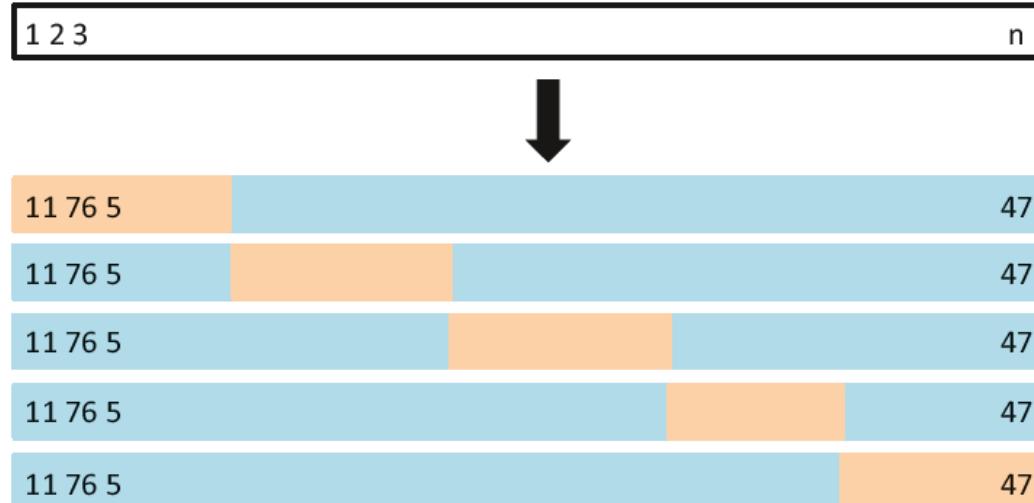
**FIGURE 5.1.** A schematic display of the validation set approach. A set of  $n$  observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.

# Leave one out cross-validation



**FIGURE 5.3.** A schematic display of LOOCV. A set of  $n$  data points is repeatedly split into a training set (shown in blue) containing all but one observation, and a validation set that contains only that observation (shown in beige). The test error is then estimated by averaging the  $n$  resulting MSEs. The first training set contains all but observation 1, the second training set contains all but observation 2, and so forth.

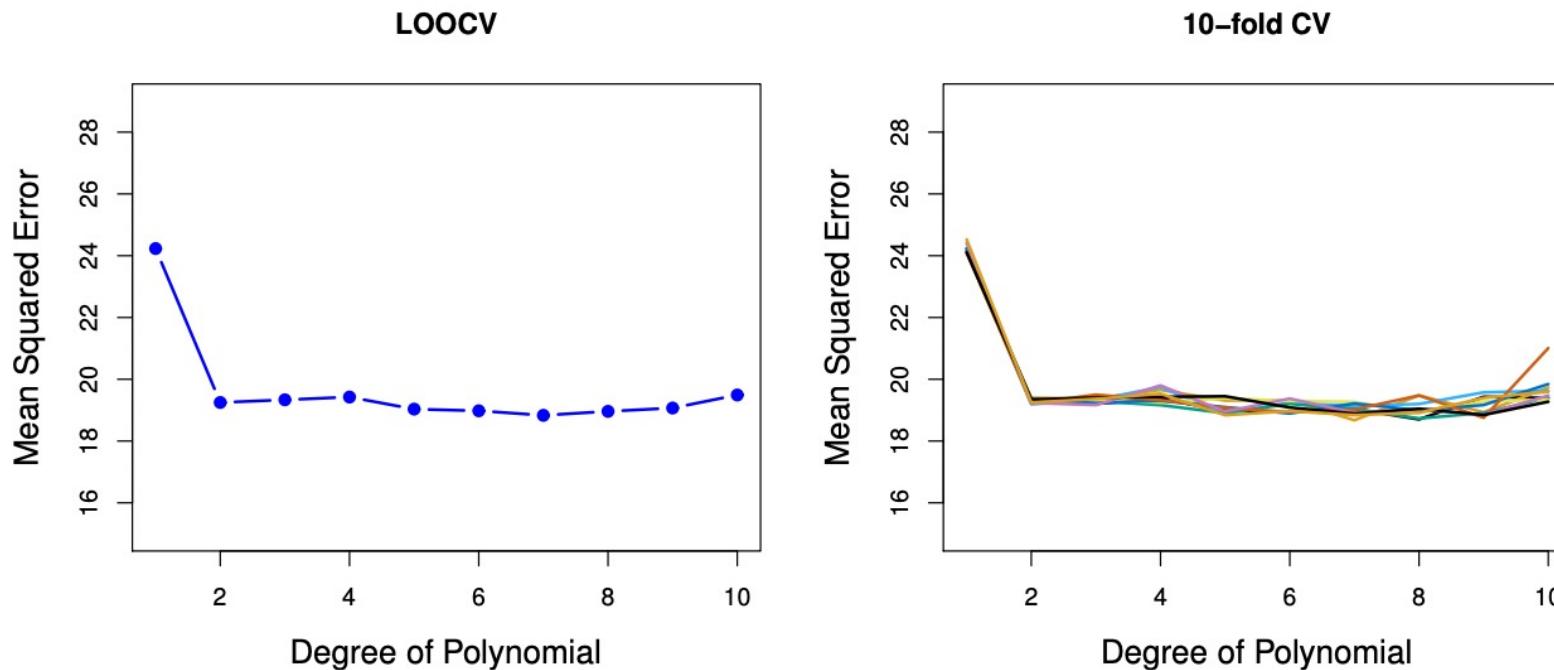
# K-fold cross-validation



**FIGURE 5.5.** A schematic display of 5-fold CV. A set of  $n$  observations is randomly split into five non-overlapping groups. Each of these fifths acts as a validation set (shown in beige), and the remainder as a training set (shown in blue). The test error is estimated by averaging the five resulting MSE estimates.

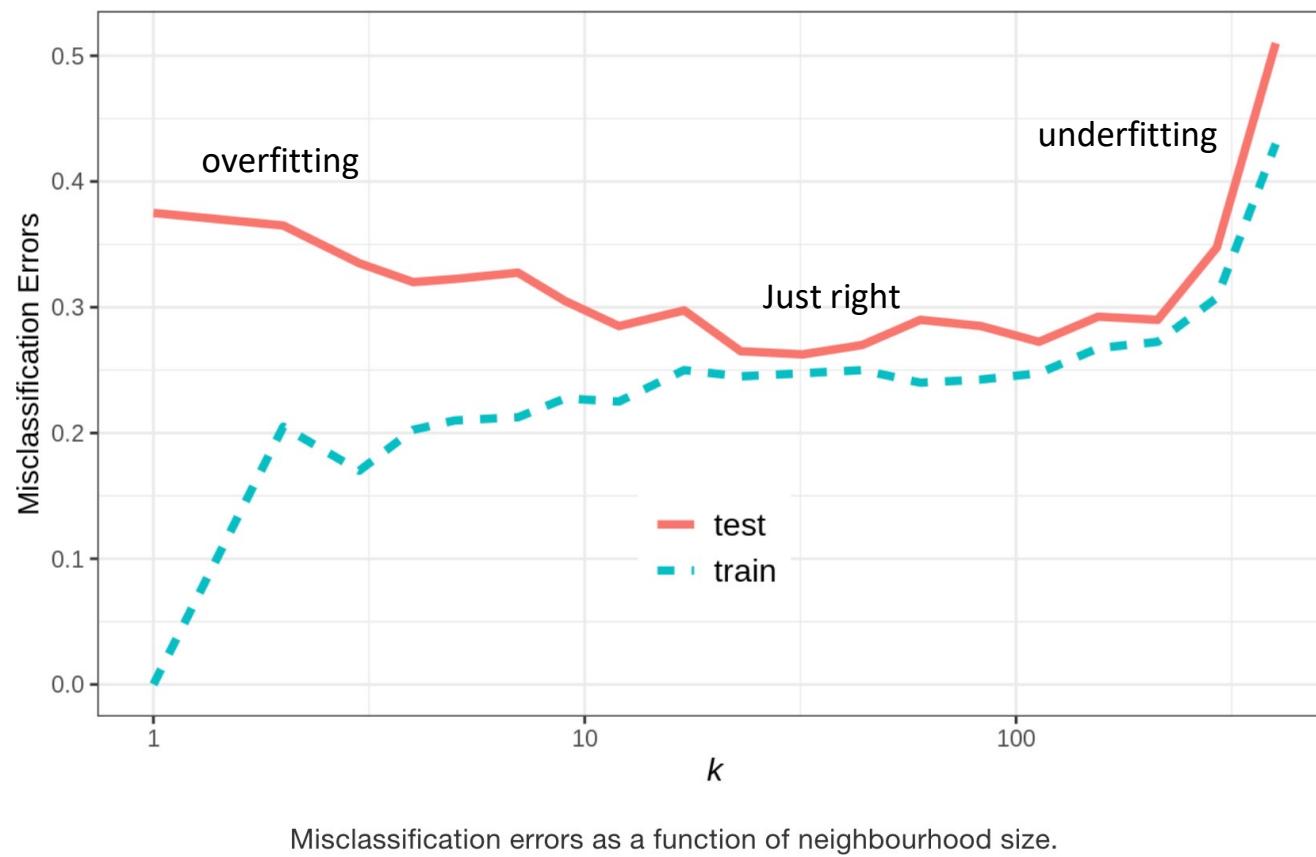
chapters. The magic formula (5.2) does not hold in general, in which case the model has to be refit  $n$  times.

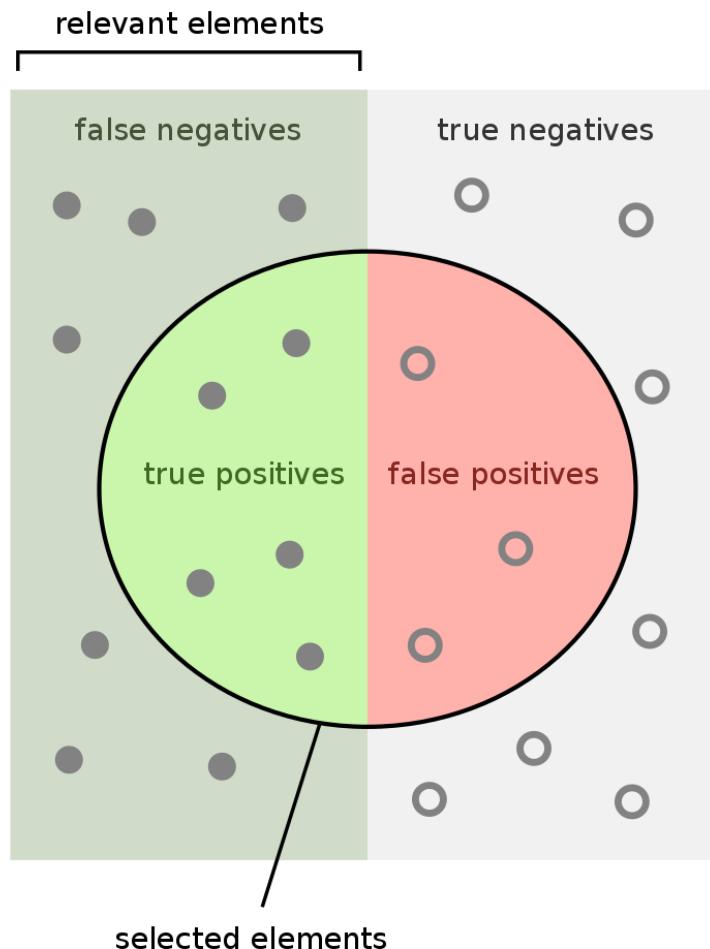
# Cross-validation



**FIGURE 5.4.** Cross-validation was used on the `Auto` data set in order to estimate the test error that results from predicting `mpg` using polynomial functions of `horsepower`. Left: The LOOCV error curve. Right: 10-fold CV was run nine separate times, each with a different random split of the data into ten parts. The figure shows the nine slightly different CV error curves.

# Cross-validation





How many relevant items are selected?  
e.g. How many sick people are correctly identified as having the condition.

$$\text{Sensitivity} = \frac{\text{true positives}}{\text{relevant elements}}$$



How many negative selected elements are truly negative?  
e.g. How many healthy people are identified as not having the condition.

$$\text{Specificity} = \frac{\text{true negatives}}{\text{not relevant elements}}$$



# Confusion matrix

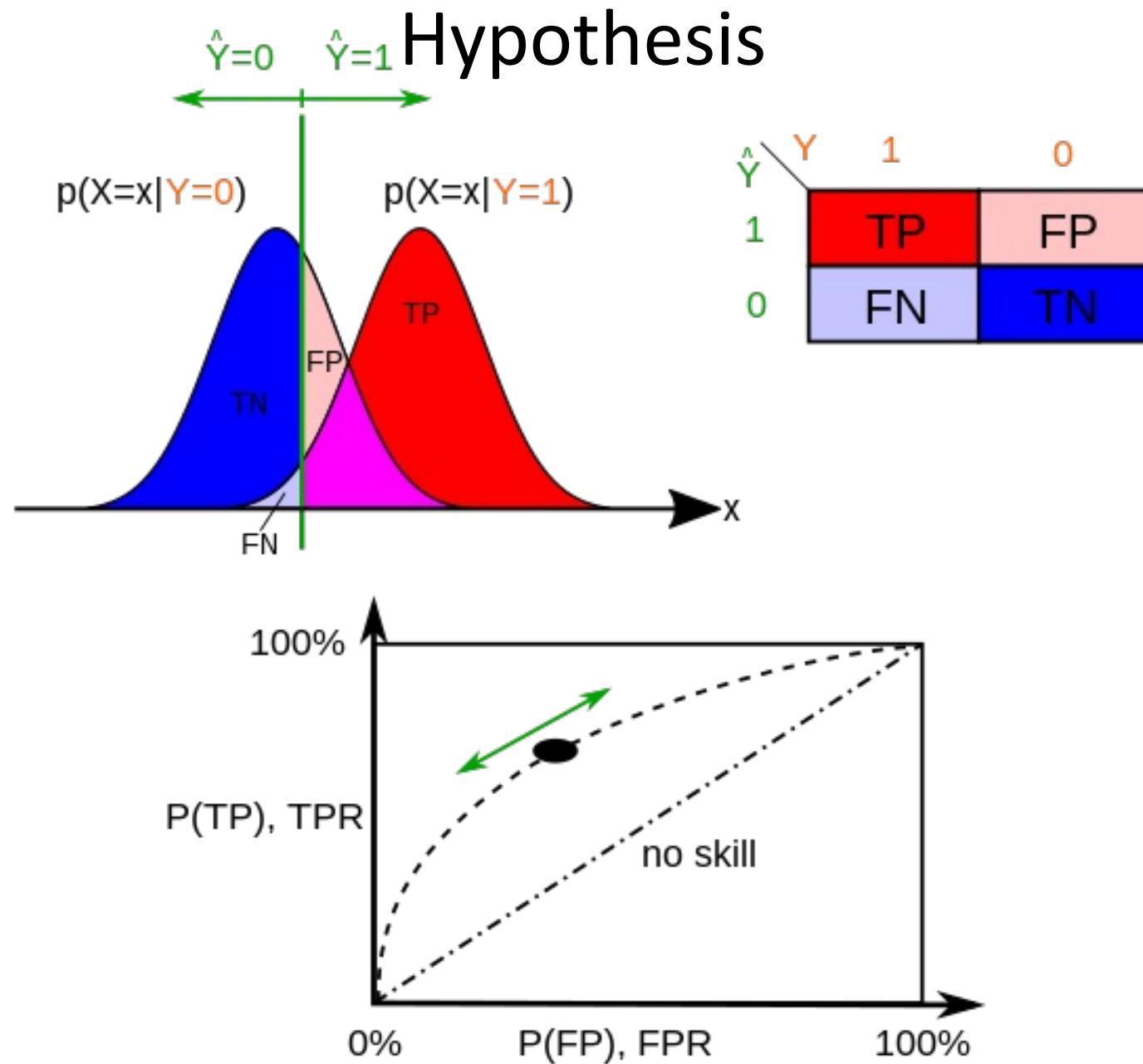
		ACTUAL VALUES	
		Positive	Negative
PREDICTED VALUES	Positive	TP	FP
	Negative	FN	TN

The predicted value is positive and its positive

Type I error : The predicted value is positive but it False

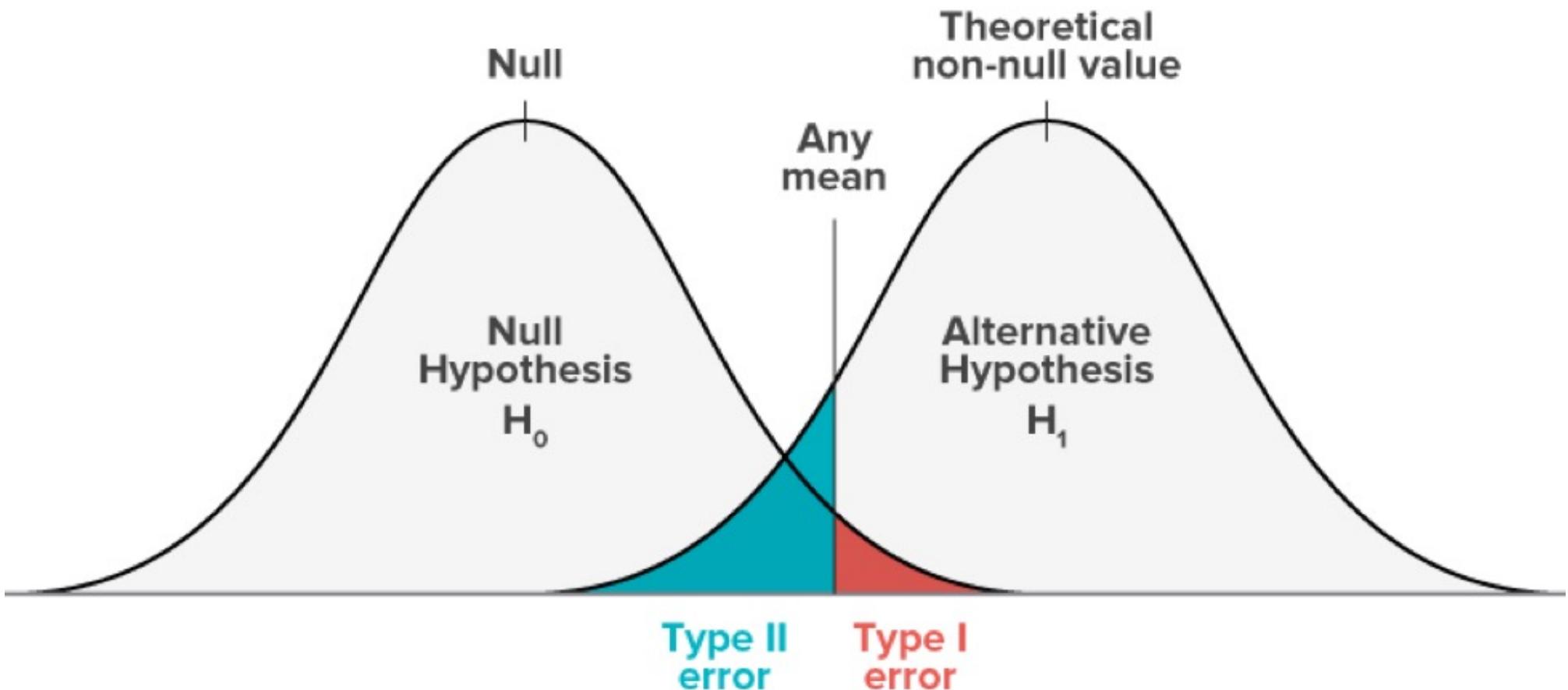
Type II error : The predicted value is negative but its positive

The predicted value is Negative and its Negative



The probability of a Type I error is the significance level (0.05)  
[https://en.wikipedia.org/wiki/Type\\_I\\_and\\_type\\_II\\_errors](https://en.wikipedia.org/wiki/Type_I_and_type_II_errors)

# Hypothesis



**The probability of a Type 1 error is the significance level (0.05)**

# Question

What happens if we use a simple classifier that just predicts randomly on a disease?

For example, say you are trying to predict a particular disease.

1. Assume you have features like age, gender and medications.
2. You have to predict whether this patient will get a disease or not.
3. Assume the disease occurs in 90% of the population.
  
4. How should you build your algorithm? What metric should you look at?

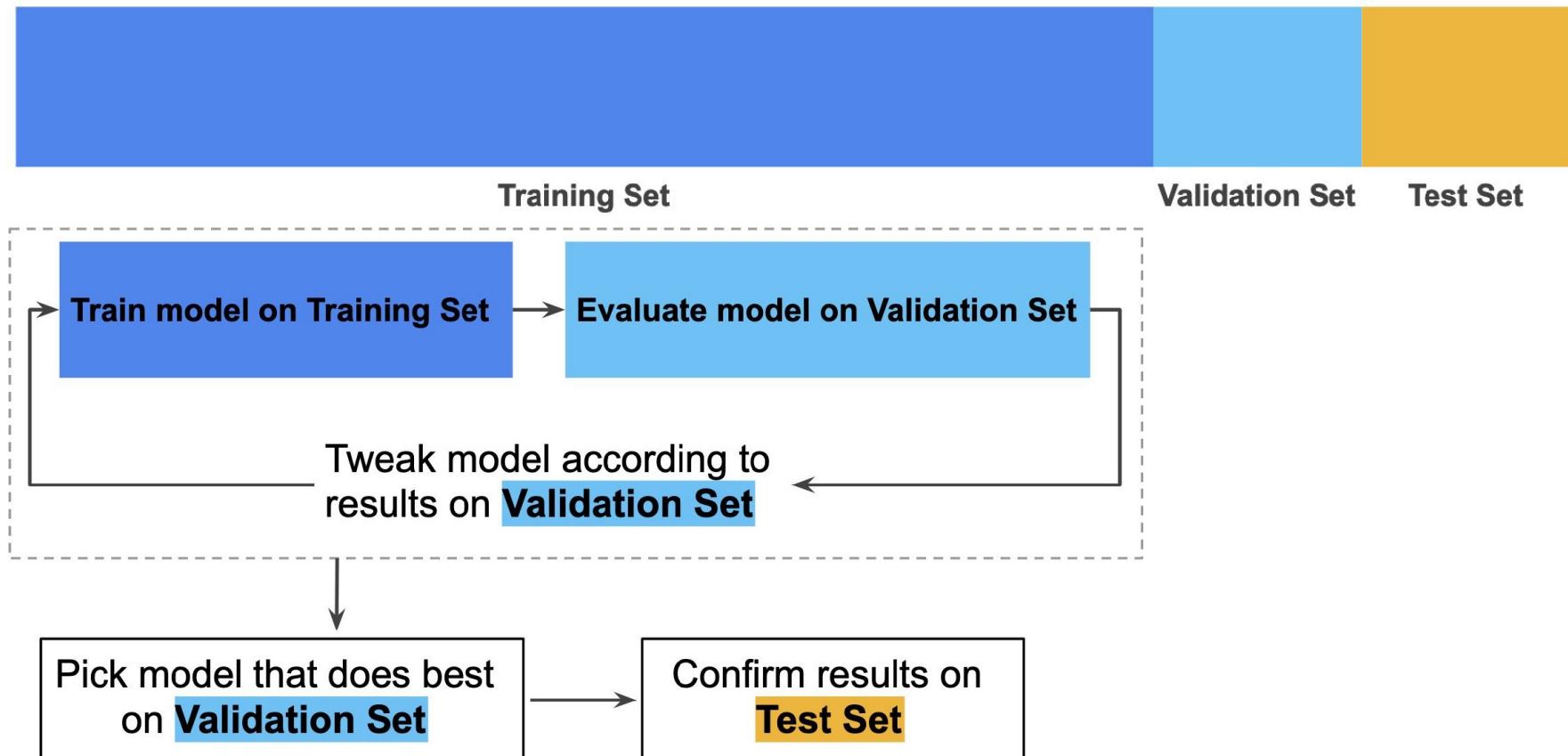
# Question

What happens if we use a simple classifier that just predicts randomly on a disease?

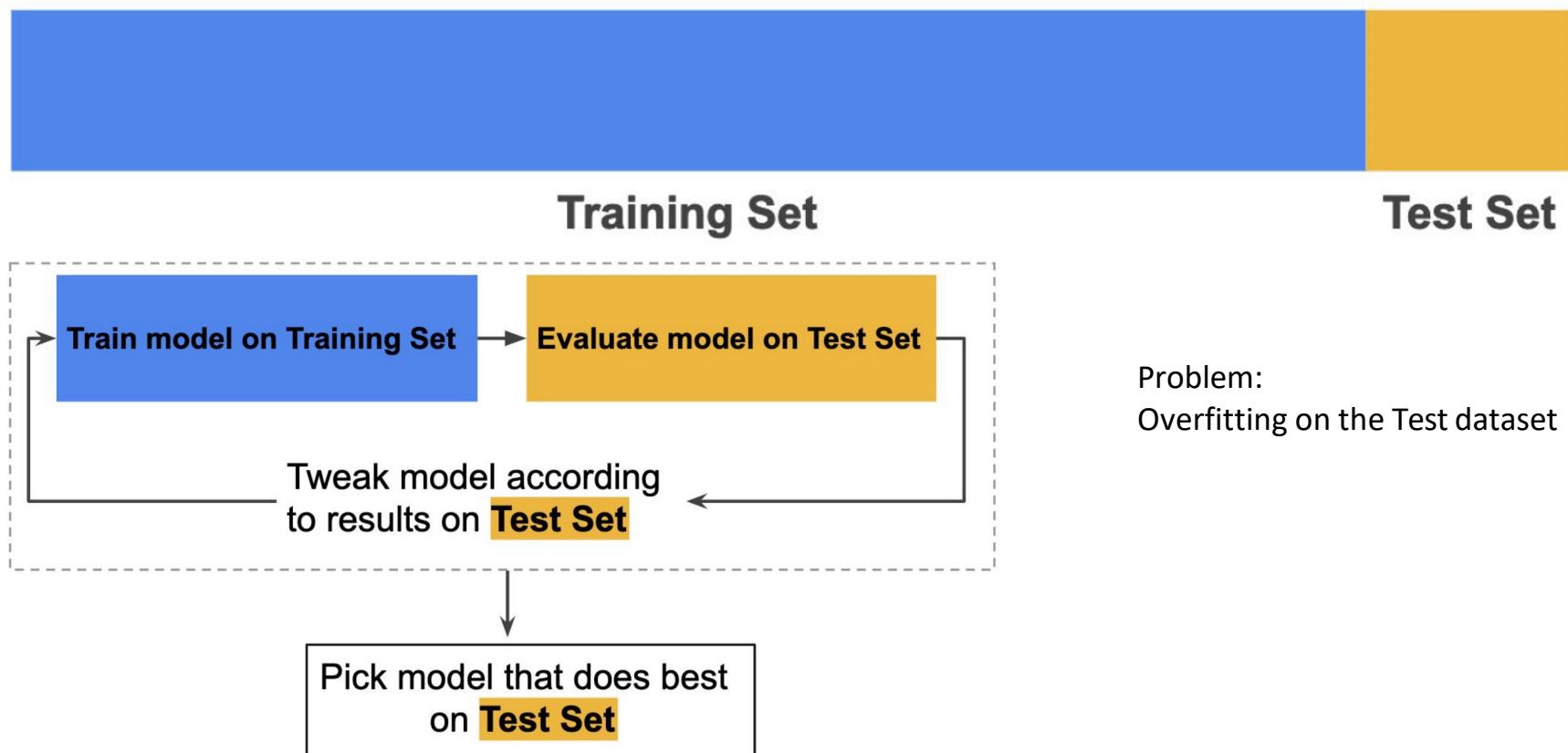
For example, say you are trying to predict a particular disease.

1. Assume you have features like age, gender and medications.
2. You have to predict whether this patient will get a disease or not.
3. Assume the disease occurs in 90% of the population.
4. How should you build your algorithm? What metric should you look at?
  
5. **Answer:** You should look at class-specific error (sensitivity/specificity).
  - a. This is unbalanced data.
  - b. Even an algorithm that randomly outputs 1 that is the person has disease, (regardless of the input features), would give 90% accuracy on the data.

# Partitioning the data



# Partitioning the data (WRONG)



# Decision Trees

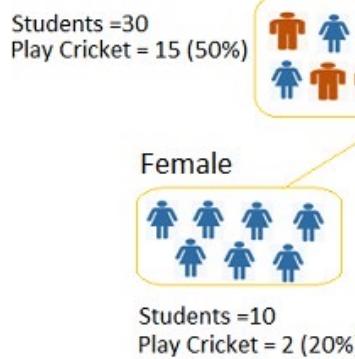
How to pack groceries?

- Tomatoes, soft vegetables
- Wine, fragile items
- Bread
- Toiletries, shampoo
- Meat, fish

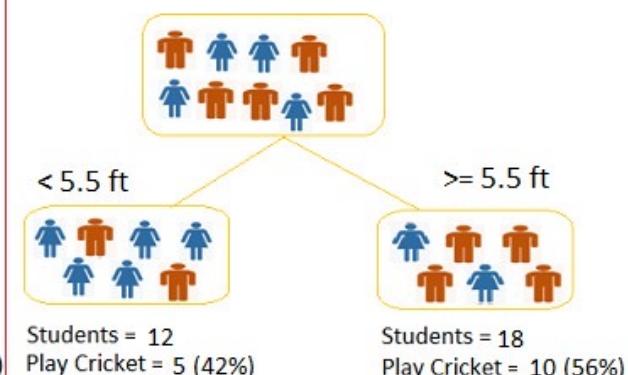


# Decision Trees

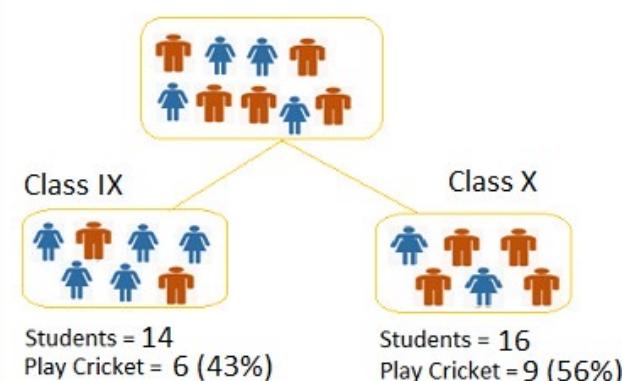
## Split on Gender



## Split on Height



## Split on Class



# Decision Trees. Splitting decision

## Information Gain

The more homogenous something is the less information is needed to describe it and hence it has gained information. Information theory has a measure to define this degree of disorganization in a system and it is known as Entropy. If a sample is completely homogeneous, then the entropy is zero and if it is equally divided (50% – 50%), it has entropy of one.

Entropy can be calculated using formula

$$\text{Entropy} = -p \log_2 p - q \log_2 q$$

Where p and q are probability of success and failure

## Reduction in Variance

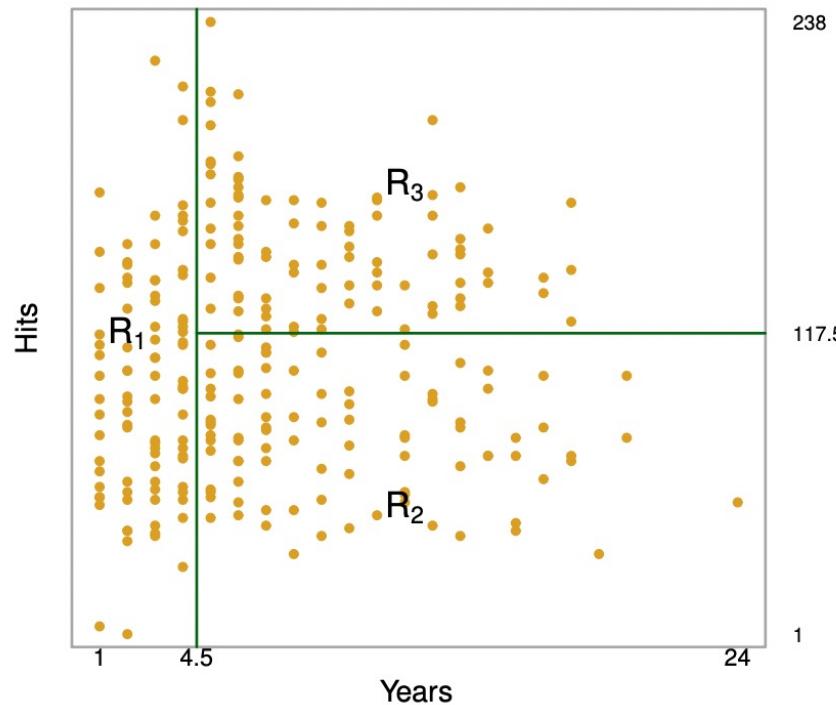
Reduction in variance is an algorithm used for continuous target variables (regression problems). This algorithm uses the standard formula of variance to choose the best split. The split with lower variance is selected as the criteria to split the population:

# Decision Trees



**FIGURE 8.1.** For the `Hitters` data, a regression tree for predicting the log salary of a baseball player, based on the number of years that he has played in the major leagues and the number of hits that he made in the previous year. At a given internal node, the label (of the form  $X_j < t_k$ ) indicates the left-hand branch emanating from that split, and the right-hand branch corresponds to  $X_j \geq t_k$ . For instance, the split at the top of the tree results in two large branches. The left-hand branch corresponds to `Years<4.5`, and the right-hand branch corresponds to `Years>=4.5`. The tree has two internal nodes and three terminal nodes, or leaves. The number in each leaf is the mean of the response for the observations that fall there.

# Decision Trees



**FIGURE 8.2.** The three-region partition for the `Hitters` data set from the regression tree illustrated in Figure 8.1.

# Decision Trees

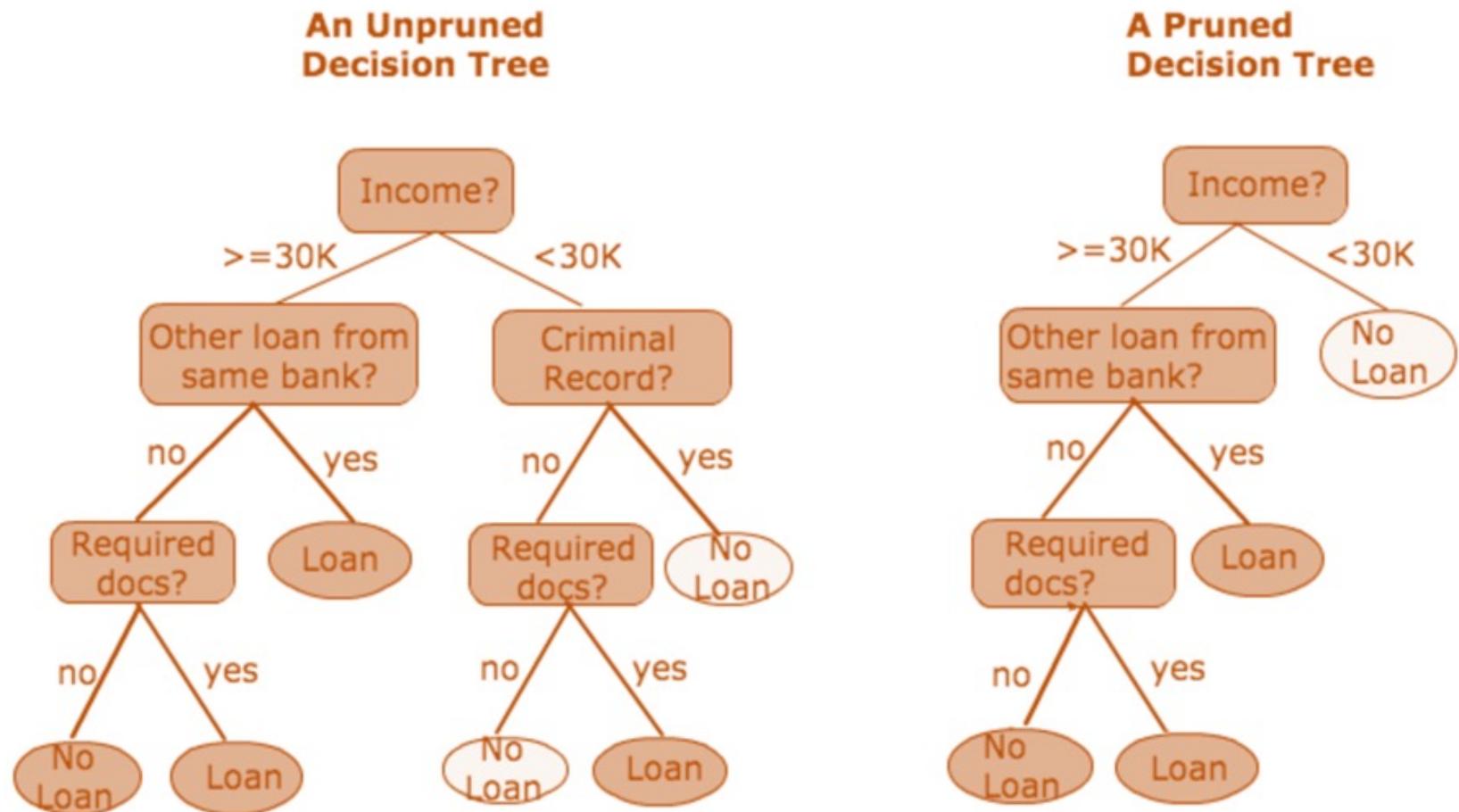
We now elaborate on Step 1 above. How do we construct the regions  $R_1, \dots, R_J$ ? In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model. The goal is to find boxes  $R_1, \dots, R_J$  that minimize the RSS, given by

$$\sum_{j=1}^J \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2, \quad (8.1)$$

where  $\hat{y}_{R_j}$  is the mean response for the training observations within the  $j$ th box. Unfortunately, it is computationally infeasible to consider every possible partition of the feature space into  $J$  boxes. For this reason, we take a *top-down, greedy* approach that is known as *recursive binary splitting*. The approach is *top-down* because it begins at the top of the tree (at which point all observations belong to a single region) and then successively splits the predictor space; each split is indicated via two new branches further down on the tree. It is *greedy* because at each step of the tree-building process, the *best* split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.

recursive  
binary  
splitting

# Decision Trees: how to get smaller trees



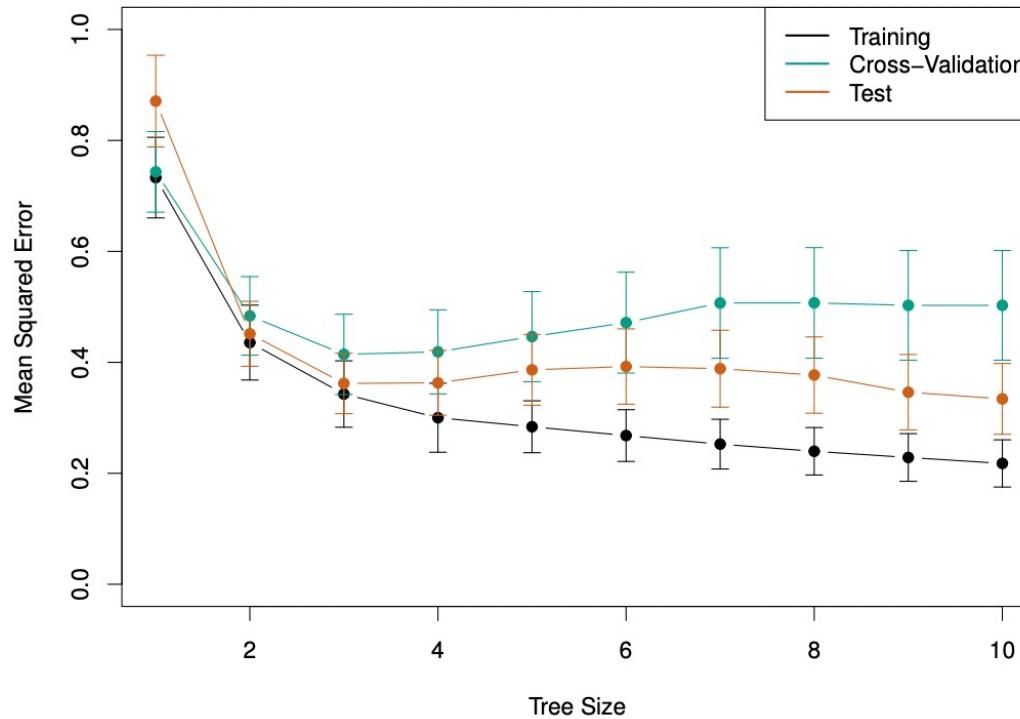
# Decision Trees: how to get smaller trees

*Cost complexity pruning*—also known as *weakest link pruning*—gives us a way to do just this. Rather than considering every possible subtree, we consider a sequence of trees indexed by a nonnegative tuning parameter  $\alpha$ . For each value of  $\alpha$  there corresponds a subtree  $T \subset T_0$  such that

cost  
complexity  
pruning  
weakest link  
pruning

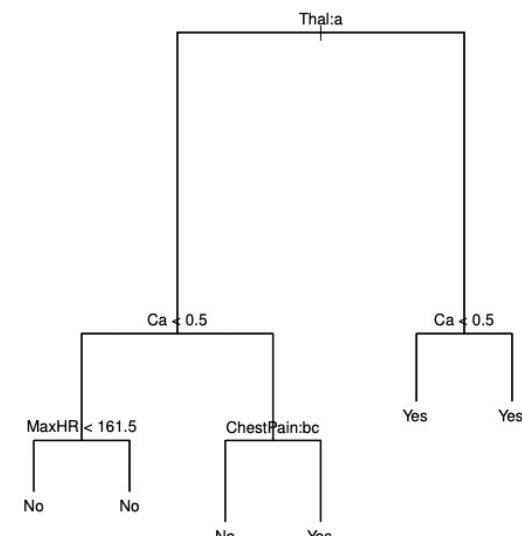
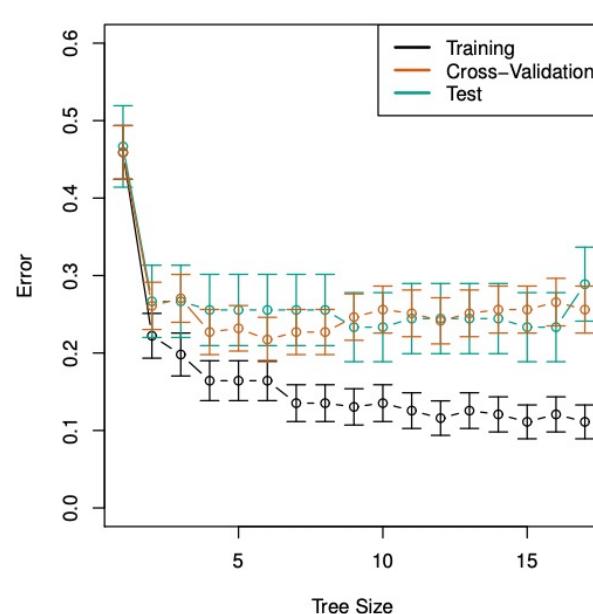
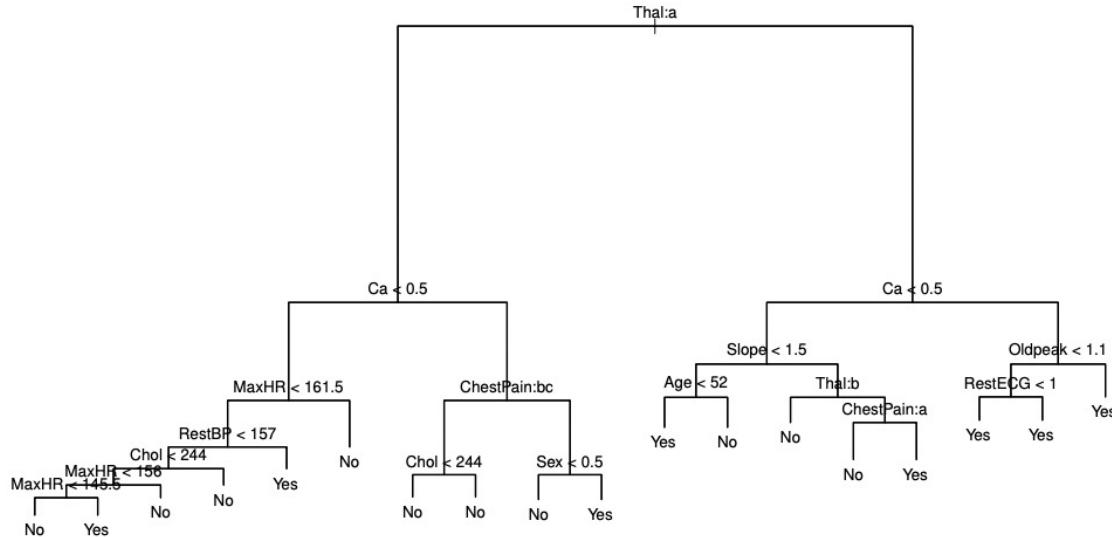
$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha|T| \quad (8.4)$$

# Decision Trees



**FIGURE 8.5.** Regression tree analysis for the `Hitters` data. The training, cross-validation, and test MSE are shown as a function of the number of terminal nodes in the pruned tree. Standard error bands are displayed. The minimum cross-validation error occurs at a tree size of three.

# Classification trees



**FIGURE 8.6.** Heart data. Top: The unpruned tree. Bottom Left: Cross-validation error, training, and test error, for different sizes of the pruned tree. Bottom Right: The pruned tree corresponding to the minimal cross-validation error.

# Pros and cons of trees

## *8.1.4 Advantages and Disadvantages of Trees*

Decision trees for regression and classification have a number of advantages over the more classical approaches seen in Chapters 3 and 4:

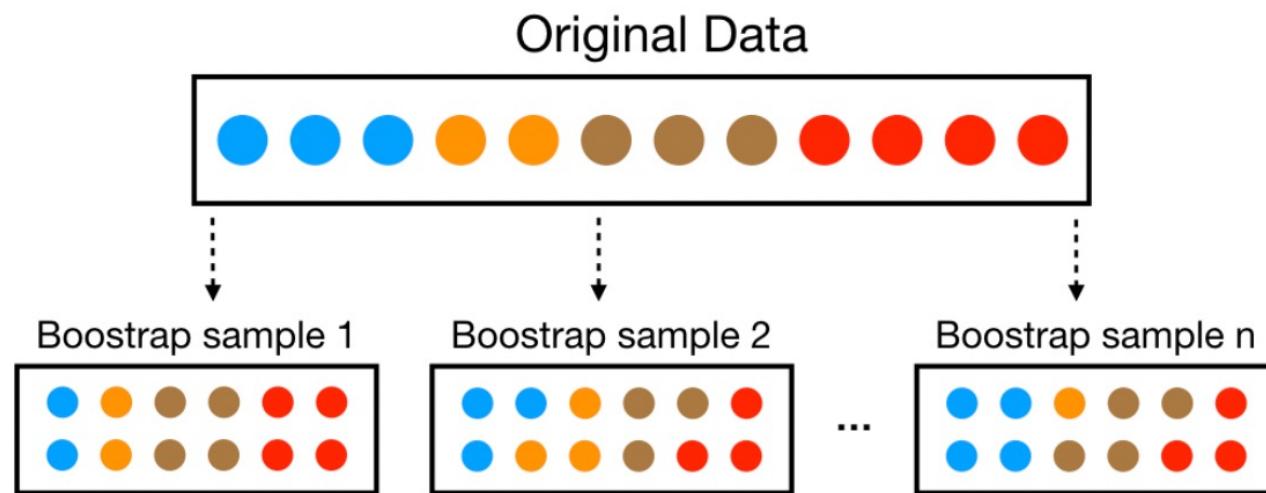
- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.

# Pros and cons of trees

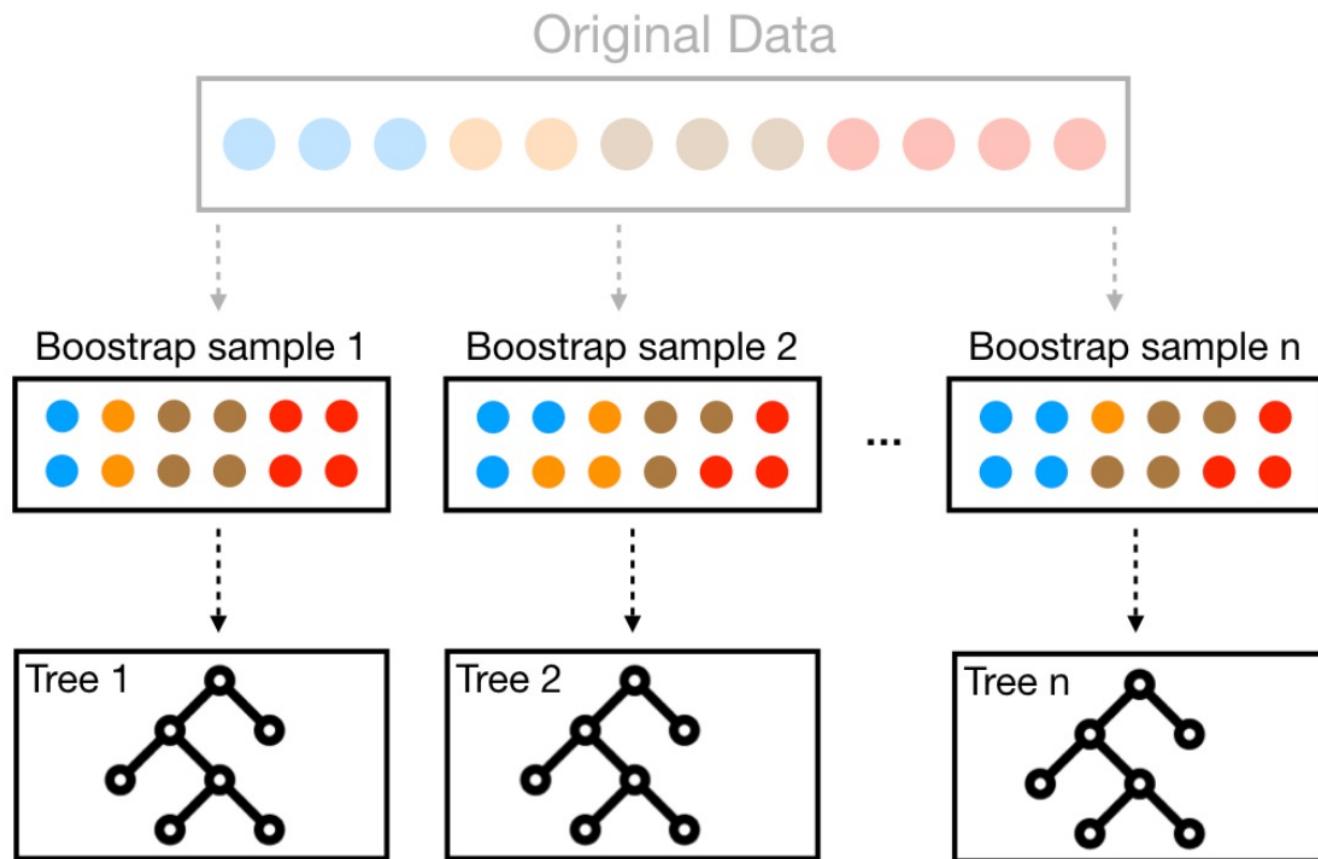
- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.
- ▼ Additionally, trees can be very non-robust. In other words, a small change in the data can cause a large change in the final estimated tree.

However, by aggregating many decision trees, using methods like *bagging*, *random forests*, and *boosting*, the predictive performance of trees can be substantially improved. We introduce these concepts in the next section.

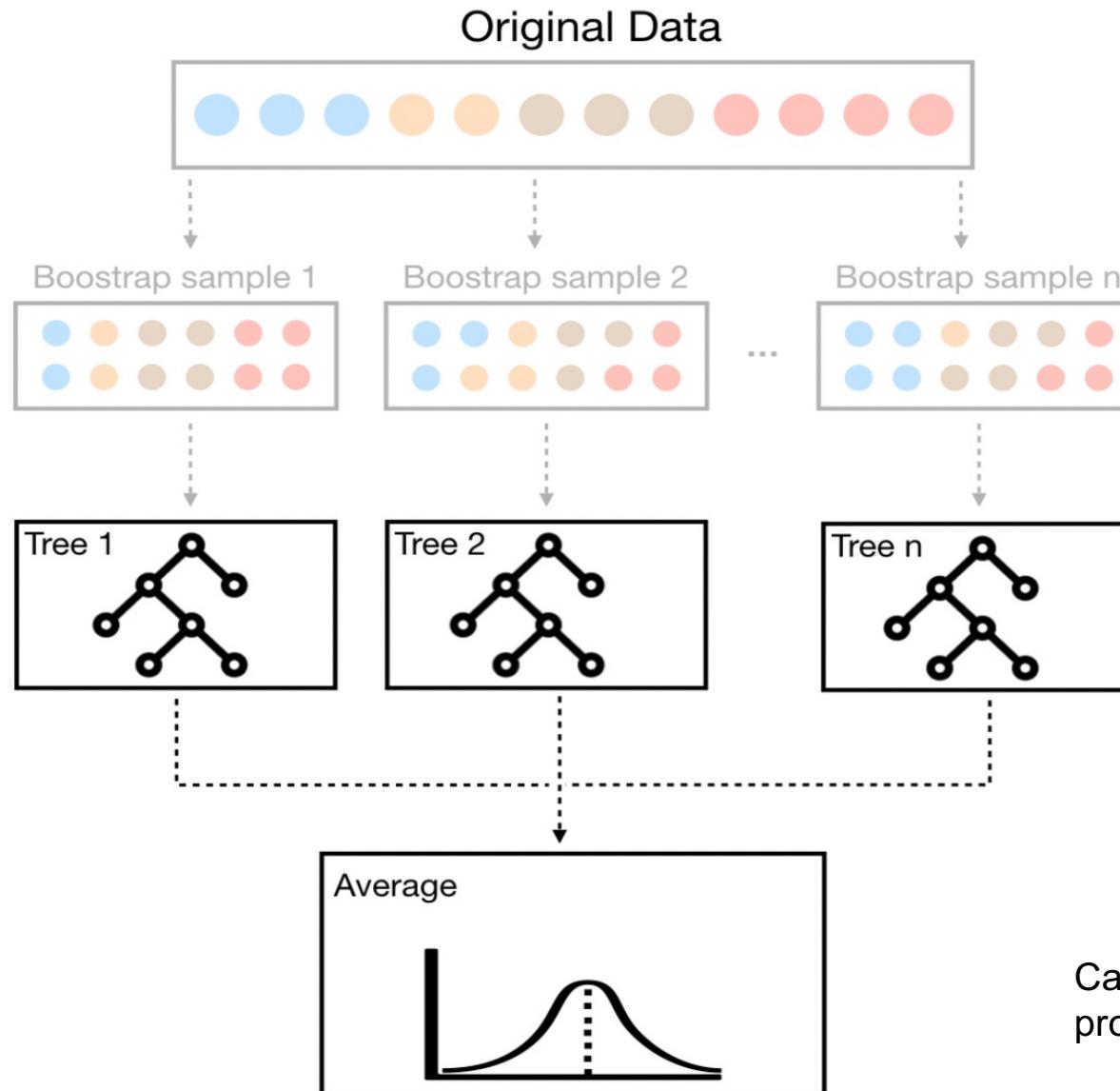
# Bagging



# Bagging

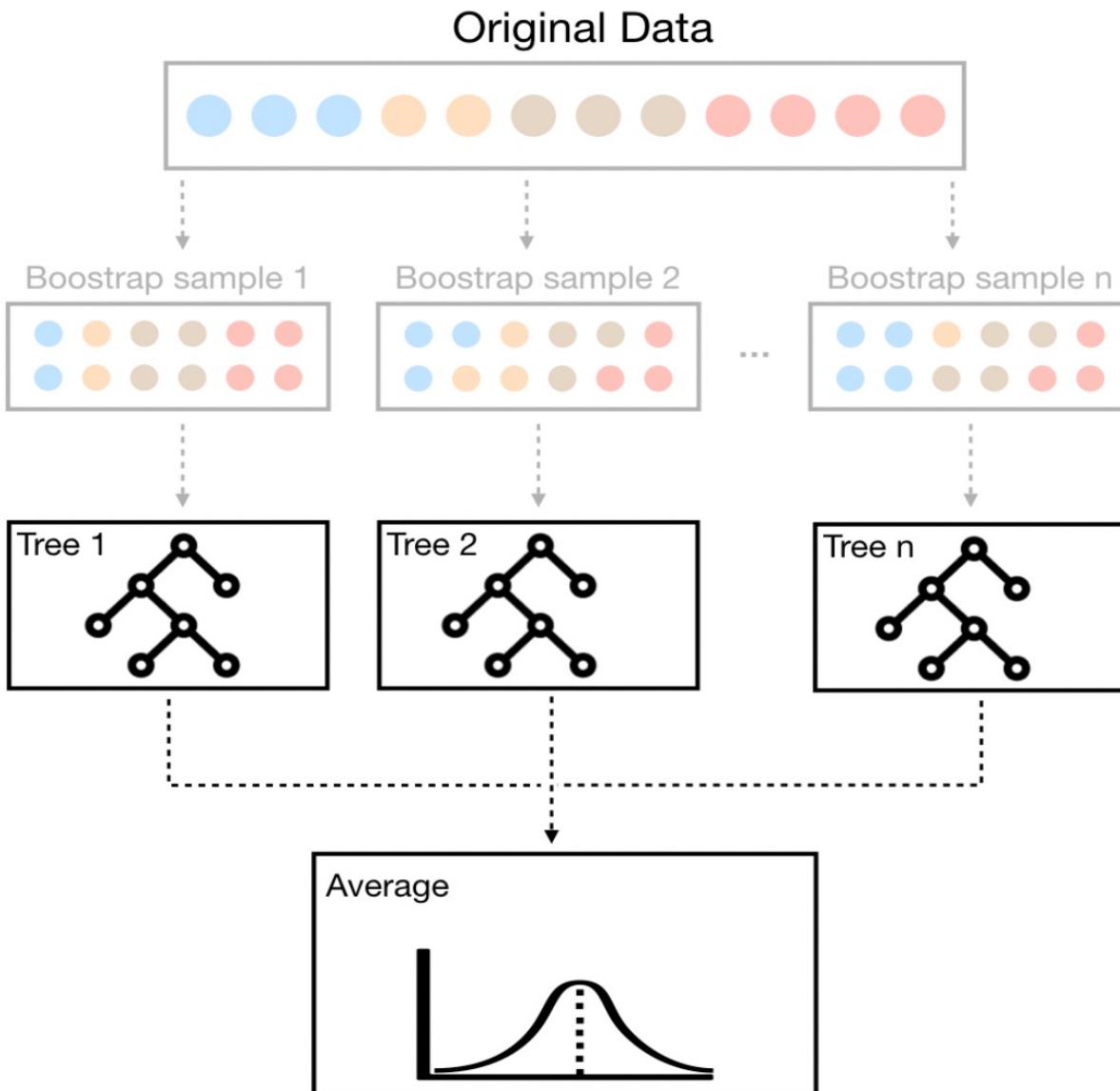


# Bagging



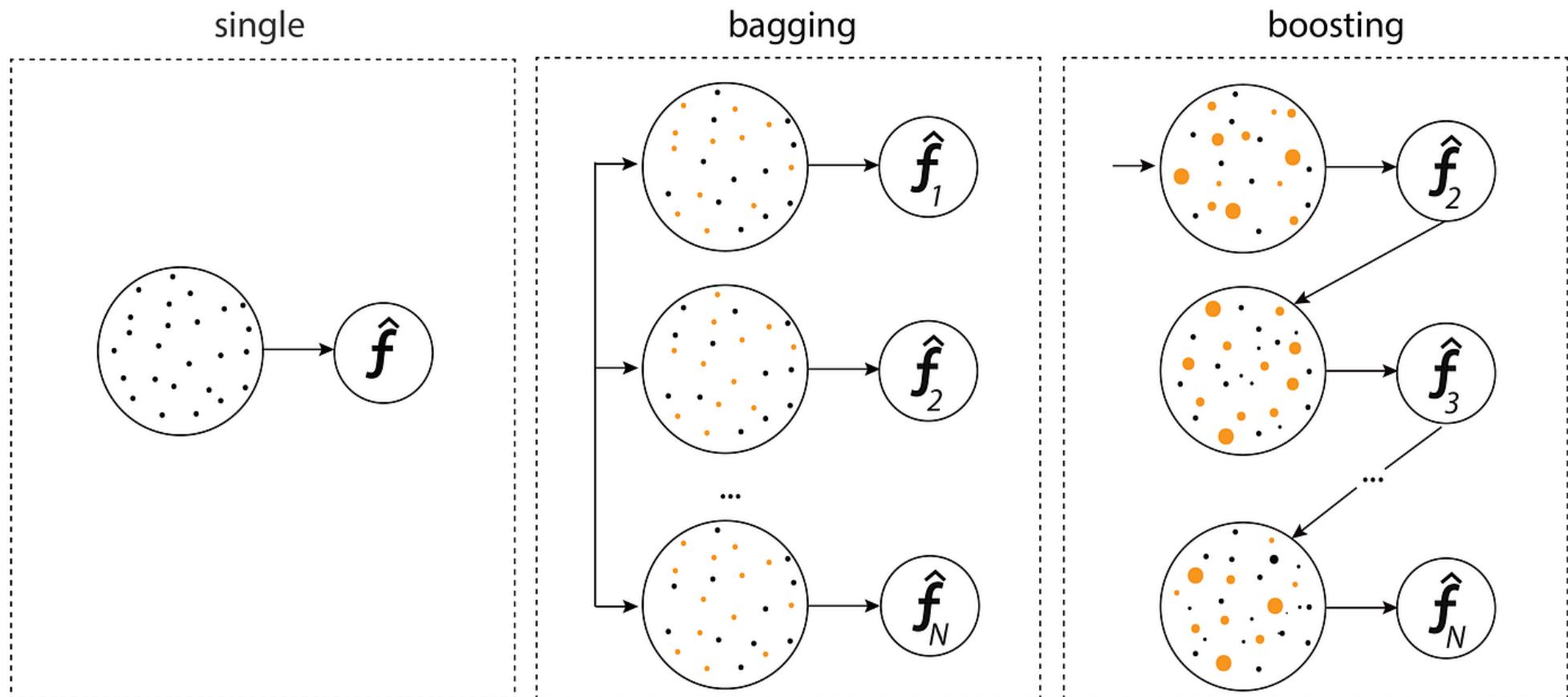
Can this lead to any problems?

# Random forests + Bagging



1. Decorrelate trees
2. When deciding on a split, a random sample of features is taken
3. Each tree is built on a different random sample of features!

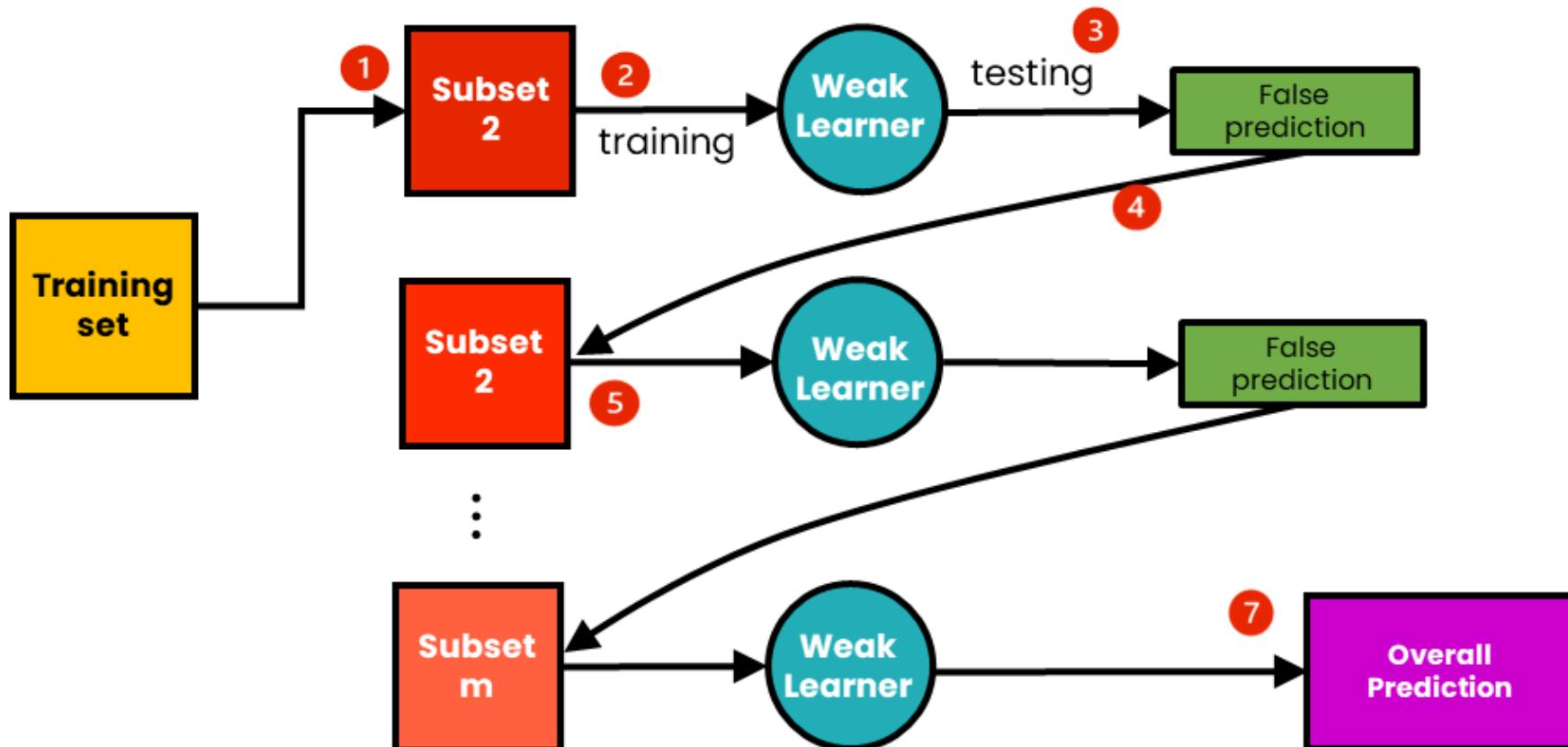
# Random Forests (Bagging vs. Boosting)



<https://medium.com/@senozanAleyna/ensemble-boosting-bagging-and-stacking-machine-learning-6a09c31df778>

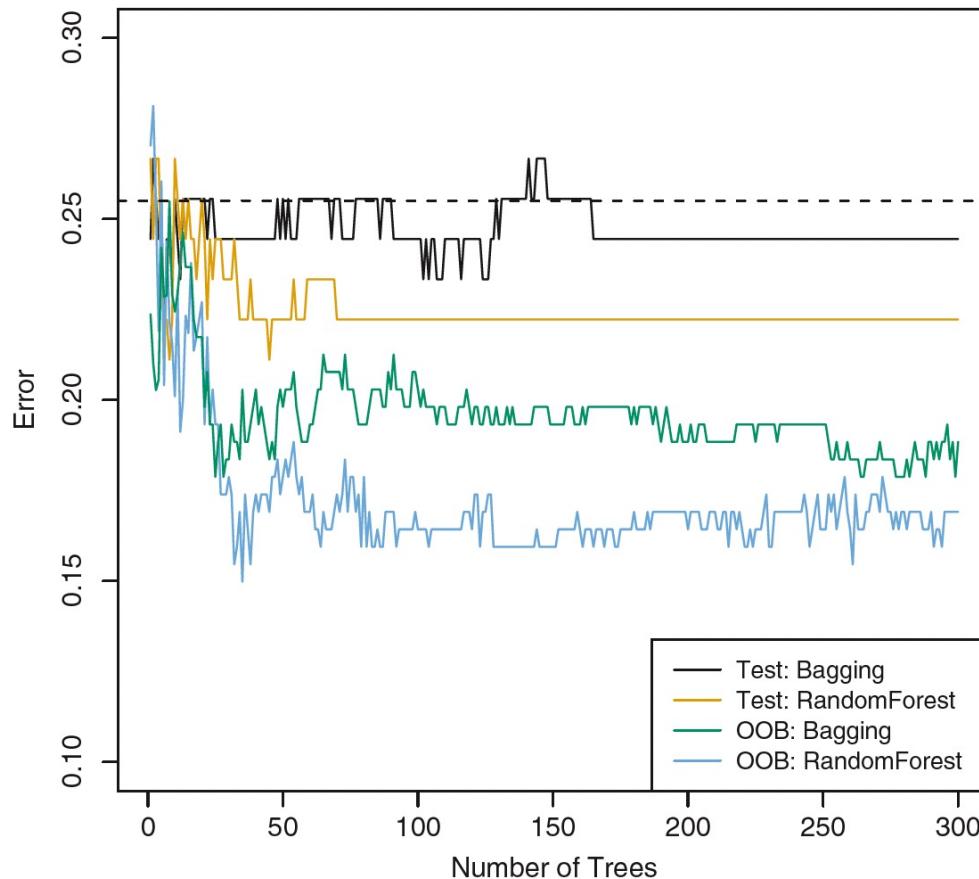
# Random Forests (Boosting)

## The Process of Boosting



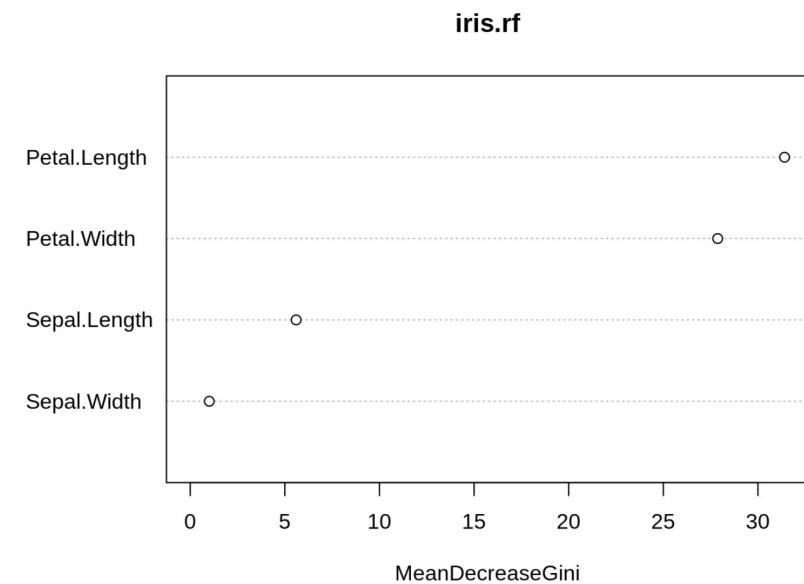
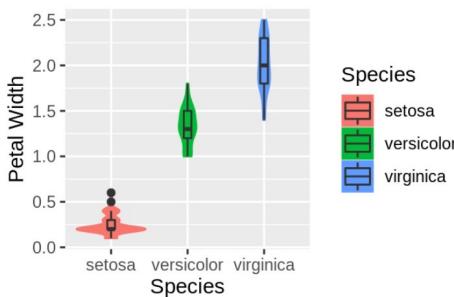
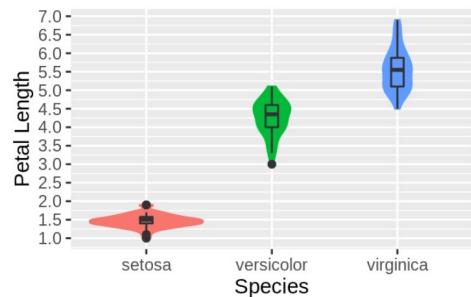
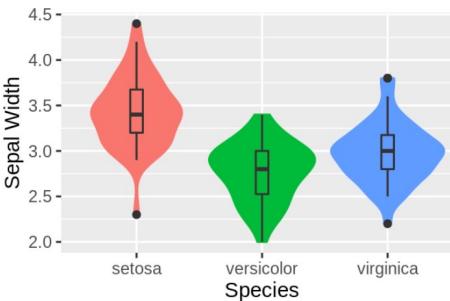
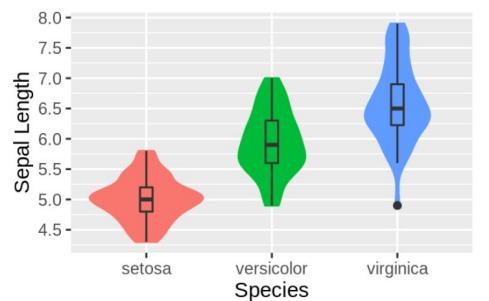
[https://medium.com/@brijesh\\_soni/understanding-boosting-in-machine-learning-a-comprehensive-guide-bdeaa1167a6](https://medium.com/@brijesh_soni/understanding-boosting-in-machine-learning-a-comprehensive-guide-bdeaa1167a6)

# Random Forests (PRACTICAL)



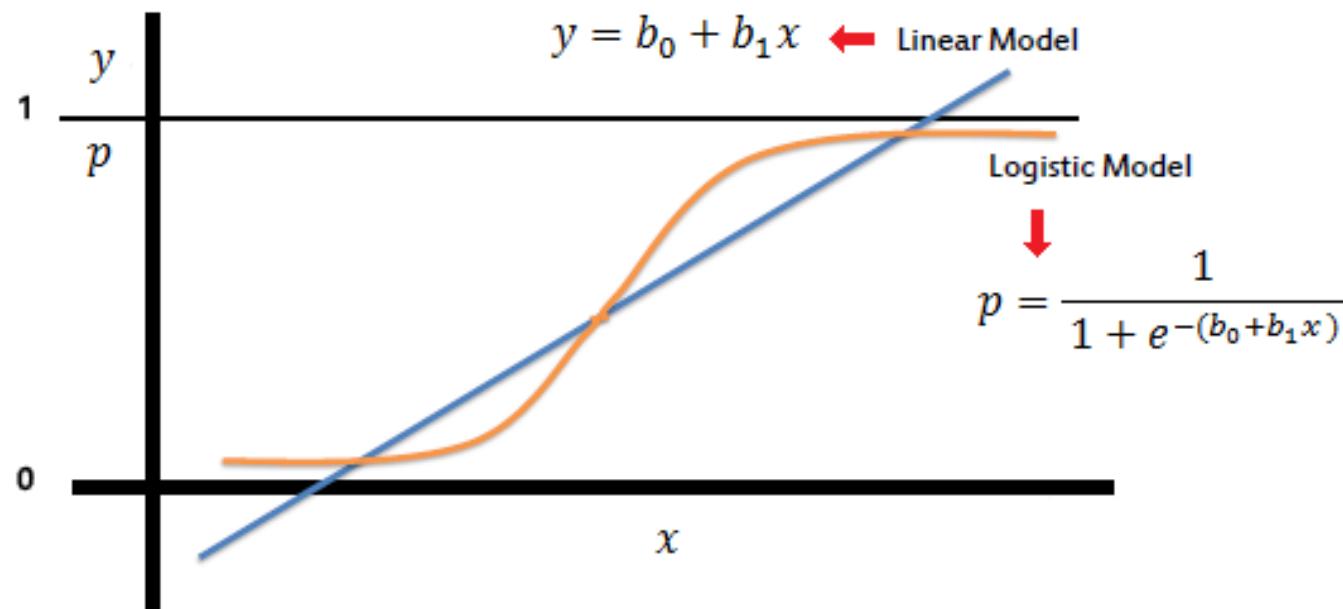
**FIGURE 8.8.** Bagging and random forest results for the Heart data. The test error (black and orange) is shown as a function of  $B$ , the number of bootstrapped training sets used. Random forests were applied with  $m = \sqrt{p}$ . The dashed line indicates the test error resulting from a single classification tree. The green and blue traces show the OOB error, which in this case is considerably lower.

# Random Forests: Variable Importance (PRACTICAL)



# Artificial Neural Networks

Iterated logistic regression

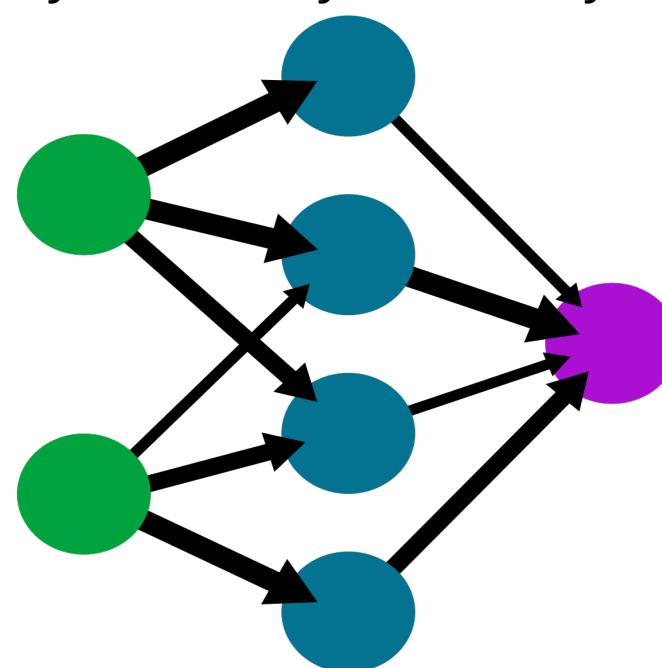


# Artificial Neural Networks

Iterated logistic regression

A simple neural network

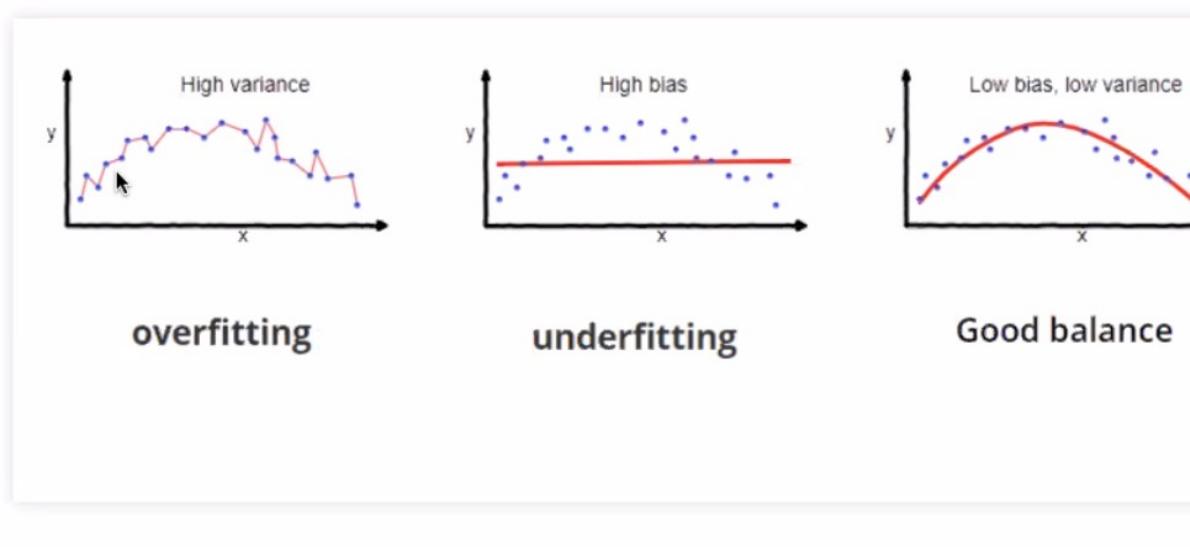
input layer      hidden layer      output layer



[https://en.wikipedia.org/wiki/Neural\\_network#/media/File:Neural\\_network\\_example.svg](https://en.wikipedia.org/wiki/Neural_network#/media/File:Neural_network_example.svg)

# A fundamental concept in machine learning

## Bias-variance tradeoff

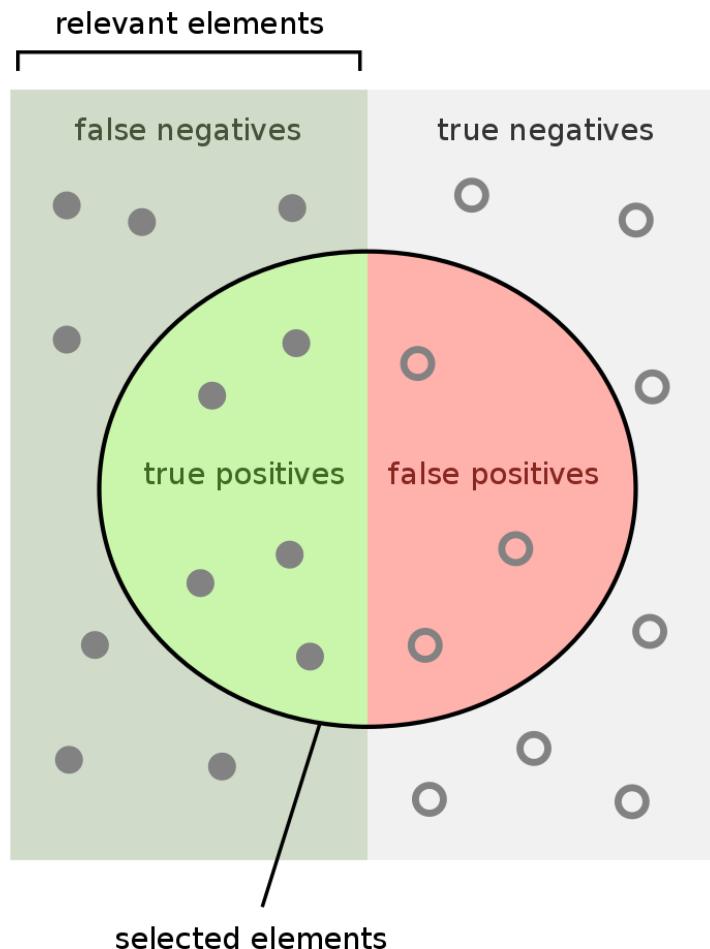


- Bias is residual error from fitting the Training data
- Variance is generalization error when applying the model fit to

# A fundamental concept in machine learning



**FIGURE 5.1.** A schematic display of the validation set approach. A set of  $n$  observations are randomly split into a training set (shown in blue, containing observations 7, 22, and 13, among others) and a validation set (shown in beige, and containing observation 91, among others). The statistical learning method is fit on the training set, and its performance is evaluated on the validation set.



How many relevant items are selected?  
e.g. How many sick people are correctly identified as having the condition.

$$\text{Sensitivity} = \frac{\text{true positives}}{\text{relevant elements}}$$

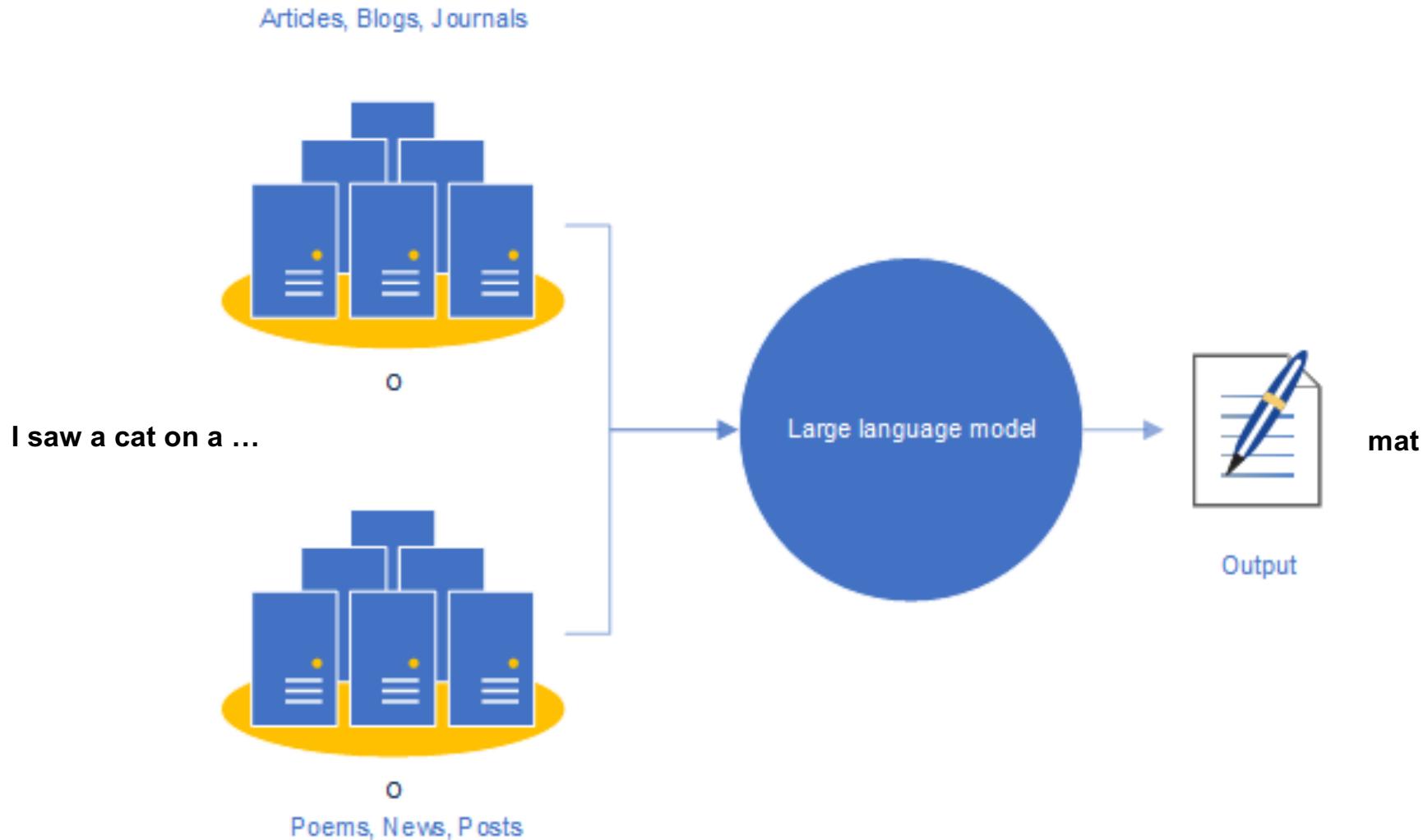


How many negative selected elements are truly negative?  
e.g. How many healthy people are identified as not having the condition.

$$\text{Specificity} = \frac{\text{true negatives}}{\text{relevant elements}}$$

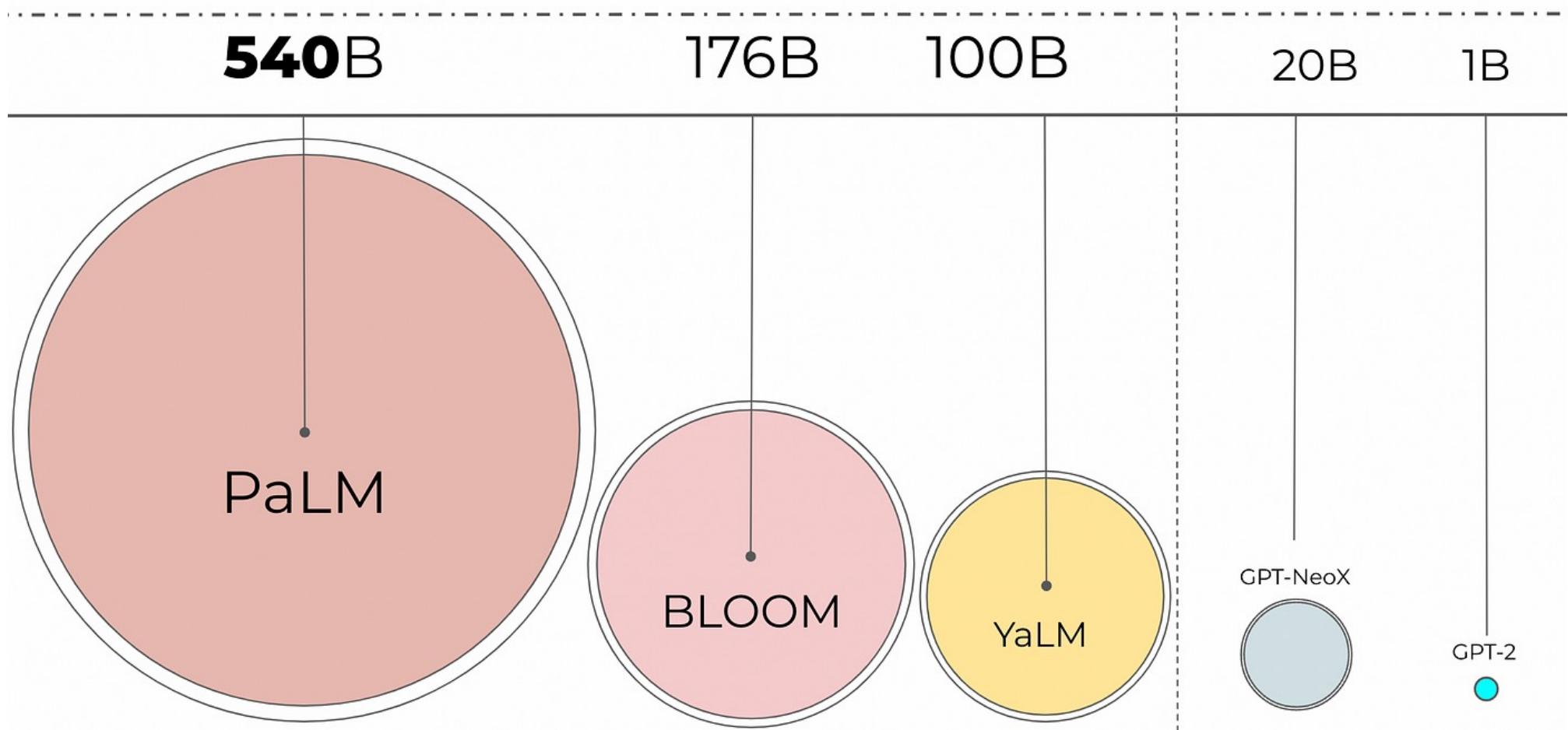


# Applications of this concept



# Applications of this concept

## Large Language Models - sorted by billion parameters



# Resources

- <https://www.statlearning.com/>
- Free PDF of book and R code
- <https://www.statlearning.com/resources-second-edition>
- More practical tutorials and R code
- <https://cambiotraining.github.io/intro-machine-learning/>

# Questions