Introduction to Machine Learning

Ryan Miller

Grinnell College
Statistics

What is Machine Learning?

- What comes to mind when you hear "Machine Learning"?
- How does machine learning relate to other areas such as AI, traditional statistical modeling, and traditional computer science?

Machine Learning vs. Computer Science

Our definition:

> *Machine learning is the study of algorithms that learn patterns from data to make accurate predictions on new, unseen examples*

- Traditional computer science: *Rules + Data -> Answers*
  - Examples: tax software, retirement planning software, etc.
- Machine learning: *Data + Answers -> Learned Rules -> Predictions*
  - Examples: spam detection, home price prediction, Netflix recommendations

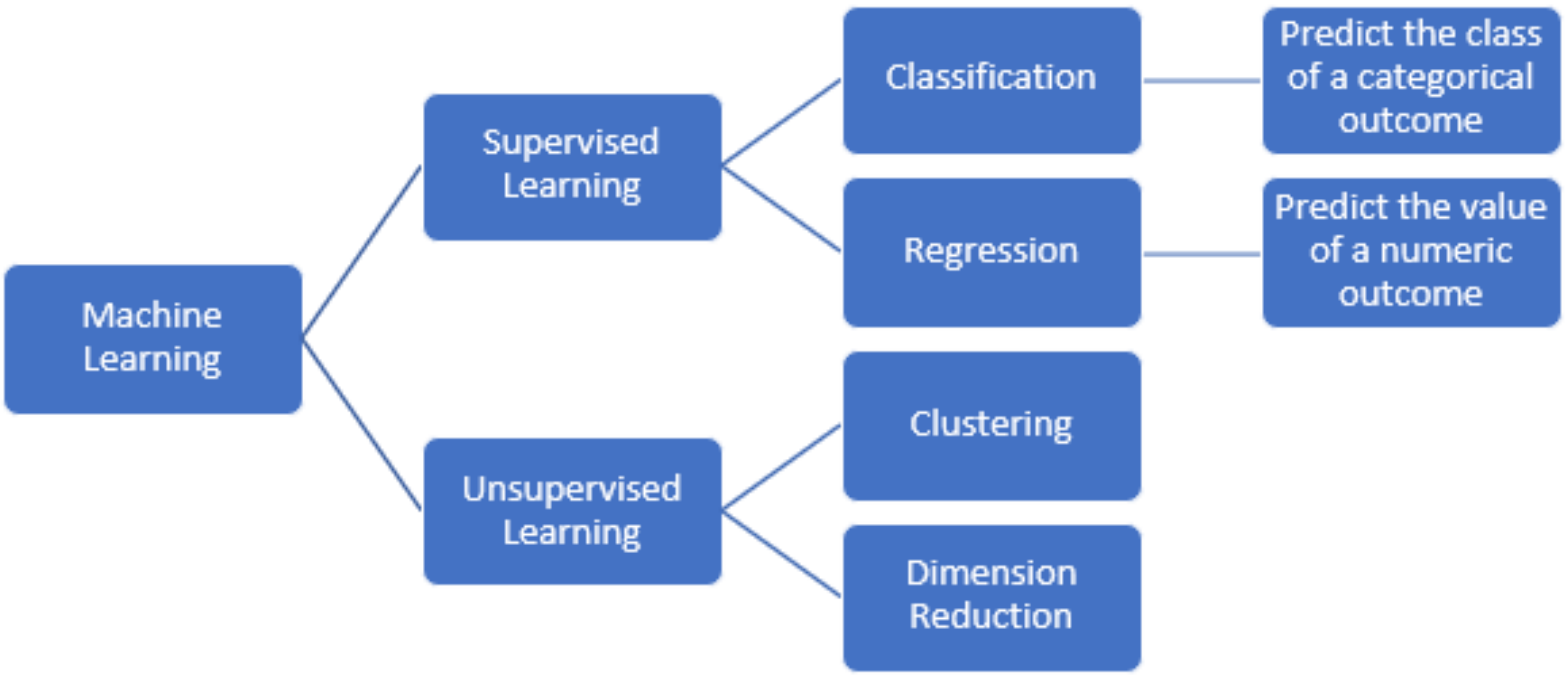
Machine Learning vs. Statistics

- Traditional statistics: understand relationships and draw scientific conclusions
  - Focused on inference, accounting for uncertainty, simplified explanations
  - Example: how do lifestyle factors affect the risk of diabetes?
- Machine learning: learn patterns from training data to make accurate predictions on new data
  - Focused on predictive accuracy, generalization, computational constraints
  - Example: can we predict who will develop diabetes?

Machine Learning vs. AI

- Machine learning provides the foundation for modern AI systems (ChatGPT, Claude, etc.)
  - At a high level, these models are trained to predict the next piece of text/content when given input text/content
  - For example, when given "The capital of Iowa is ____" the model might predict "Des Moines" given the patterns it learned from its training data (Wikipedia, news articles, etc.)
- Our course is an introduction to machine learning - namely the concepts of learning from data, generalization, and model evaluation that will underlie AI systems regardless of the specific models being used

An Overview of Machine Learning



Getting Started

We'll begin by working with **tabular data** that is organized in a matrix, $\mathbf{X}$, consisting of n samples (rows) with p features (columns)

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,p} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,p} \end{pmatrix}$$

- Bold capital letters denote a matrix, bold lower-case letters denote a vector
- Samples are indexed by i, such that $\mathbf{x}_i$ denotes the feature vector for the i^{th} sample, and features are indexed by j

Getting Started (cont.)

In supervised learning we focus on a target variable, Y. We'll adopt the general framework for regression tasks:

$$y_i = f(\mathbf{x}_i) + \text{Noise}$$

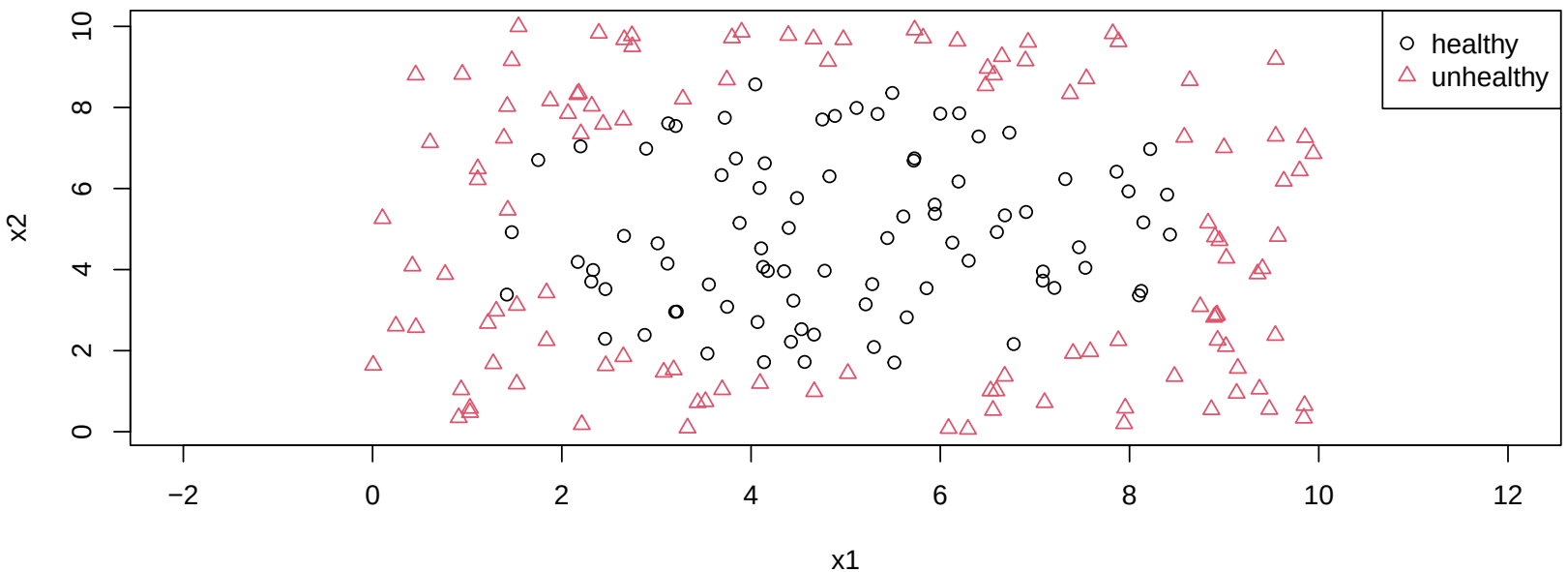
For classification tasks, we instead model class probabilities:

$$Pr(Y = 1 \mid \mathbf{x}_i) = f(\mathbf{x}_i)$$

In both scenarios, $f()$ is an unknown function of the data, and we seek to find an algorithm that approximates the behavior of $f()$

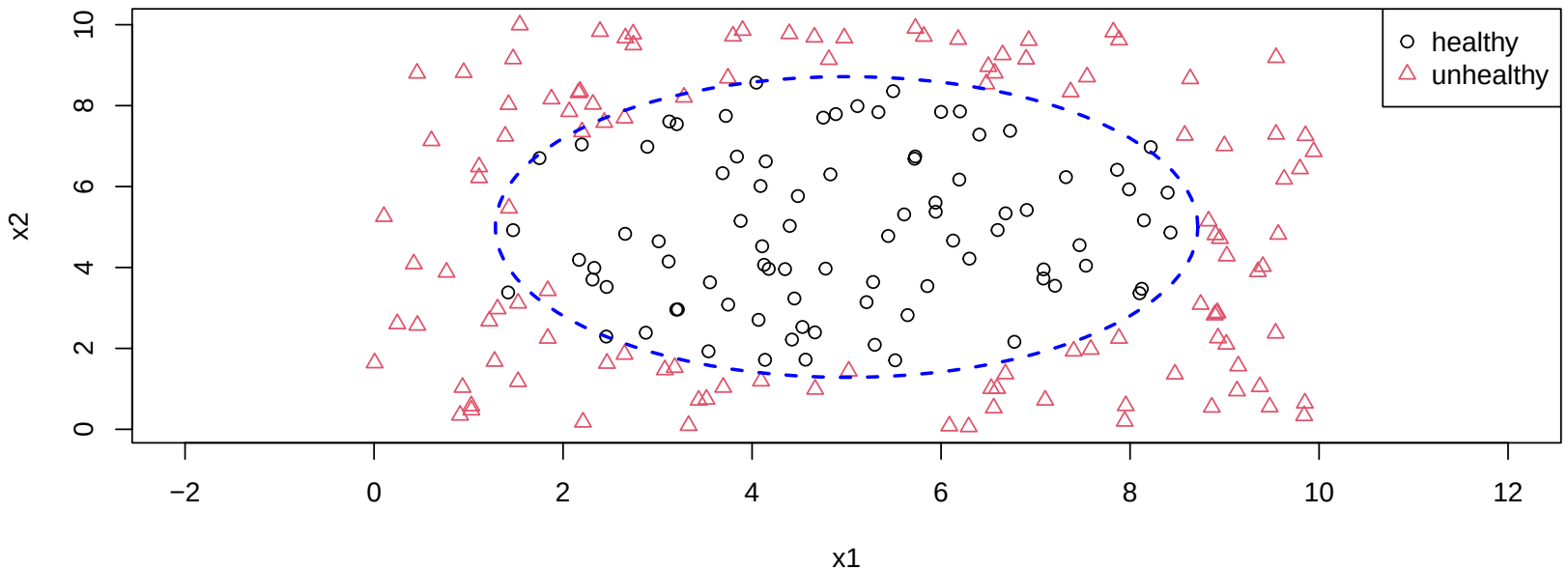
A Simple Example

Consider two predictors, X_1 and X_2, and an outcome y of "healthy" or "unhealthy". Can these predictors be used to accurately *classify* an observation? What might be a good estimate of $f()$?



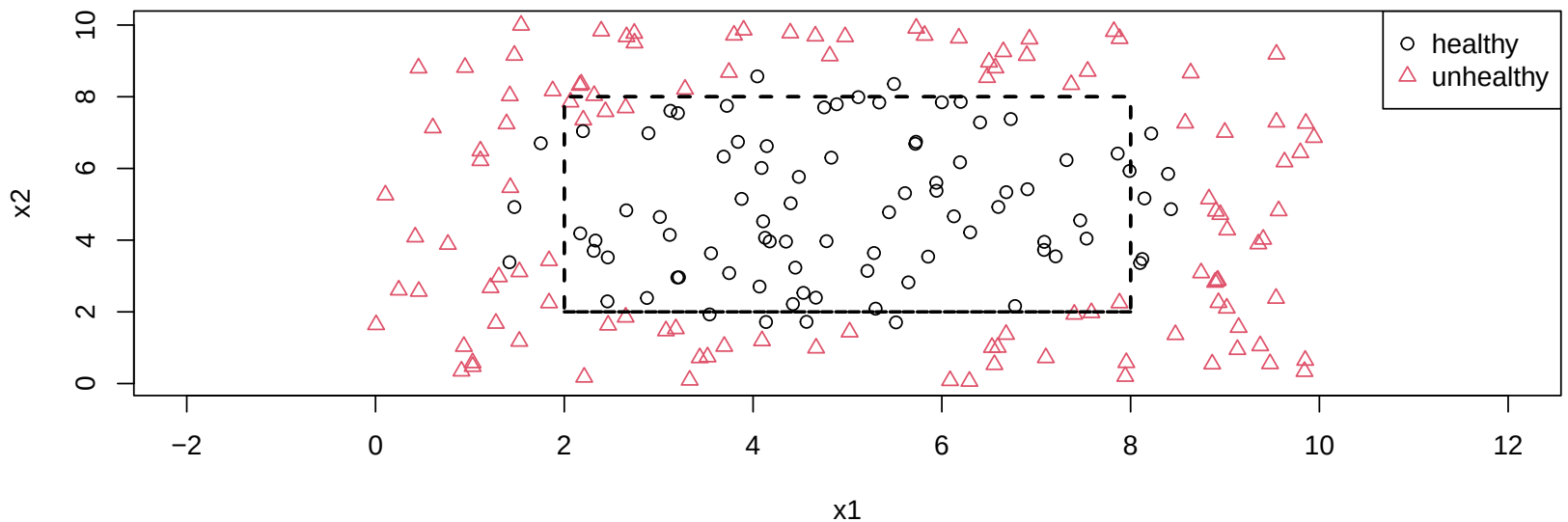
Example (cont.)

Yes! In this example, the true $f()$ is given by the blue ellipse:



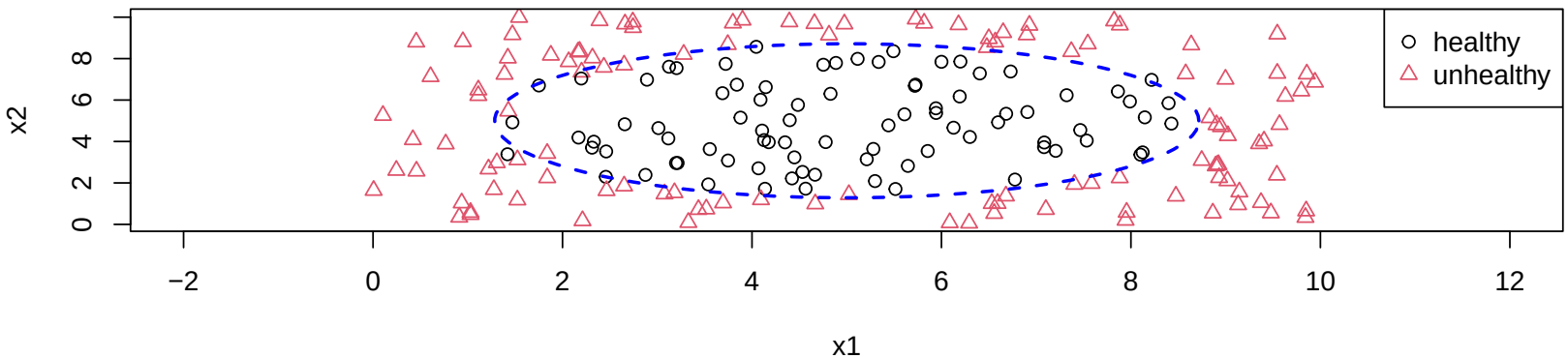
Example (cont.)

As a human, you might observe that the healthy data-points tend to fall between 2 and 8 in x_1 and x_2, so you might propose the *classification model* shown below. This model correctly classifies 178 of 200 data-points (89% accuracy).



Reducible vs. Irreducible Error

Now let's revisit the true relationship between x_1, x_2, and y. Notice that some "healthy" data-points are outside the ellipse, and some "unhealthy" ones are inside it:



These misclassifications contribute to **irreducible error**. Even if we perfectly recover $f()$ we will not achieve 100% accuracy.

Irreducible Error in Real Life

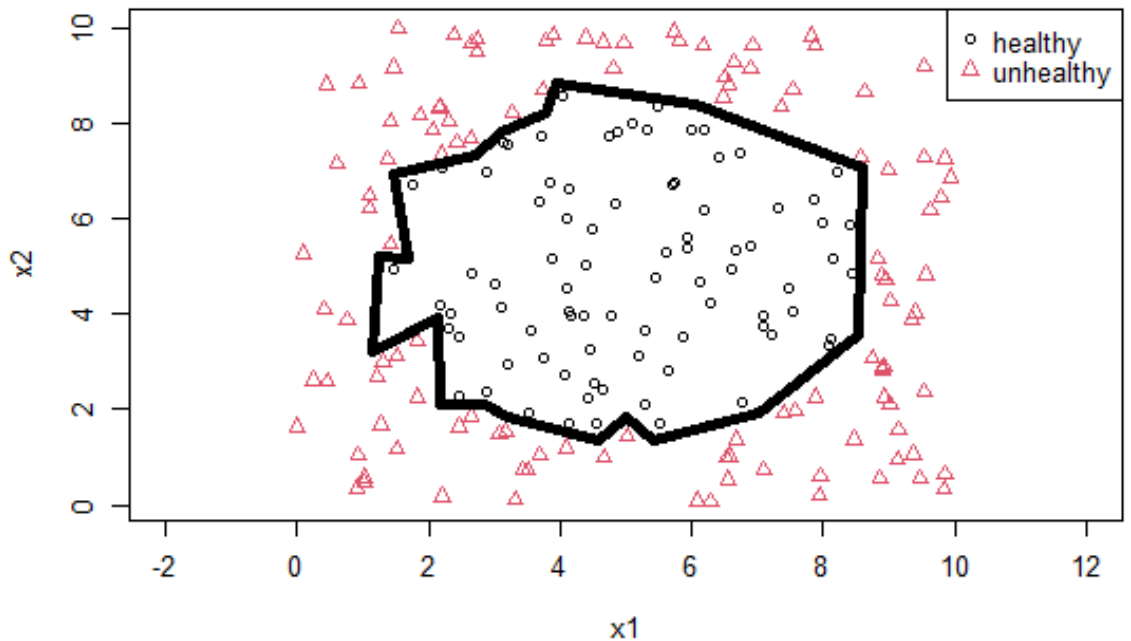
- A group of Grinnell students and I developed ML models to predict the destination of incoming ITS help tickets
  - Historically, an ITS employee routed tickets to the appropriate person/department
  - We trained our models using previous tickets and where they were sent
- Why might we not be able to predict destinations with perfect accuracy?

Irreducible Error in Real Life

- The human employees who routed the tickets might not have made perfect decisions
  - That is, they might rely upon some rules, but they likely also introduce noise into their decision making (subjective feelings, etc.)
- No matter how good our models were, they are limited by what was present in the training data

Reducible Error

Considering the presence of irreducible error, we can frame the goal of machine learning as minimizing *reducible error*.



Has this classifier (the black line) reduced the error rate to zero?

Training vs. Testing Splits

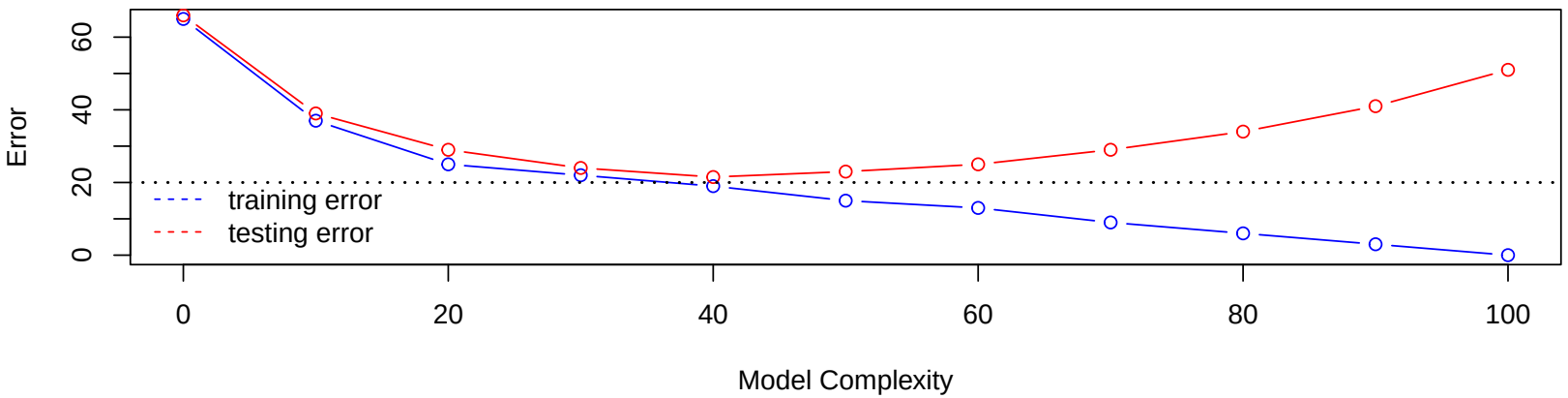
- We generally aren't interested in the error rate for the *observed data*
  - Instead, we'd like to minimize reducible error on *new data* that our model *hasn't yet seen*

Training vs. Testing Splits

- We generally aren't interested in the error rate for the *observed data*
  - Instead, we'd like to minimize reducible error on *new data* that our model *hasn't yet seen*
- Standard procedure is to divide the available data into **training** and **testing** sets
  - The training set is used to learn how to make predictions (ie: used to estimate $f()$)
  - The testing set is used to evaluate how well the learned patterns will generalize to unseen data

Training, Testing, and Error

Consider an example with an irreducible error of "20 units":



- Training error generally decreases with increasing the model complexity (ie: learning more rules)
- Testing error will never drop below the irreducible error rate (probabilistically speaking, it might for a specific test set)

Data Leakage

- In order to truly evaluate a model's effectiveness on unseen data it is essential that no information from the test set was used in any stage of model development
- Common examples of "data leakage" include:
  - Performing variable selection or exploratory data analysis before splitting off the test set
  - Applying scaling, standardization, or transformations before splitting
  - Checking performance on the test set more than once

Bias vs. Variance

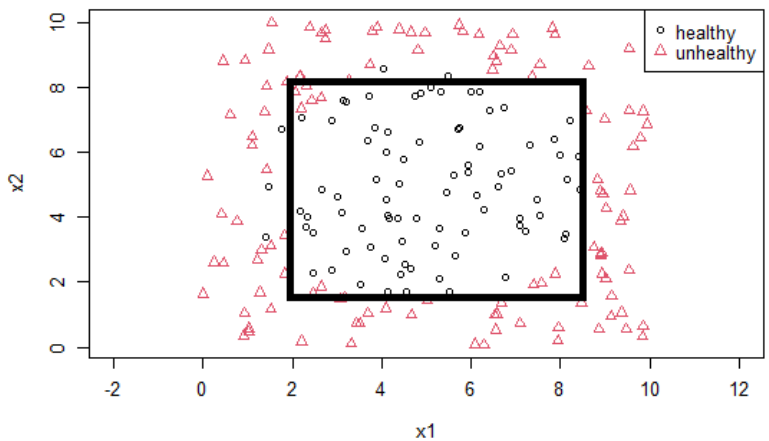
Reducible error can be attributed to one of two sources: **bias** or **variance**

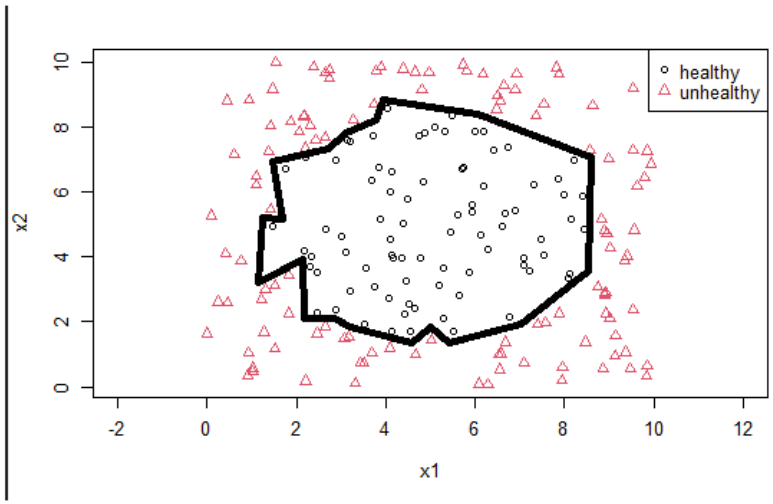
- **Bias** is when a learner lacks the structural flexibility to detect aspects of the true relationship between the predictors and the outcome
- **Variance** is when a learner is overly sensitive to chance artifacts present in the data (ie: the manifestations of irreducible error)

Poor performance due to high bias is called *underfitting*, while poor performance due to high variance is called *overfitting*

Bias vs. Variance

How would you compare the bias and variance of the following learners (a rectangle vs. an n-dimensional polygon)?





Defining Error

- So far we've focused on classifying a binary categorical outcome, a scenario where *classification accuracy* provides a natural framework for understanding a method's error
  - We'll talk about more sophisticated ways to evaluate error for classification tasks later on
- What if our goal is to predict a numeric outcome?

Defining Error

For a numeric outcome, it's most natural to measure error by summarizing the distances between predicted and observed outcomes:

- **Root Mean Squared Error**: $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$
- **Mean Absolute Error**: $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

In each definition, y_i is the observed outcome for the i^{th} example (data-point) and $\hat{y}_i$ is the predicted outcome for that example.

Preparing for Quiz #1

The following topics from these slides might appear on our first quiz:

1. Training vs. testing and data leakage
2. Reducible vs. irreducible error
3. The bias-variance trade off