Plan for the week

Classification

- M: *k*-nearest neighbors
- W: Decision Trees
- Model Selection
  - Bias-Variance Tradeoff
  - Cross Validation
- Friday Section
  - Bayes' Rule
  - HW 2 Review

Refresher:
Overview of Machine Learning

Types of Machine Learning

	Supervised learning	**Unsupervised learning**
Discrete	Classification	Clustering (Dimensionality Reduction)
Continuous	Regression	Density Estimation

Image Source

Supervised Learning (a.k.a. Function approximation)

- Learning from labeled data
- This labeled data is called training data, and consists of observations and corresponding labels (*a.k.a.* ground truth).
- The goal of a supervised learning algorithm is to approximate a function that generalizes well to unseen examples.
- Supervised algorithms are typically evaluated on test data, which is distinct from training data.

Supervised Learning

Given a labeled set of training data $D = \{(x_i, y_i) \mid i = 1, \ldots, n\}$, where each x_i is an instance and each y_i is a label, the goal of a supervised learner is to learn a function from instances to labels, so that it can appropriately label instances in the test data

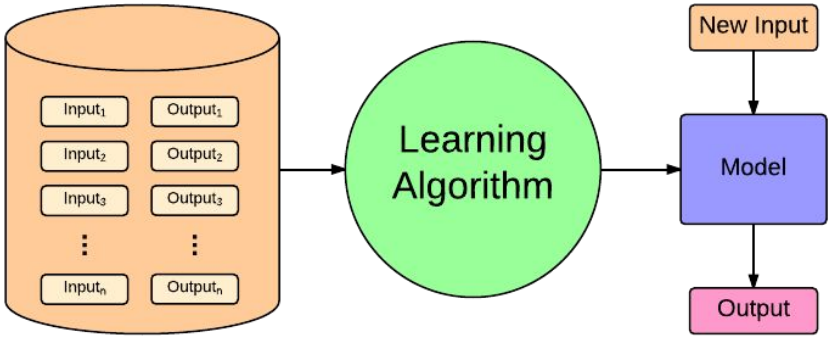


Image Source

Supervised Learning

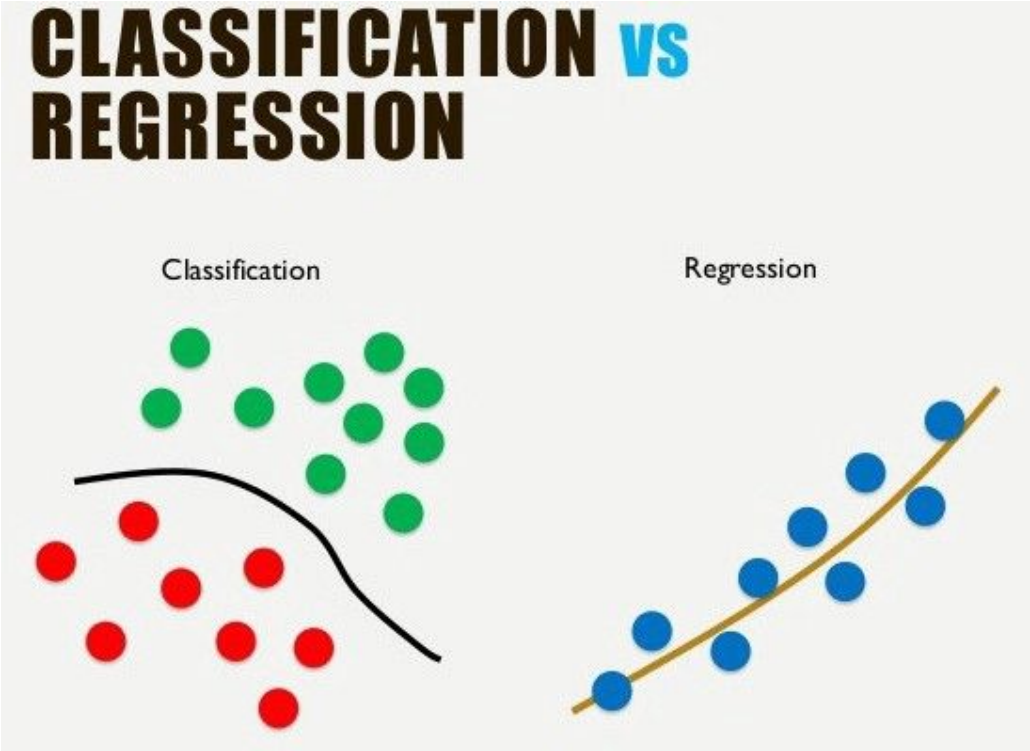


Image Source

Classification

- Supervised learning when labels are categorical
- Binary classification: Spam or not? Malignant or benign?
- Multiclass classification: Part-of-speech tagging,
  Face recognition, Spell checking.
- Popular algorithms:
  - Logistic Regression
  - Naive Bayes
  - Decision trees (easily adopted to regression, as well)
  - *k* nearest neighbor (easily adopted to regression, as well)

Errors

Evaluation Metrics

The results of a binary classification task can be depicted in matrix form, in a confusion matrix

- Accuracy is the ratio of correct classifications to all classifications: (TP + TN) / (P + N)
- Error is the ratio of *in*correct classifications to all classifications: (FP + FN) / (P + N)

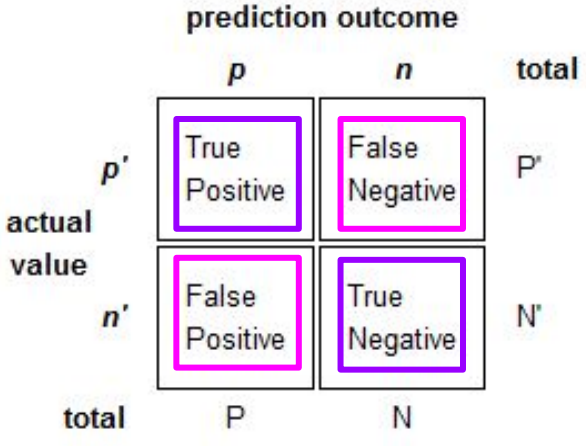


Image Source

Evaluation Metrics

The results of a binary classification task can be depicted in matrix form, in a confusion matrix

- Sensitivity is the true positive rate: proportion of positives correctly identified as such (TP / (TP + FN))
  - Also called recall or the hit-rate
  - Remember: FNs are positive!
- Specificity is the true negative rate: proportion of negatives correctly identified as such (TN / (FP + TN))
  - The opposite of specificity (FP / (FP + TN)) is called the fall-out rate
  - Remember: FPs are negative!
- Precision is the proportion of true positives among all predicted positives (TP / (TP + FP))
- The F_1 score is the harmonic mean of precision and recall:
  I.e., 2 / ((1 / precision) + (1 / recall)) = 2 (precision · recall) / (precision + recall)

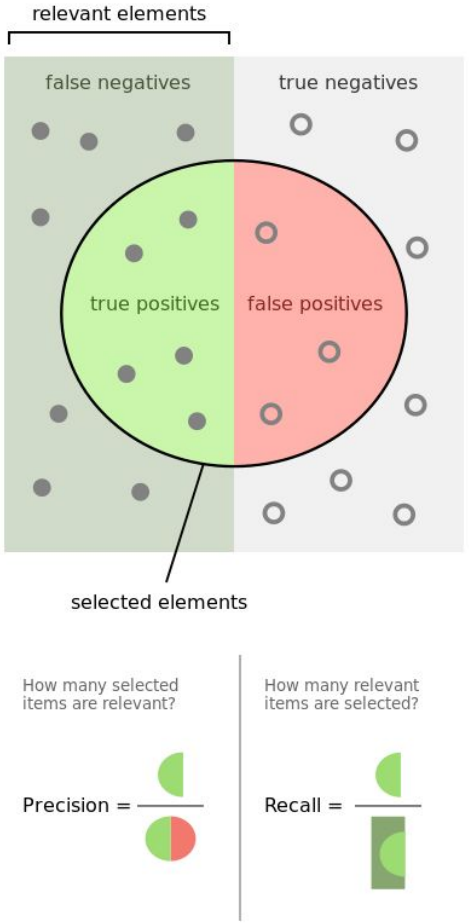


Image Source

k Nearest Neighbors

k Nearest Neighbors

To classify an observation:

- Look at the labels of some number, say *k*, of neighboring observations.
- The observation is then classified based on its nearest neighbors' labels, using, for example, majority vote.

There is no learning phase.

Image Source

Example

- Let’s try to classify the unknown green point by looking at *k* = 3 and *k* = nearest neighbors
- For *k* = 3, we see 2 triangles and 1 square; so we might classify the point as a triangle
- For *k* = 5, we see 2 triangles and 3 squares; so we might classify the point as a square
- Typically, we classify by some variant of majority vote, so use an odd value of *k* to avoid ties

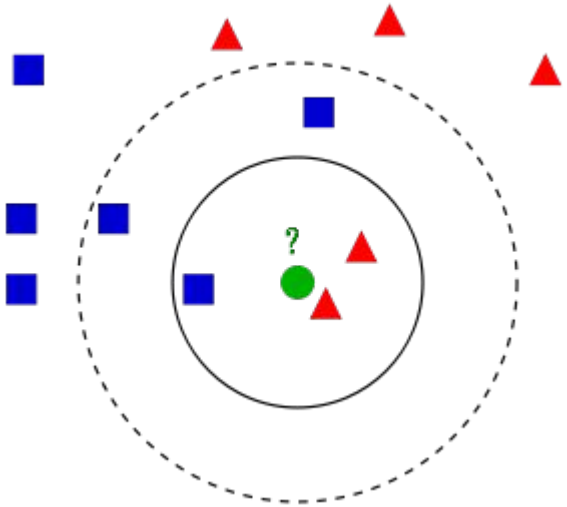


Image Source

Classifying `iris`

We're going to demonstrate the use of *k*-NN on the `iris` data set (the flower, not the part of your eye)

```
> iris
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
4            4.6         3.1          1.5         0.2     setosa
5            5.0         3.6          1.4         0.2     setosa
6            5.4         3.9          1.7         0.4     setosa
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
14           4.3         3.0          1.1         0.1     setosa
15           5.8         4.0          1.2         0.2     setosa
```

Iris setosa

Iris versicolor

Iris virginica

iris

- 3 species (i.e., classes) of iris
  - Iris setosa
  - Iris versicolor
  - Iris virginica
- 50 observations per species
- 4 variables per observation
  - Sepal length
  - Sepal width
  - Petal length
  - Petal width

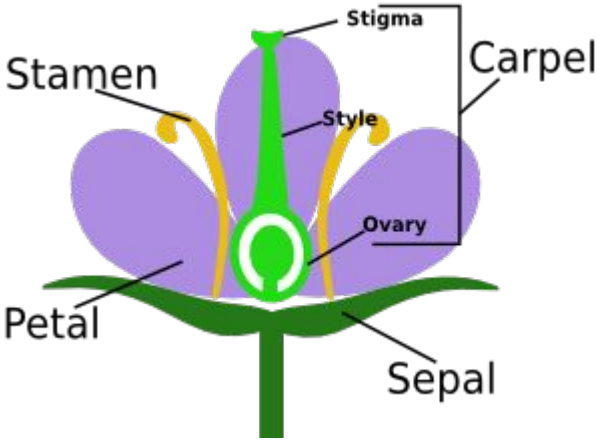
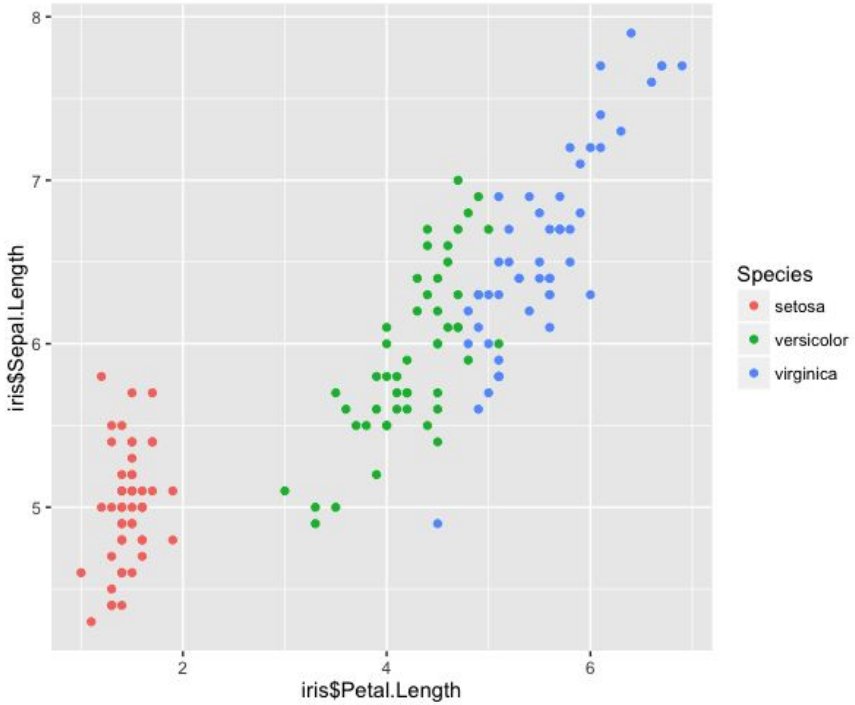
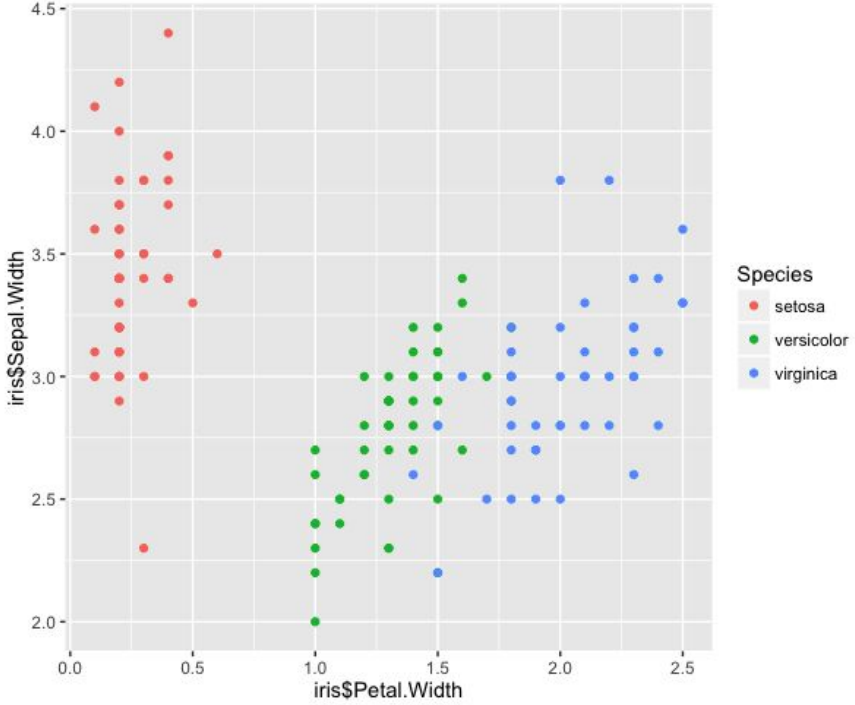


Image Source

Visualizing the data



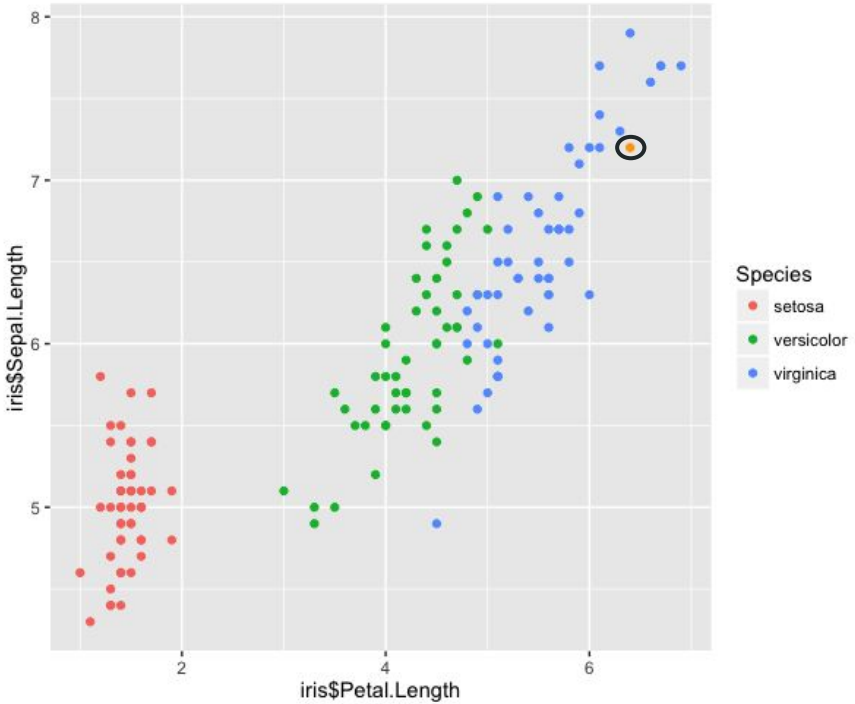


A new observation

- New orange point

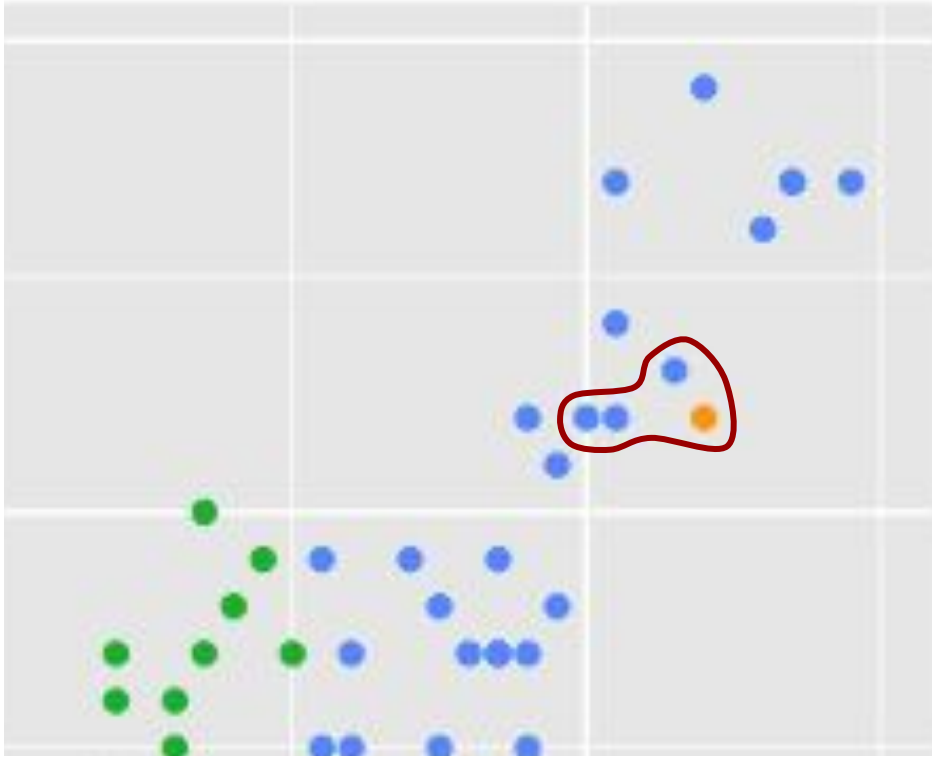
```
new_point <- data.frame
   (Sepal.Length = 7.2,
    Sepal.Width  = 3.2,
    Petal.Length = 6.4,
    Petal.Width  = 2.4)
```

- Let's run *k*-NN in R with *k* = 3



How does it work?

- Since *k* = 3, R finds the new point's closest 3 neighbors
- These are all virginica, so it's easy to classify the new point
- The new observation also gets classified as virginica

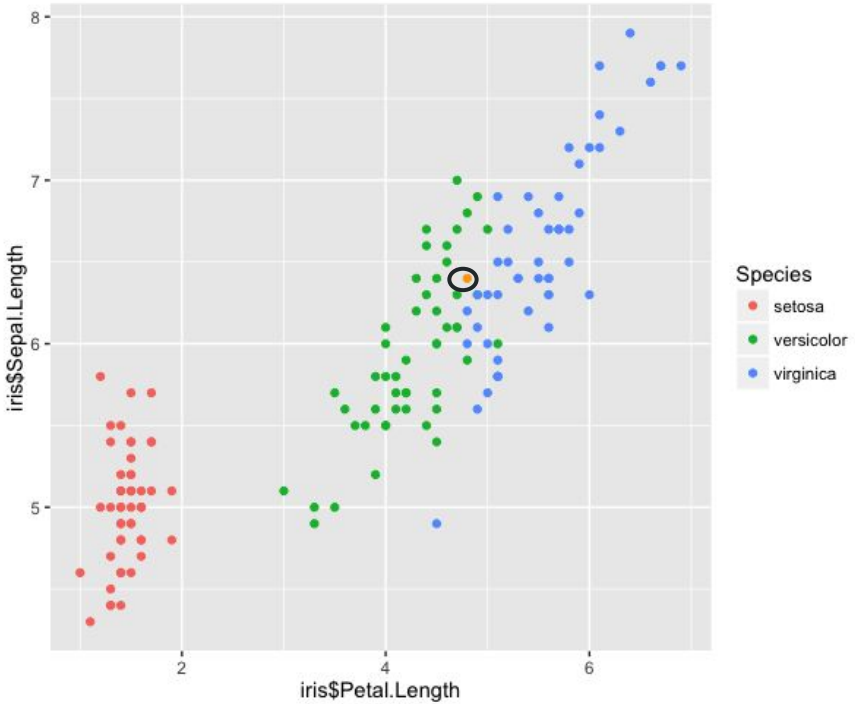


Another observation

- This one’s a bit more ambiguous

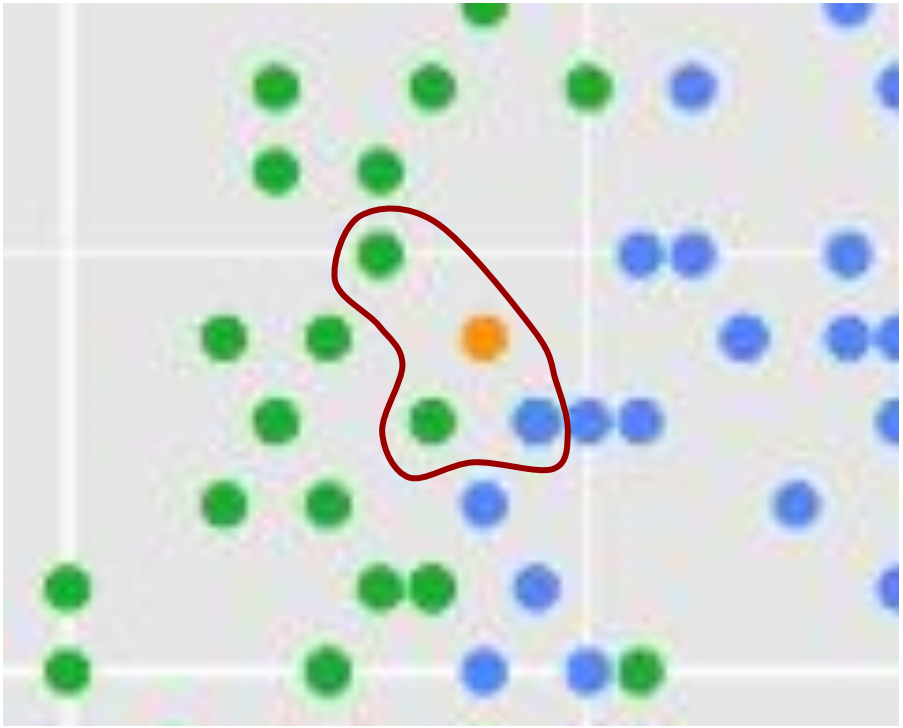
```
new_point <- data.frame
    (Sepal.Length = 6.4,
     Sepal.Width  = 2.8,
     Petal.Length = 4.9,
     Petal.Width  = 1.3)
```

- What should we expect?



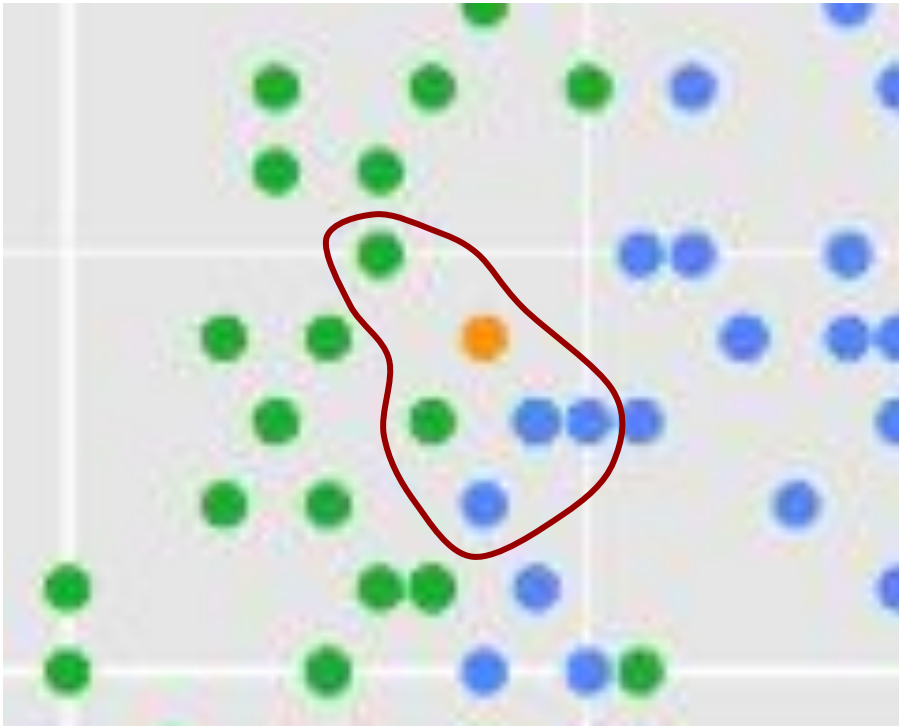
Another observation

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



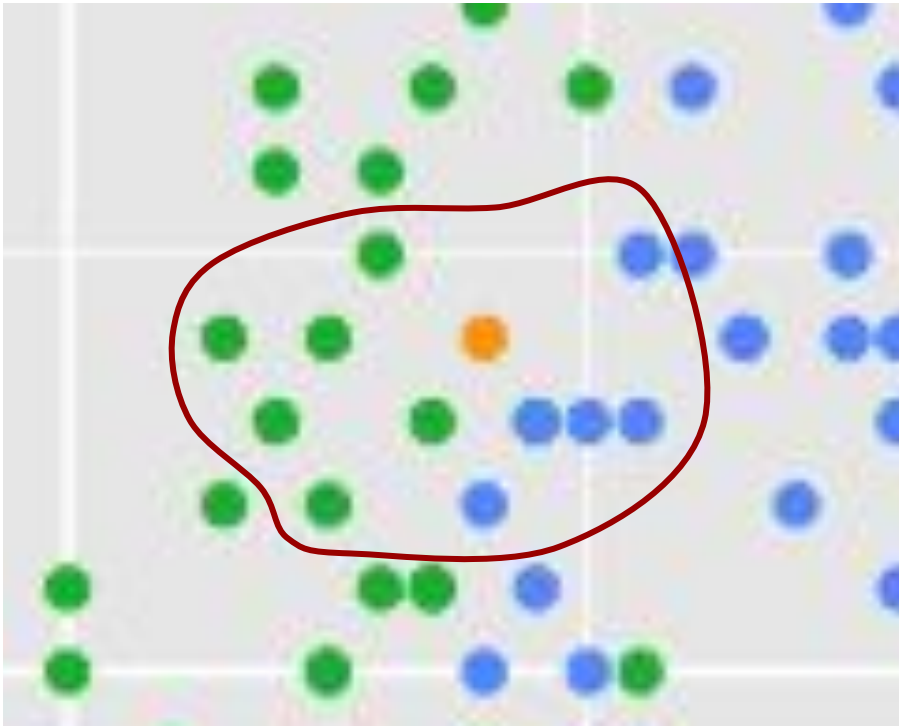
Another observation

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



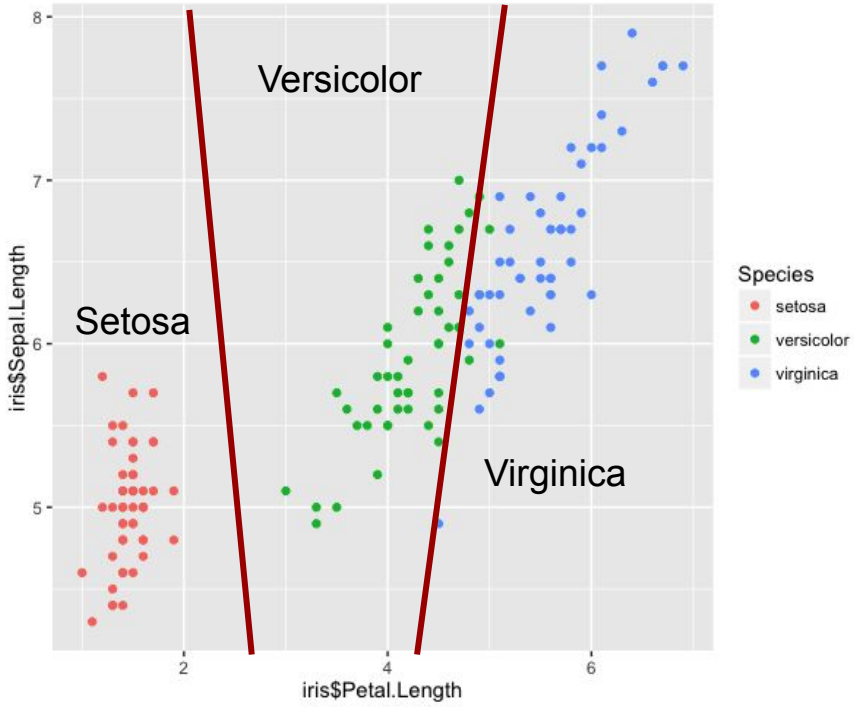
Another observation

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



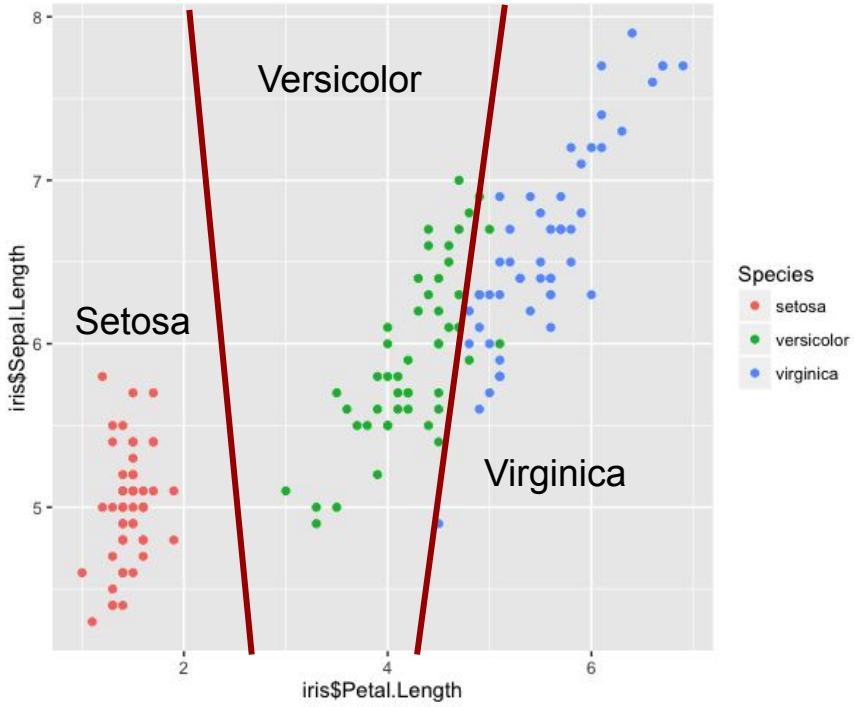
Decision Regions

- Decision regions are regions where observations are classified one way or another
- For `iris`, there are three species, and three (approximate) decision regions



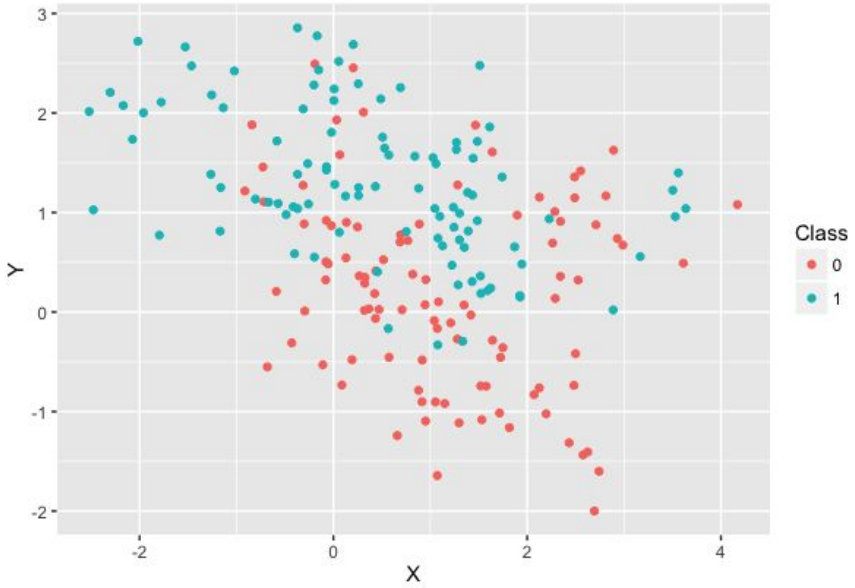
Decision Boundaries

- (Linear) decision boundaries are lines that separate decision regions
- On the boundaries, classifiers may give ambiguous results
- Changing parameters, like *k* in *k*-NN, changes both the decision regions and their boundaries
- Let's vary *k* and see what happens

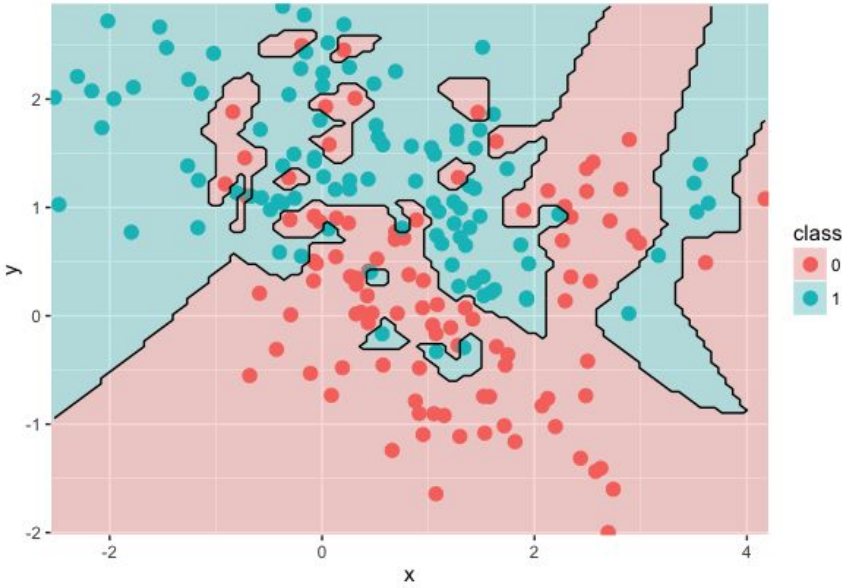


Decision Boundaries

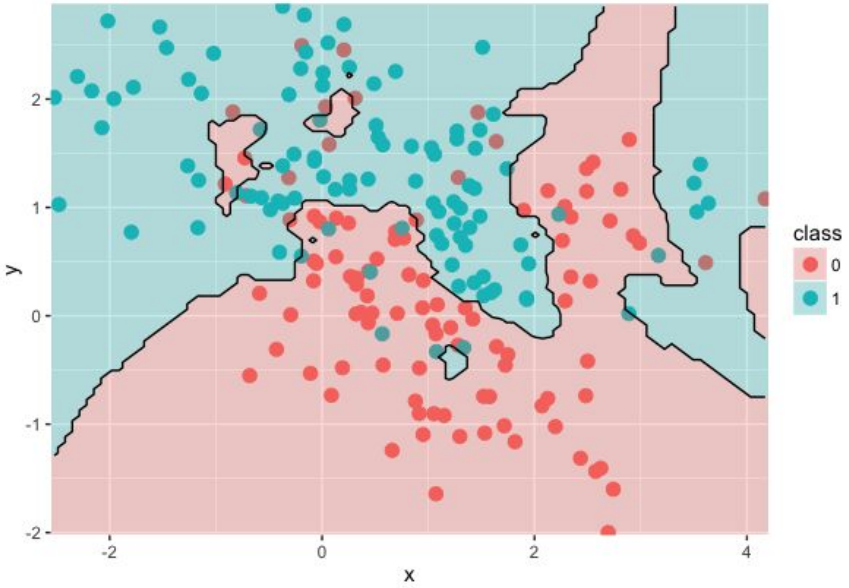
- Let’s take a closer look at decision boundaries for *k*-NN
- To do so, let’s use a new data set
- Here are some random data, classified as either 0 or 1
- These two classes overlap quite a bit, compared to the `iris` data



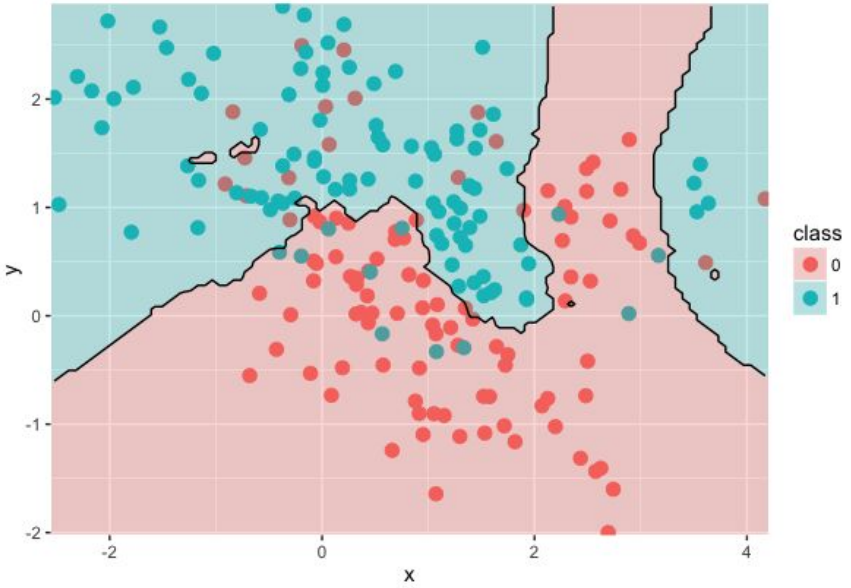
$k = 1$



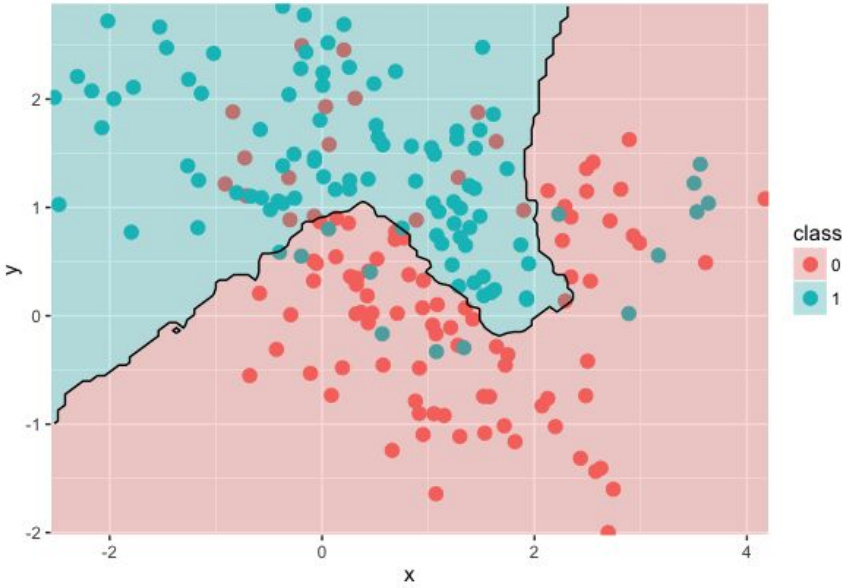
$k = 3$



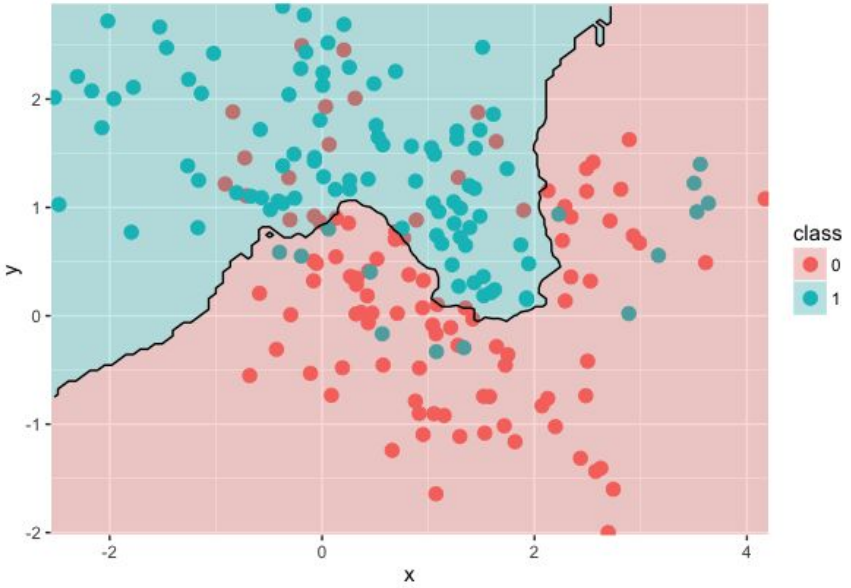
$k = 7$



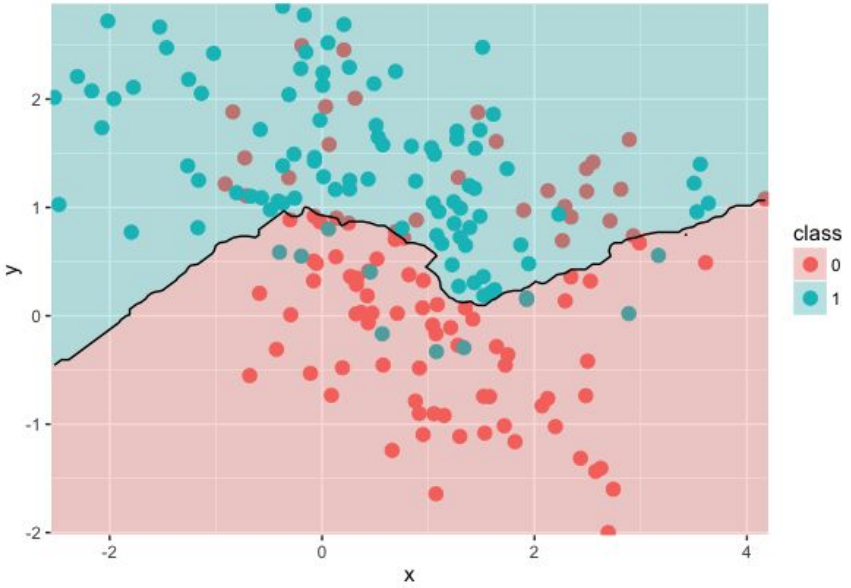
$k = 15$



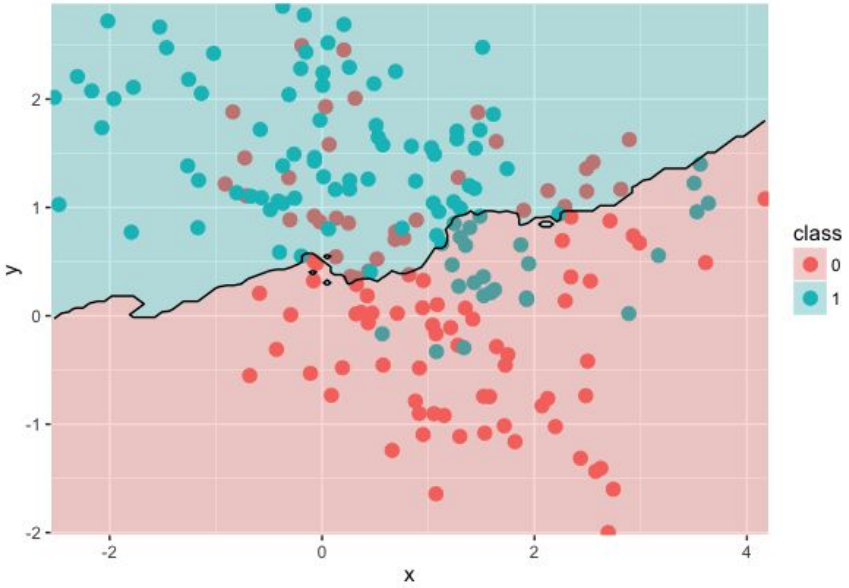
$k = 25$



$k = 51$

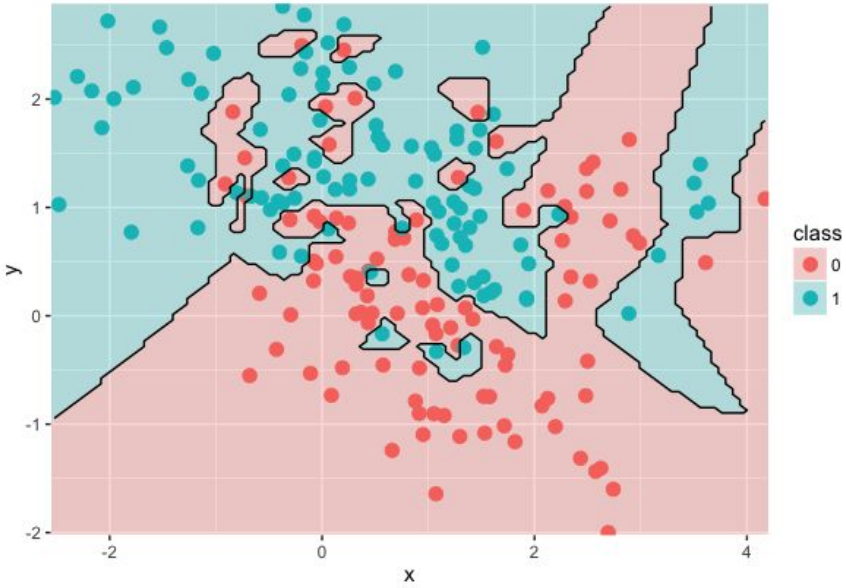


$k = 101$



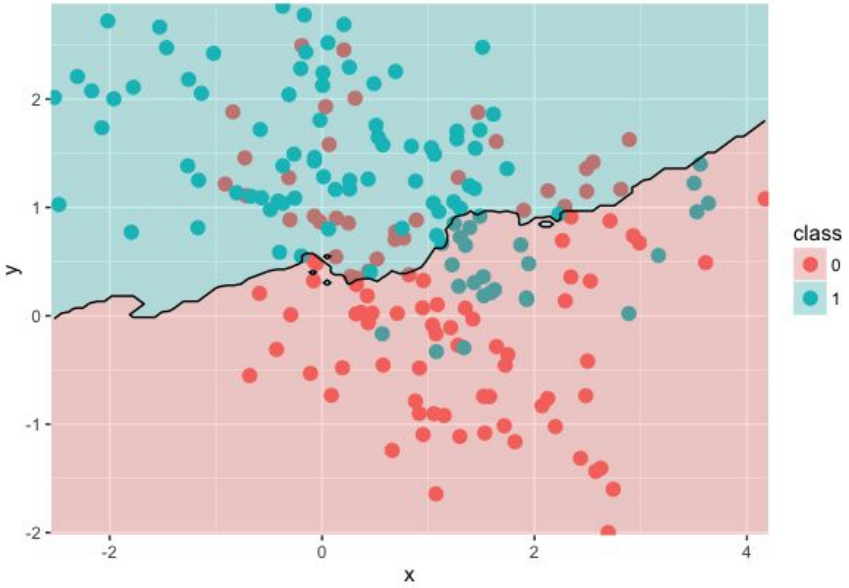
Small *k*

- $k = 1$
- Low bias
- Models like this are overfit
  - Read too much into the training data, extrapolating based on things that aren't necessarily relevant
- High variance: model varies greatly with the data



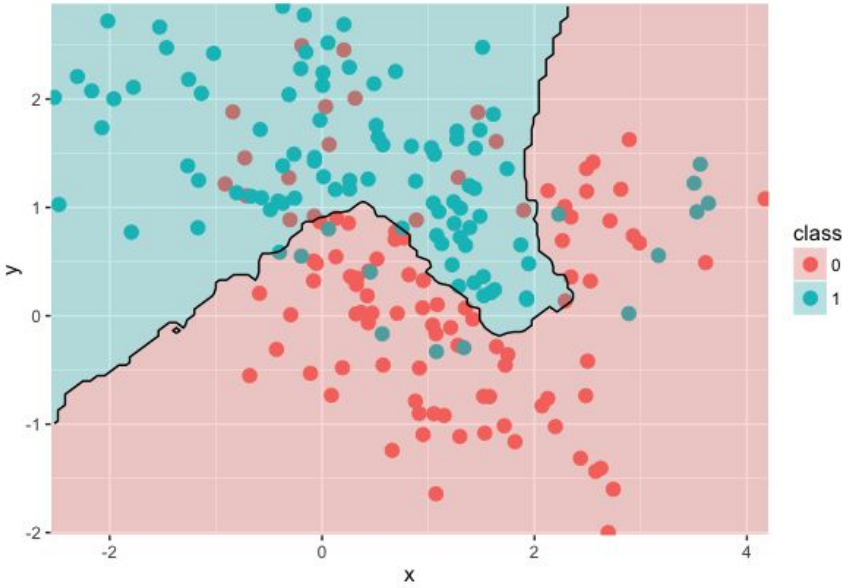
Large *k*

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



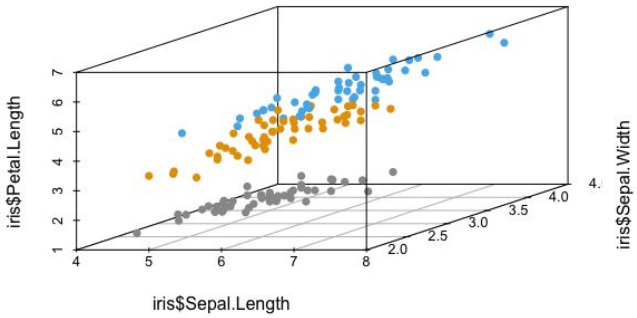
Model Selection

- Goal: minimize generalization error
  - *A.k.a.*, Out-of-sample error, or Risk
  - I.e., the error on new data
- Trade off bias vs. variance
- k = 15 “seems” just right
- This choice can/should be optimized using cross validation



3D and Beyond

- Our visualizations depicted only two features
- But *k*-NN is not limited to only two features
- *k*-NN can also use 3, 4, ..., *n* features
- knn uses all available numeric features



Key Design Decisions

- Choose *k*
  - Choose a threshold
- Define “neighbor”
  - Define a measure of distance/closeness
    (Make sure measurement values are comparable)
- Decide how to classify based on neighbors’ labels
  - By a majority vote, or
  - By a weighted majority vote (weighted by distance), or …

Model Selection

Find a model that appropriately balances complexity and generalization capabilities: i.e., that optimizes the bias-variance tradeoff.

- High bias, low variance
  - A high value of *k* indicates a high degree of bias, but contains the variance
- Low bias, high variance
  - With low values of *k*, the very jagged decision boundaries are a sign of high variance

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

*k*NN in R

`knn` in R

- R provides a `knn` function in the `class` library

```
> library(class) # Use classification package

> knn(train, test, cl, k)
```

- `train`: training data for the *k*-NN classifier
- `test`: testing data to classify
- `cl`: class labels
- `k`: the number of neighbors

knn in R (cont'd)

```
> knn(data.frame(iris$Petal.Length, iris$Sepal.Length),
      data.frame(new_point$Petal.Length,
                 new_point$Sepal.Length),
      iris$Species,
      k = 3)

[1] virginica
Levels: setosa versicolor virginica
```

knn in R (cont'd)

```
> knn(data.frame(iris$Petal.Length,
                 iris$Sepal.Length,
                 iris$Sepal.Width),
      data.frame(new_point$Petal.Length,
                 new_point$Sepal.Length,
                 new_point$Sepal.Width),
      iris$Species,
      k = 11)

[1] virginica
Levels: setosa versicolor virginica
```

knn in R (cont'd)

```
# Create training and test data
shuffled <- sample_n(iris, nrow(iris))
split <- 0.8 * nrow(shuffled)

training_data <- shuffled[1:split, 1:4]
test_data <- shuffled[(split + 1):nrow(shuffled), 1:4]

training_labels <- shuffled[1:split, 5]
correct_labels <- shuffled[(split + 1):nrow(shuffled), 5]

iris_pred <- knn(training_data,
                 test_data,
                 training_labels,
                 k = 5)
```

knn in R (cont'd)

```
> table(correct_labels, iris_pred)
```

```
                iris_pred
correct_labels setosa versicolor virginica
    setosa          10          0         0
    versicolor       0          9         1
    virginica        0          0        10
```