Introduction to Machine Learning

Eric Ewing

Classical AI

Search and Planning

Optimization

Knowledge Representation and Reasoning

All rely on how the "world" is represented

What if we don't know all the rules of our world?

- Bayes Nets rely on the structure of the network…
- We need to know what the constraints are to run constrained optimization…
- If we don't know the rules of a game, we can't write down the PDDL for it…

Learning

What does it mean to *learn*?

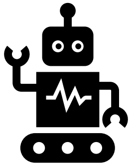
The more information you consume,
the more you know about a topic

The more you practice something, the
better you'll do

A teacher/coach teaches you a specific
skill by providing examples

Learning

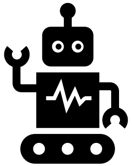
What does it mean to *learn*?



The more data we have, the better our algorithms should perform

The more experience we have, the better our algorithms should perform.

Types of Learning



Supervised Learning

Our dataset consists of a large number of examples of a task **and** the correct answers

Unsupervised Learning

Our dataset consists of a large number of examples, our algorithm tries to find patterns in this data

Reinforcement Learning

Our agent takes actions in an environment and is rewarded for taking “good” actions

Function Approximation

Supervised Learning

Our dataset consists of a large number of examples of a task **and** the correct answers

Inputs to our algorithm:

x: features of an example
y: Label

Our goal: Learn a **function approximation** that maps from features to labels

f() = Cat

Example image

Supervised Learning

Inputs:

Data: $X \in \mathbb{R}^{n \times d}$

n examples, each with *d* features

Labels: $y \in \mathbb{R}^{n}$ (for regression) or $y \in \{0, 1, \dots, \mathrm{num\_classes}\}$ (for classification)

n ground-truth labels

Outputs:

A Function approximation: $\tilde{f}\colon \mathbb{R}^{d} \rightarrow \mathbb{R}$ (for regression)

Goal: Approximate the true function f(x)=y as closely as possible

Supervised Learning

Step 1: How should we model f ?

Linear Regression: Perhaps there's a (close-to) linear trend between X and y… model $\tilde{f}$ as a line

Neural Network: can model even more complex relationships between X and y

Polynomial Regression: Instead of a linear relationship, model $\tilde{f}$ as a polynomial of some degree (e.g., 2 is a quadratic, 3 a cubic, and so on)

The space of all different types of functions we might choose from is quite large…

K Nearest Neighbors

Key Intuition: Examples that are “close” to each other should have similar labels to each other

Given dataset (X, y)

To predict a label for a new example input *x*:

Find the closest *k* examples in X

use the labels of those *k* examples to label *x*

Two possible classes, red triangles and blue squares

How should we label the green circle?

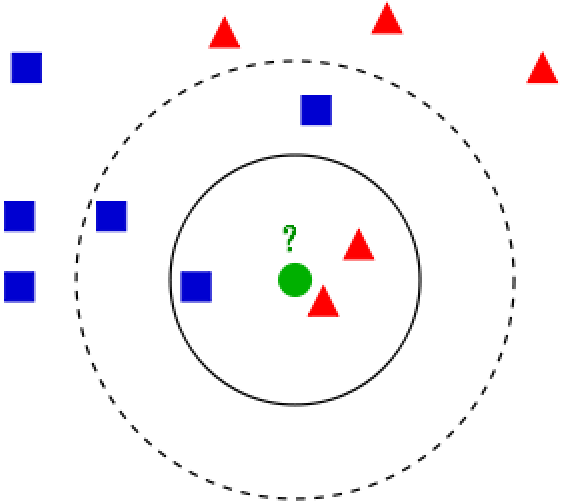


Image Source

K Nearest Neighbors

Key Intuition: Examples that are “close” to each other should have similar labels to each other

Given dataset (X, y)

To predict a label for a new example input *x*:

Find the closest *k* examples in X

use the labels of those *k* examples to label *x*

In classification (i.e., discrete labels): Use majority voting

In regression (i.e., continuous labels): Use average of *k* labels

iris

3 species (i.e., classes) of iris

- Iris setosa
- Iris versicolor
- Iris virginica

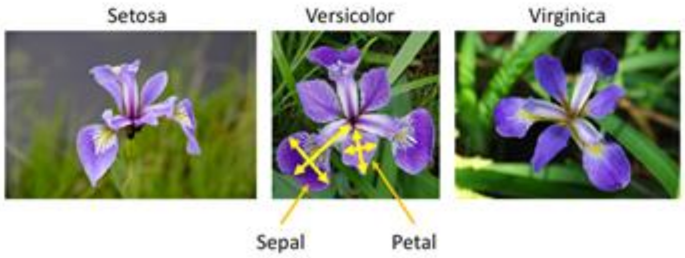


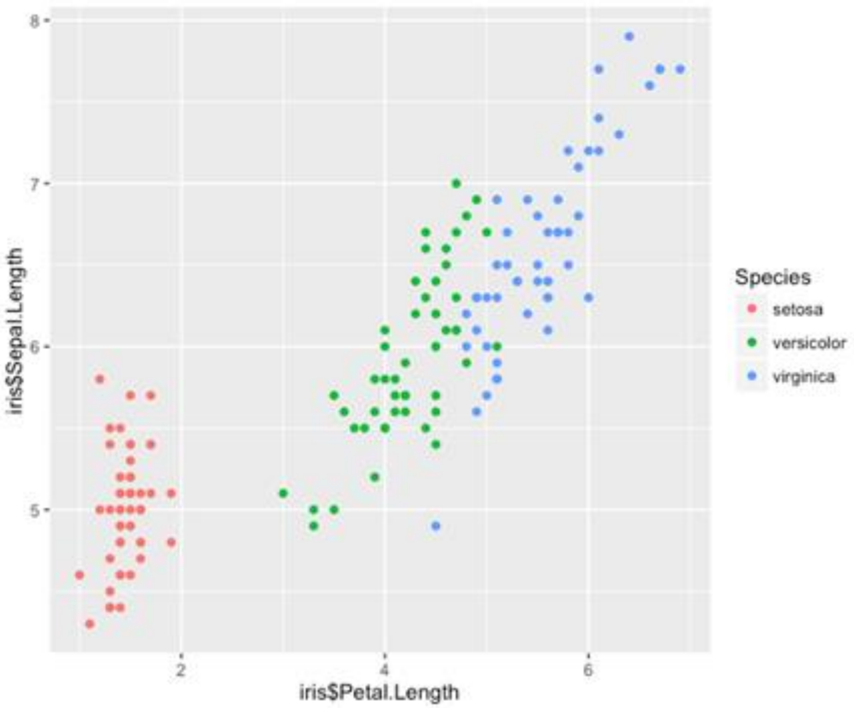
Image Source

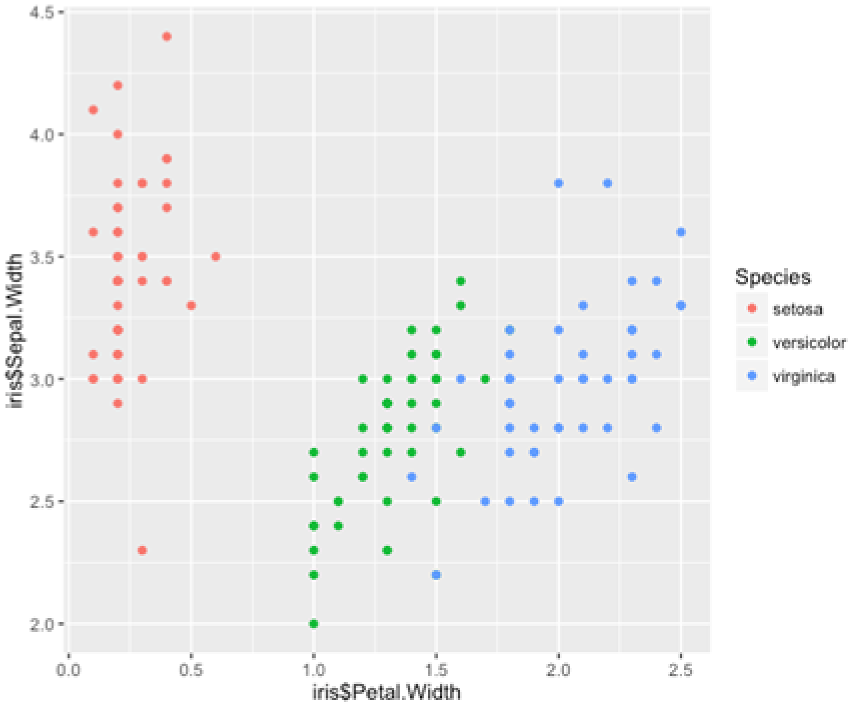
iris

- 50 observations per species
- 4 variables per observation
  - Sepal length
  - Sepal width
  - Petal length
  - Petal width

Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa

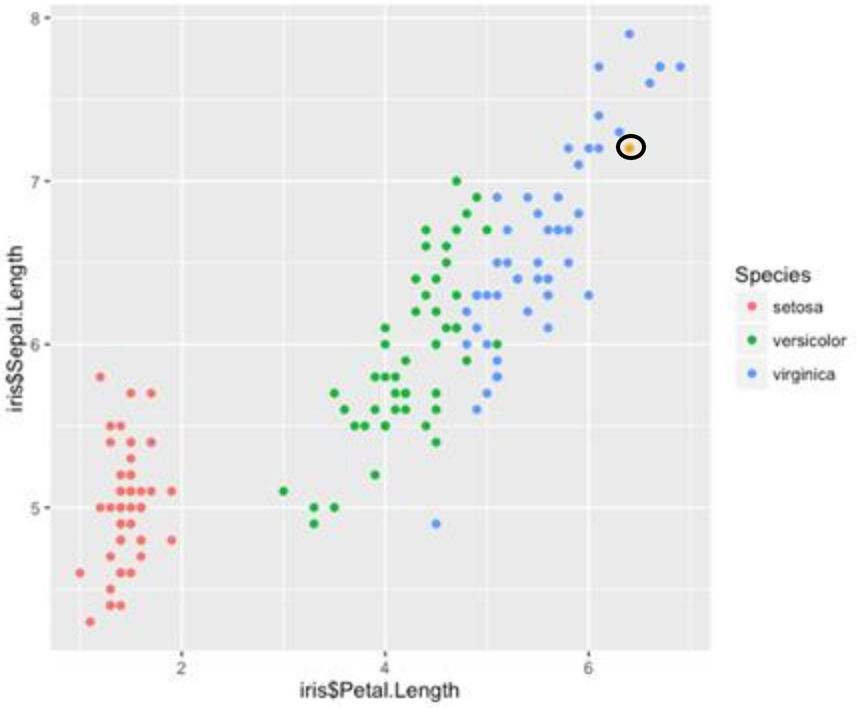
Visualizing the data





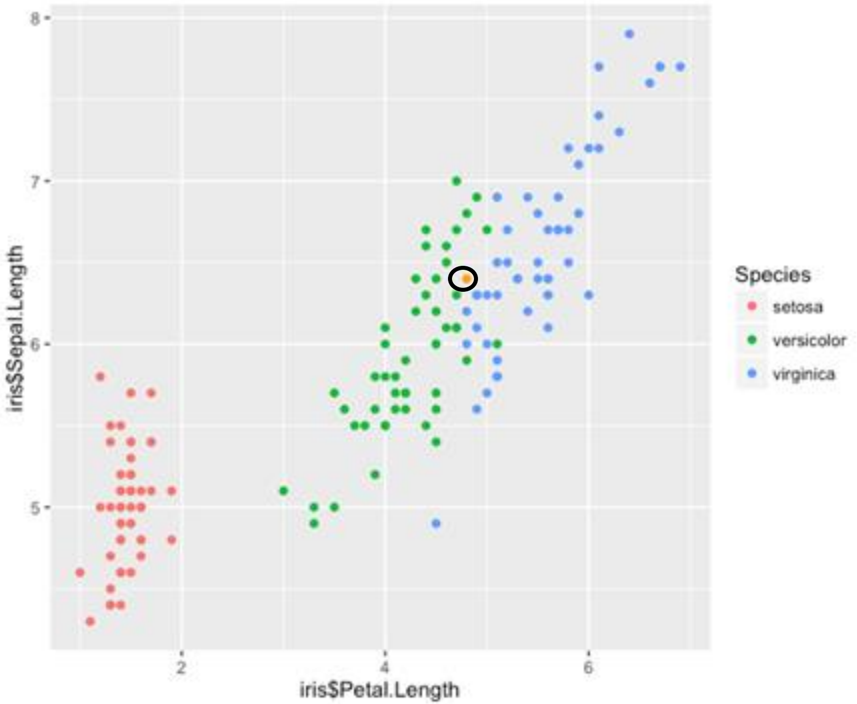
A new observation

```
new_point <- data.frame
    (Sepal.Length = 7.2,
     Sepal.Width  = 3.2,
     Petal.Length = 6.4,
     Petal.Width  = 2.4)
```



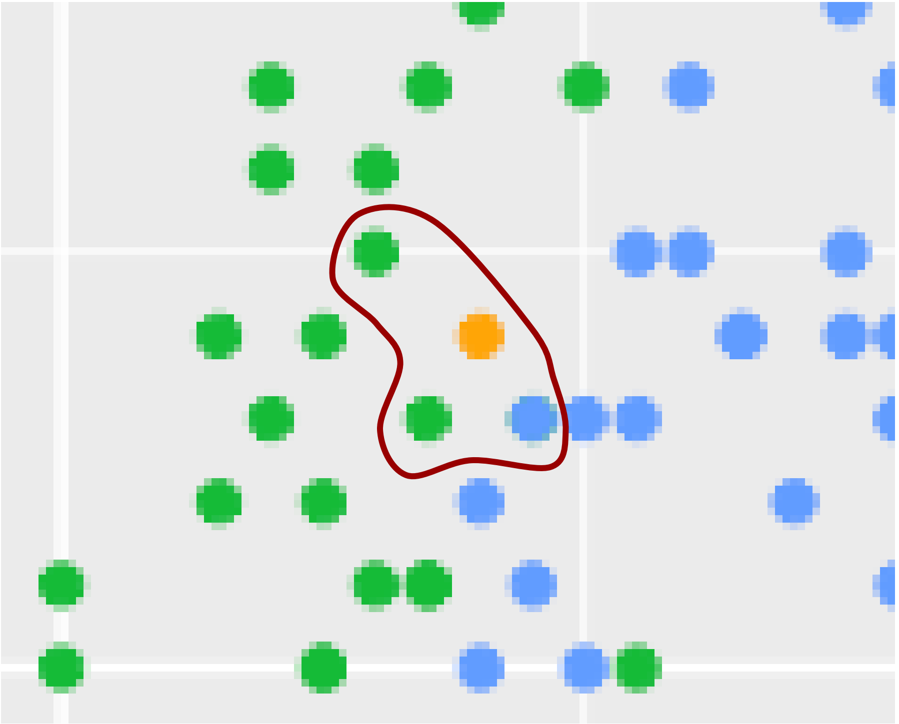
Another observation

```
new_point <- data.frame
    (Sepal.Length = 6.4,
     Sepal.Width  = 2.8,
     Petal.Length = 4.9,
     Petal.Width  = 1.3)
```



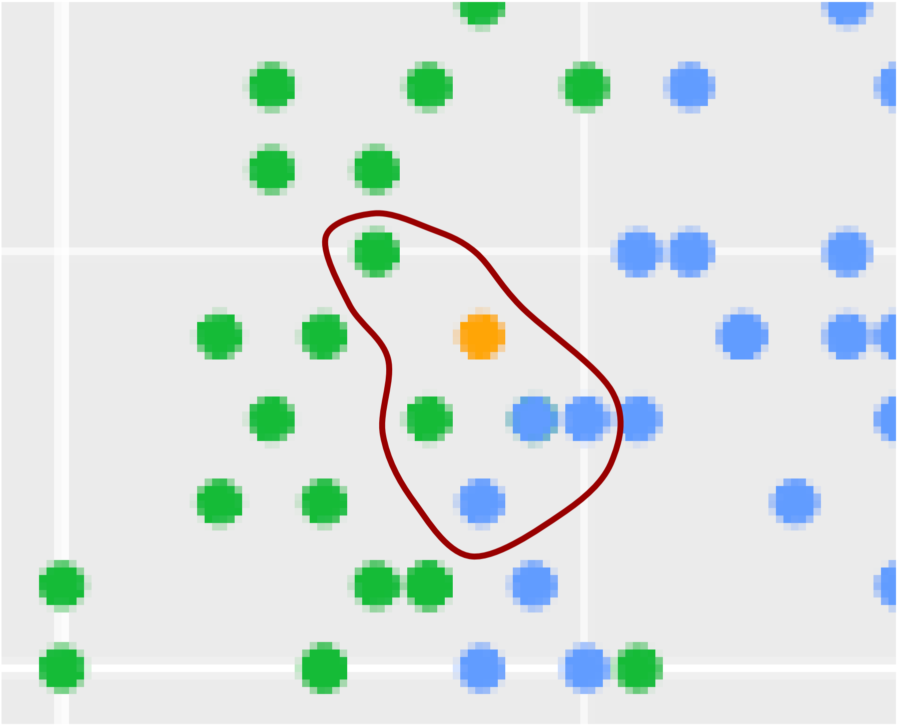
Another observation

```
[1] k = 3
[1] versicolor
```



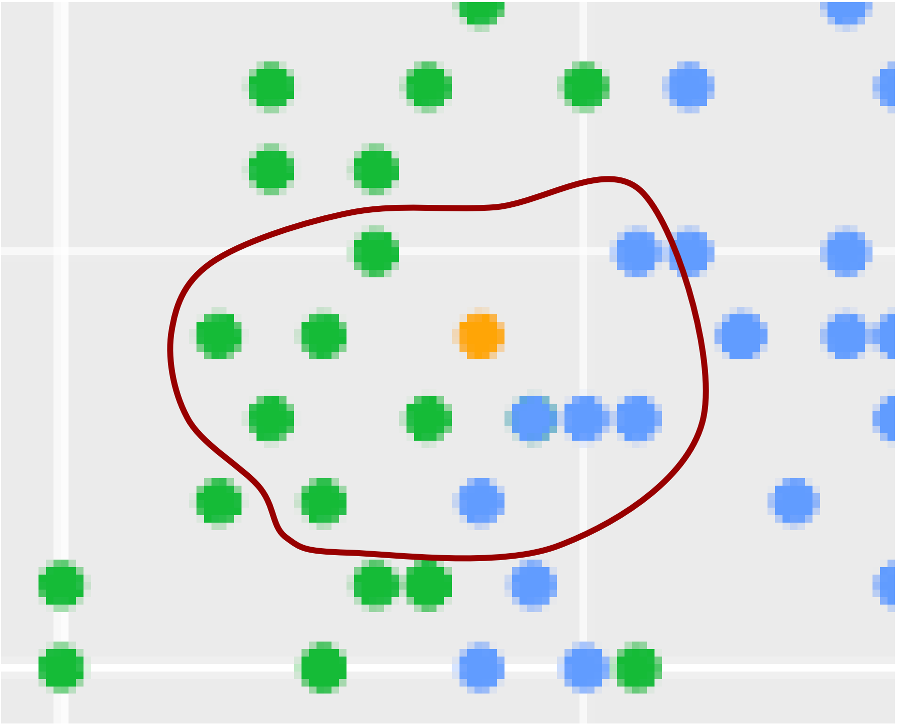
Another observation

```
[1] k = 3
[1] versicolor
[1] k = 5
[1] virginica
```

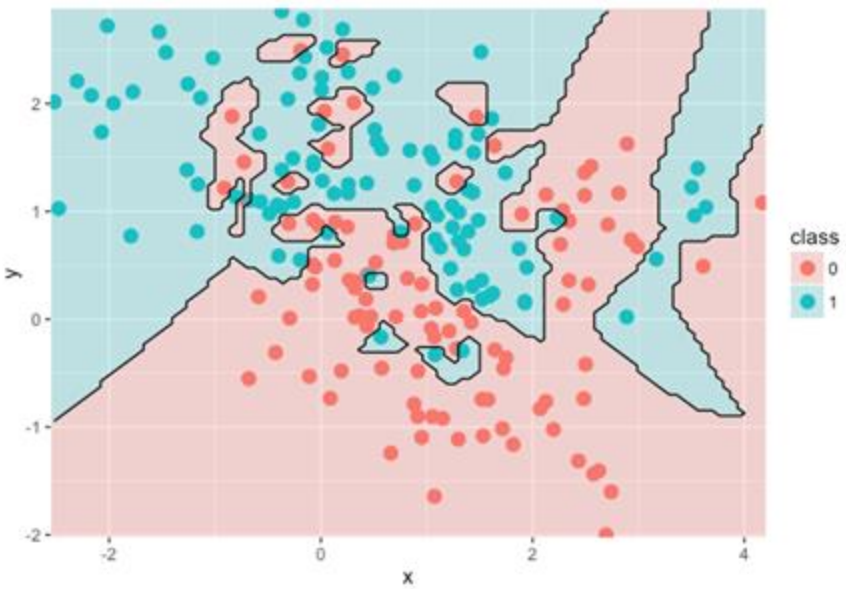


Another observation

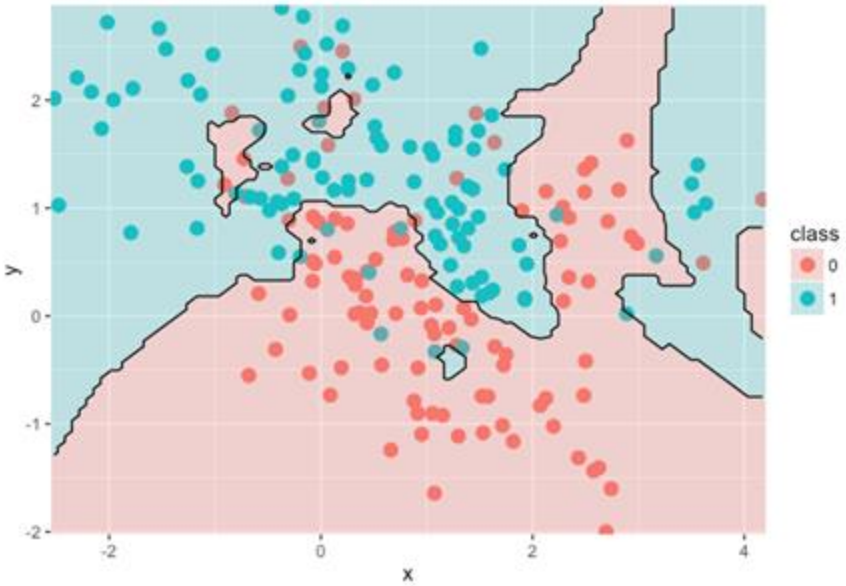
```
[1] k = 3
[1] versicolor
[1] k = 5
[1] virginica
[1] k = 11
[1] versicolor
```



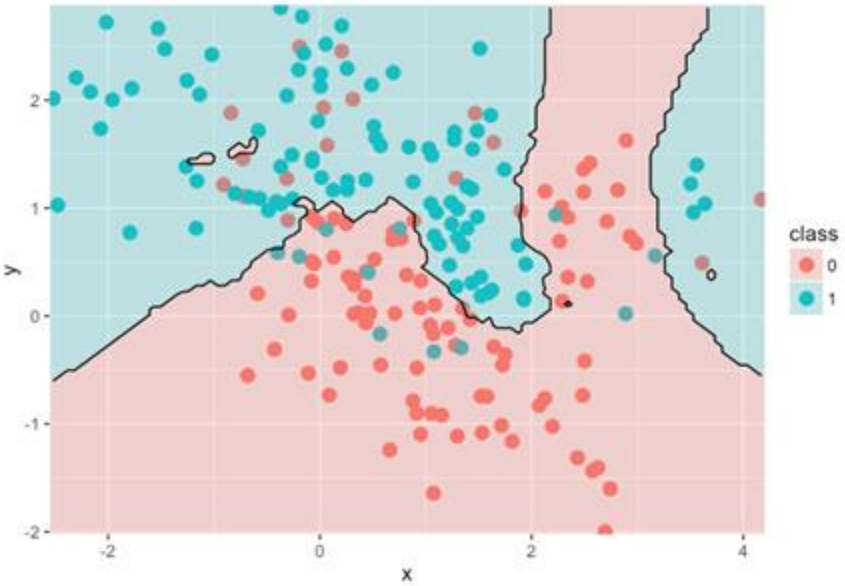
The choice of *k* has a large impact on model outputs *k* = 1



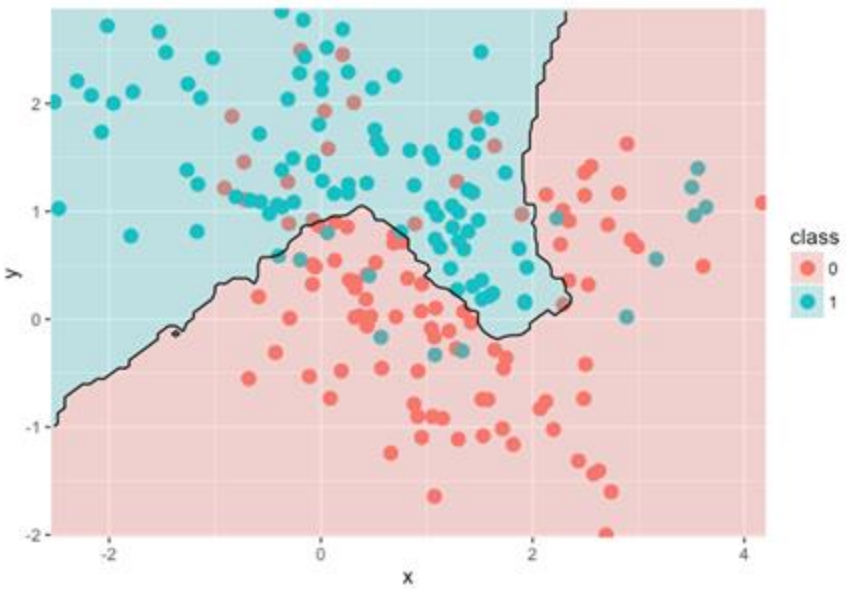
$k = 3$



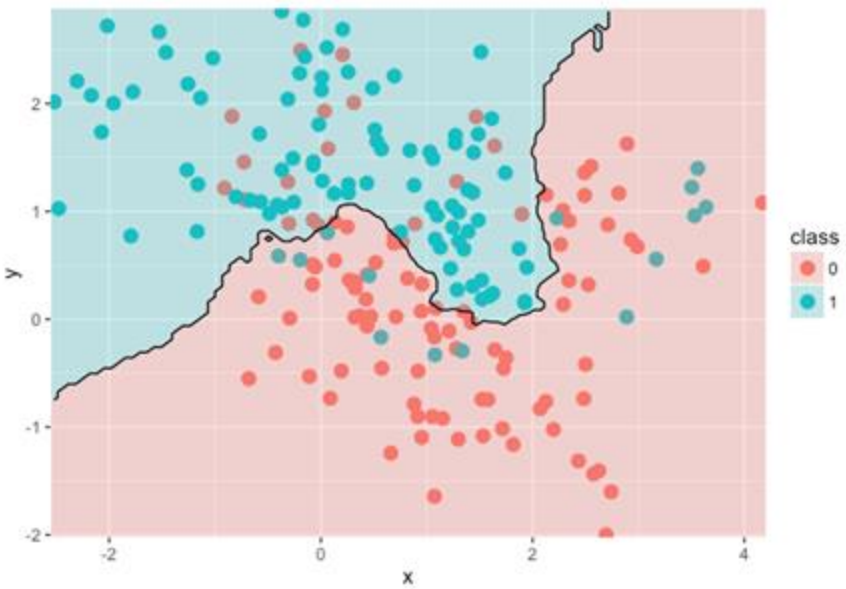
$k = 7$



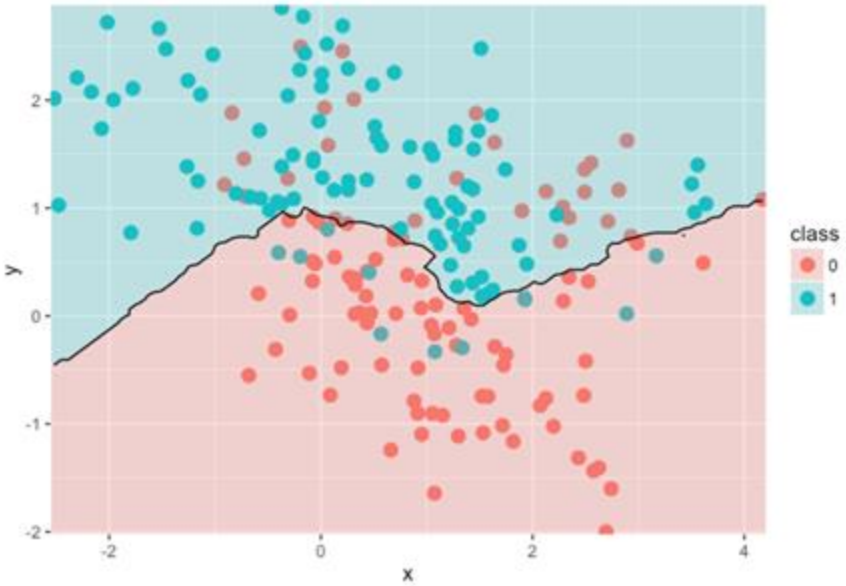
$k = 15$



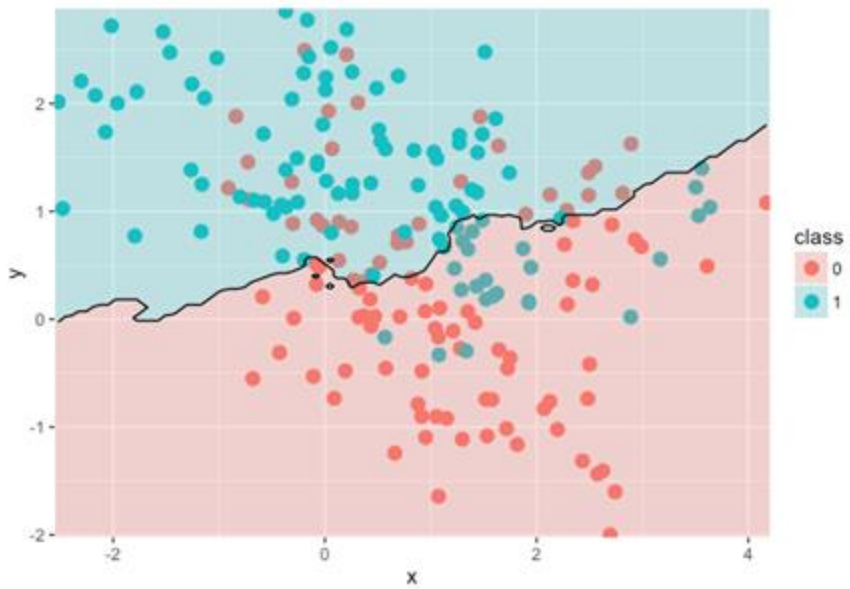
$k = 25$



k = 51



k = 101



What's the best *k*?

Idea: try out different values of *k*, use the best one

But what does it mean to be best? What is our objective?
What does it mean to have a "good" function approximation?

Idea: We'd like to optimize for accuracy (in classification).

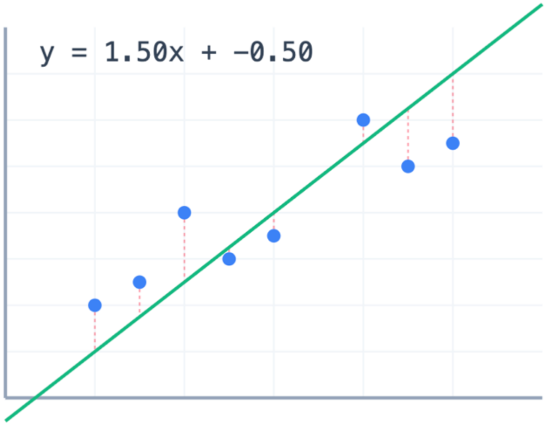
Let's check how the model would classify each example input in X and choose k with highest accuracy

The goal of machine learning is to learn a function approximation using X, and use that function approximation in the **real world** on many **new examples**

What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

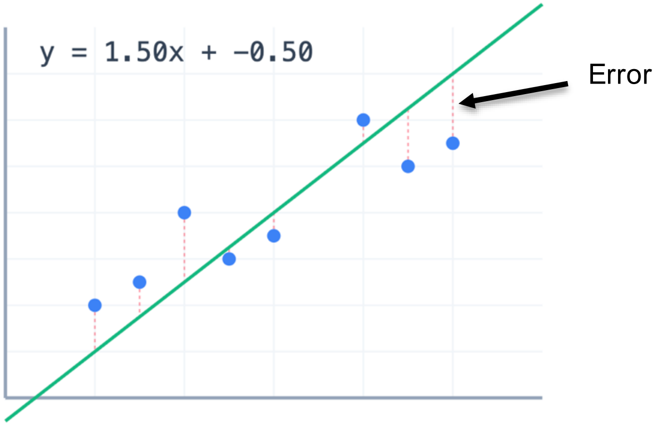
The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

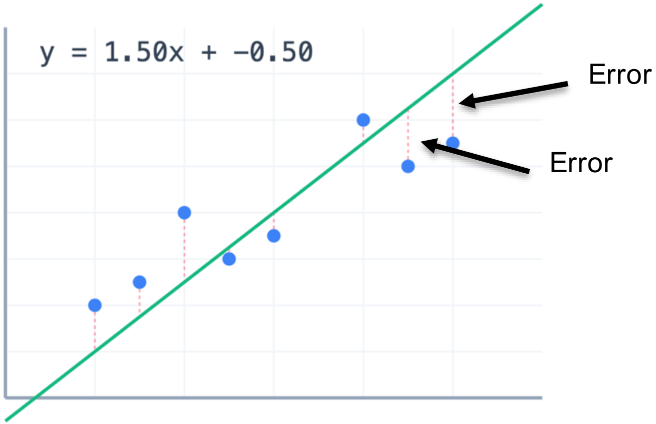
The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

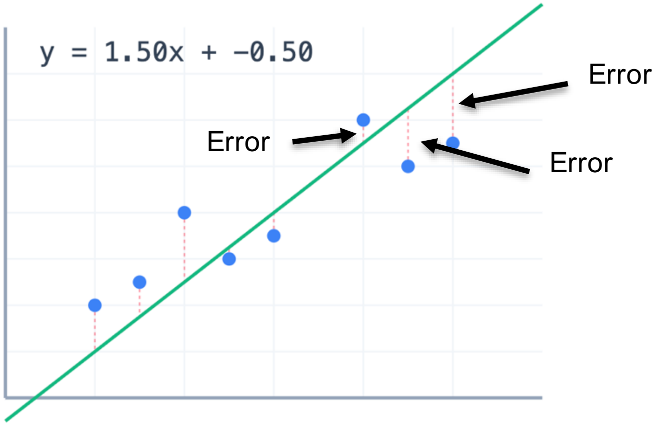
The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

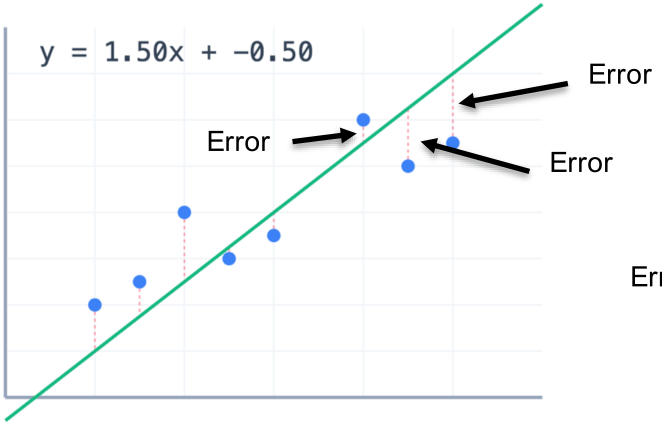
The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

The standard loss function for Linear Regression is **Mean Squared Error (MSE)**



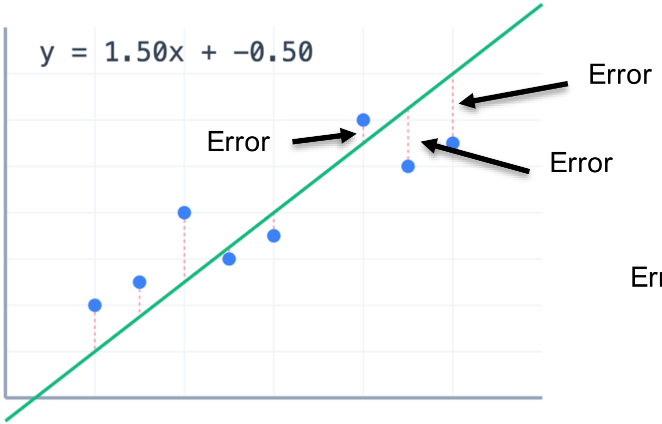
Error = $f\left(x^{(i)}\right) - y^{(i)}$

What makes a good approximation?

Loss Function: A function that describes how closely our approximation matches our data

The standard loss function for Linear Regression is **Mean Squared Error (MSE)**

$$MSE = \frac{\sum_i^n \left(f\left(x^{(i)}\right) - y^{(i)}\right)^2}{n}$$



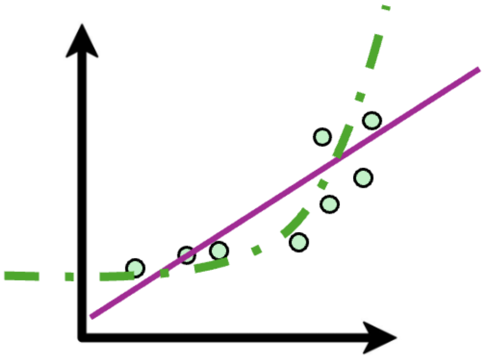
Error = $f\left(x^{(i)}\right) - y^{(i)}$

How to know which function is the best?

$\mathbb{X}$

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
$x^{(5)}$
$x^{(6)}$
$x^{(7)}$
$x^{(8)}$

f_1 or f_2?

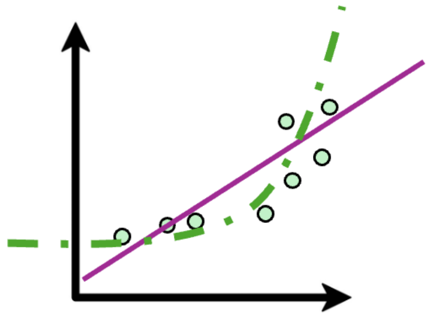


How to know which function is the best?

- $=$ $\mathbb{X}$

$x^{(1)}$
$x^{(2)}$
$x^{(3)}$
$x^{(4)}$
$x^{(5)}$
$x^{(6)}$
$x^{(7)}$
$x^{(8)}$

f_1 or f_2?



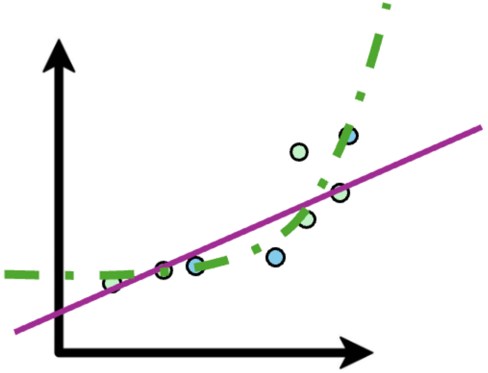
Compare MSE between them?

How to know which function is the best?

$\mathbb{X}$

$x^{(1)}$	
$x^{(2)}$	
$x^{(3)}$	Training Set
$x^{(4)}$	
$x^{(5)}$	
$x^{(6)}$	
$x^{(7)}$	Test Set
$x^{(8)}$	

f_1 or f_2?

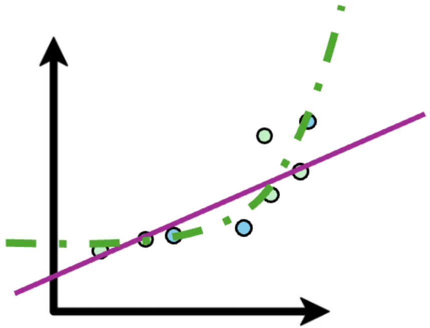


How to know which function is the best?

$\mathbb{X}$

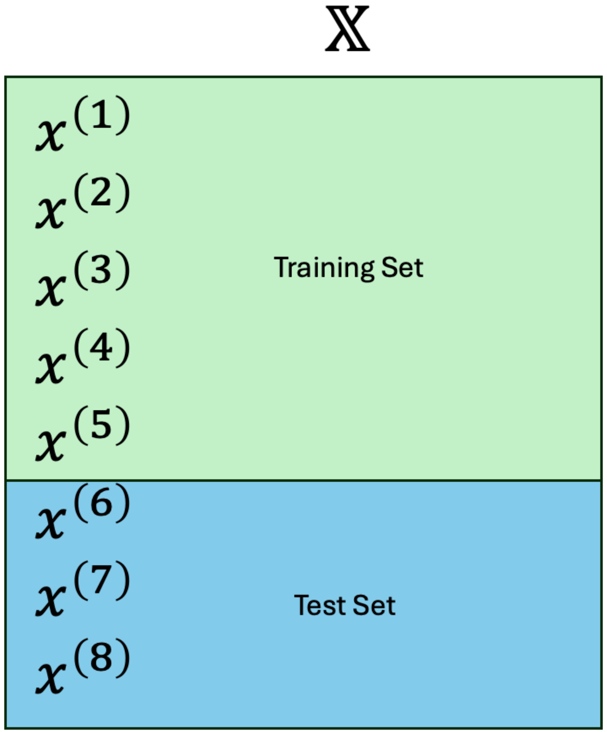
$x^{(1)}$	
$x^{(2)}$	
$x^{(3)}$	Training Set
$x^{(4)}$	
$x^{(5)}$	
$x^{(6)}$	
$x^{(7)}$	Test Set
$x^{(8)}$	

f_1 or f_2?

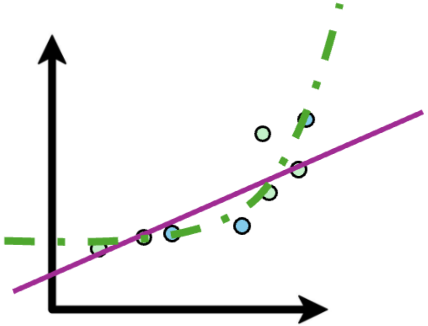


Compare MSE on what data?

How to know which function is the best?



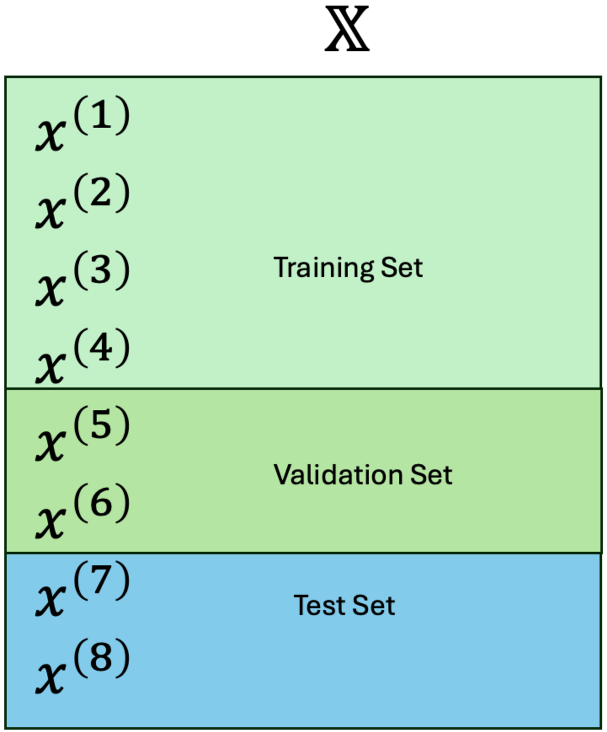
f_1 or f_2?



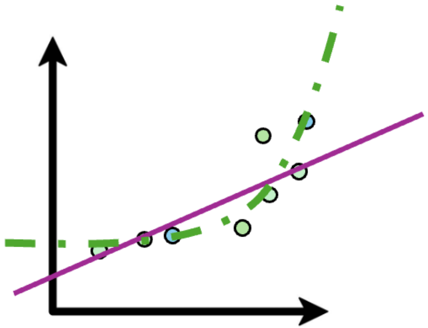
Compare MSE on what data?

When might we want to overfit?

How to know which function is the best?



f_1 or f_2?



1. Train model on training set
2. Validate performance on validation set
3. Report results on test set

Data Sets

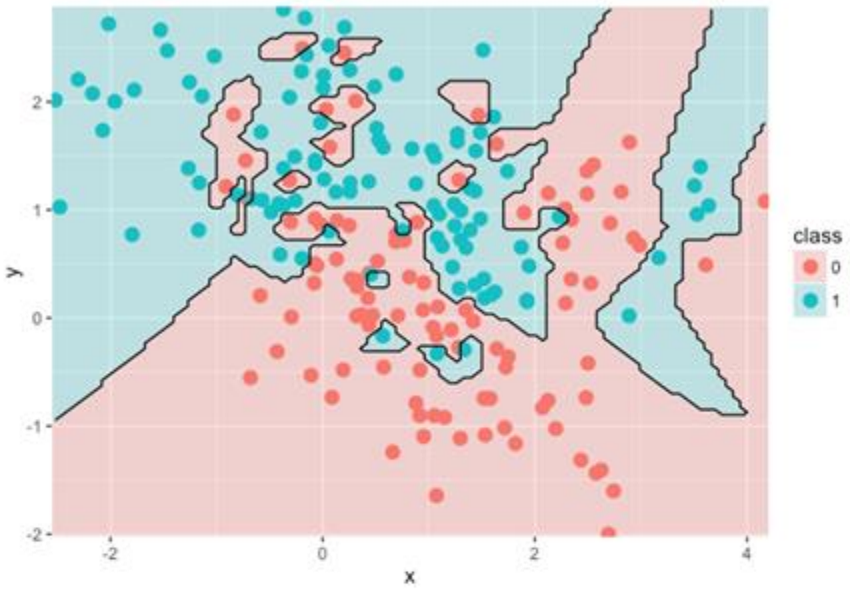
- **Training Set**: Used to adjust parameters of model
- ***Validation set*** — used to test how well we're doing as we develop
  - Prevents ***overfitting***
- ***Test Set*** — used to evaluate the model once the model is done

In k-NN:

Nearest neighbors are found in Training Set
Validation set is used to test model's performance on "new" examples
Test set is used to measure final performance of model

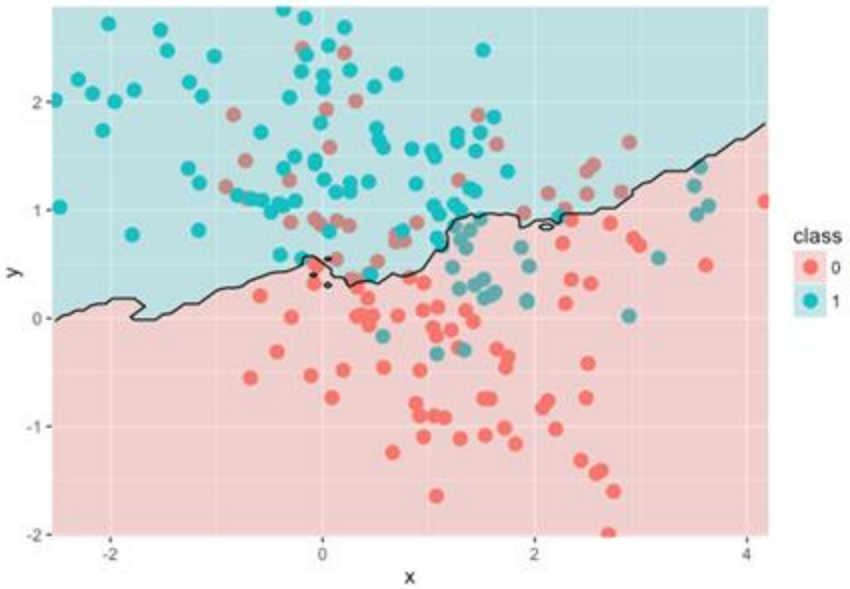
Small *k*

- $k = 1$
- Low bias
- High variance:
  model varies greatly with the data
- Models like this are overfit
  The model reads too much into the data, extrapolating based on randomness, rather than relevant feature values

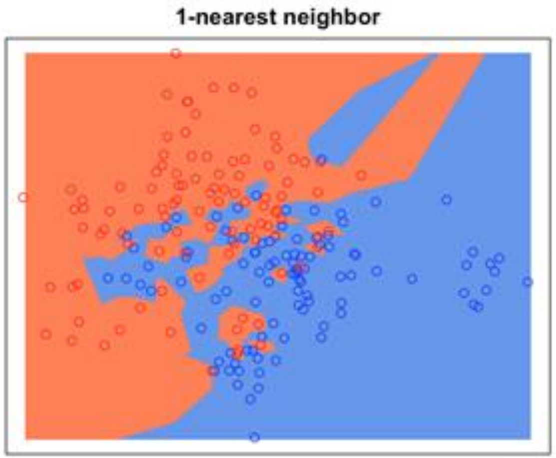


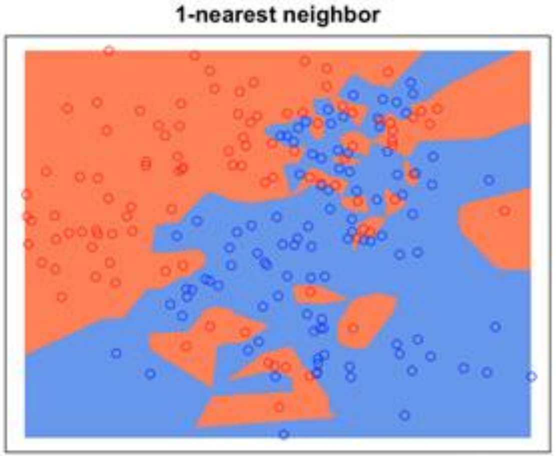
Large *k*

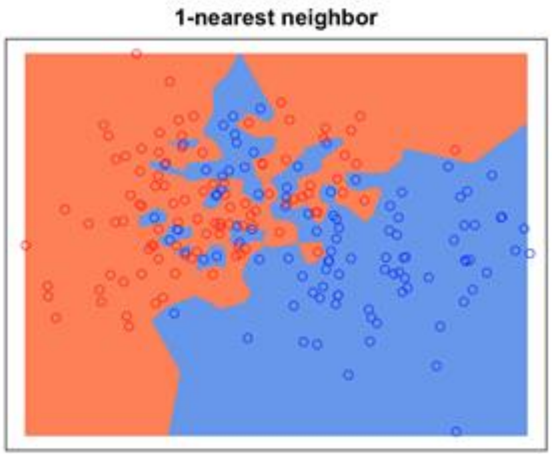
- k = 101
- High bias: systematic errors
- Low variance:
  model barely varies with the data
- Rather than being overfit, this model is underfit
  The decision boundary doesn't capture enough of the relevant information encoded in the data



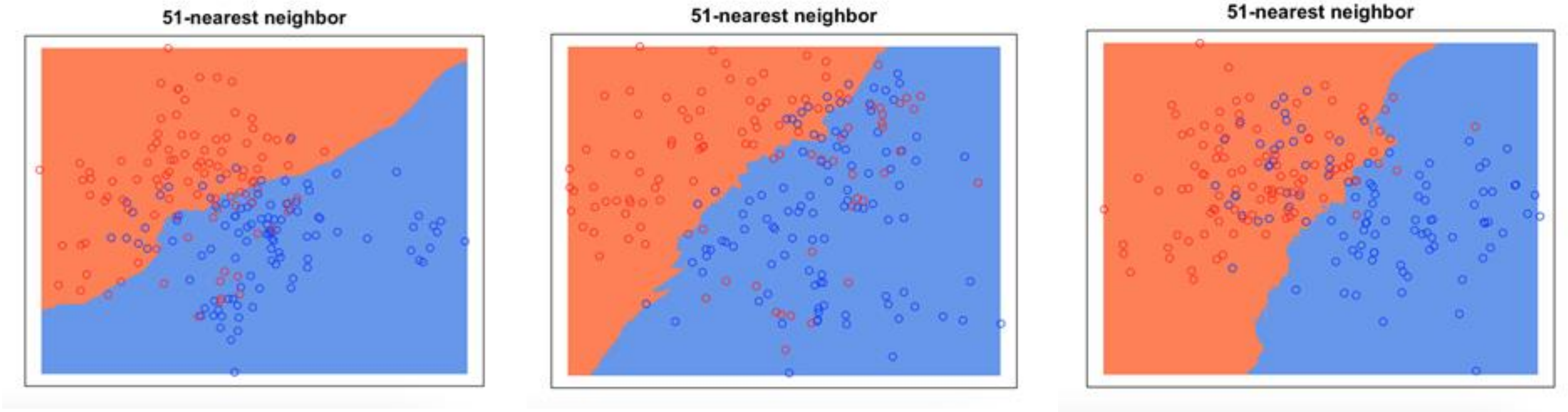
$k = 1$







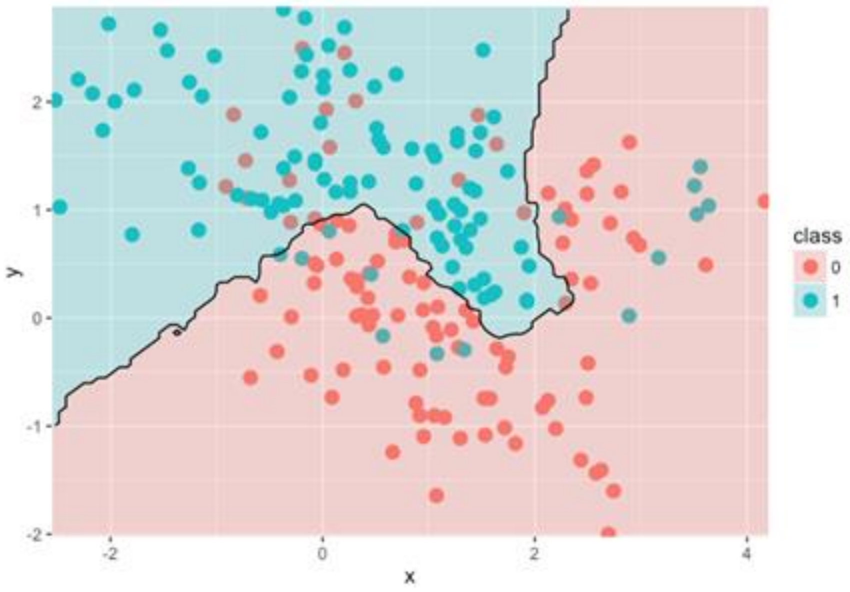
$k = 51$


51-nearest neighbor
51-nearest neighbor
51-nearest neighbor

Model Selection

Find a model that balances the bias-variance tradeoff.

- A high value of *k* correspond to a high degree of bias, but contains the variance
- With low values of *k*, the jagged decision boundaries are a sign of high variance
- *k* = 15 seems “just right”



k-NN caveats

- *k*-NN can be very slow, especially for very large data sets
    - *k*-NN is not a learning algorithm in the traditional sense, because it doesn't actually do any learning: i.e., it doesn't preprocess the data
    - Instead, when it is given a new observation, it calculates the distance between that observation and every existing observation in the data set
- *k*-NN works better with quantitative data than categorical data
    - Data must be quantitative to calculate distances
    - So categorical data must be transformed
- Without clusters in the training data, *k*-NN cannot work well