Classification

Prof Wells

STA 295: Stat Learning

April 2nd, 2024

Outline

- Introduce the Bayes Classifier
- Implement KNN as a method of approximating the Bayes classifier
- Discuss methods of assessing classification models

Section 1

Classification

The Task

Suppose Y is categorical response variable with several levels $A_1, \ldots, A_k$, and that $X_1, \ldots, X_p$ are predictors (either categorical or quantitative).

- Suppose we want to predict which of 4 presidential candidates that a voter will select, based on their age, race, gender and annual income.

The Task

Suppose Y is categorical response variable with several levels $A_1, \ldots, A_k$, and that $X_1, \ldots, X_p$ are predictors (either categorical or quantitative).

- Suppose we want to predict which of 4 presidential candidates that a voter will select, based on their age, race, gender and annual income.

Goal: Build a model $\hat{g}(X_1, \ldots, X_p)$, which outputs a level $A_1, \ldots, A_p$, that can be used to predict the class of Y based on $X_1, \ldots, X_p$.

The Task

Suppose Y is categorical response variable with several levels $A_1, \ldots, A_k$, and that $X_1, \ldots, X_p$ are predictors (either categorical or quantitative).

- Suppose we want to predict which of 4 presidential candidates that a voter will select, based on their age, race, gender and annual income.

Goal: Build a model $\hat{g}(X_1, \ldots, X_p)$, which outputs a level $A_1, \ldots, A_p$, that can be used to predict the class of Y based on $X_1, \ldots, X_p$.

- How do we measure accuracy of our model? RMSE is no longer appropriate (why?)

The Task

Suppose Y is categorical response variable with several levels $A_1, \ldots, A_k$, and that $X_1, \ldots, X_p$ are predictors (either categorical or quantitative).

- Suppose we want to predict which of 4 presidential candidates that a voter will select, based on their age, race, gender and annual income.

Goal: Build a model $\hat{g}(X_1, \ldots, X_p)$, which outputs a level $A_1, \ldots, A_p$, that can be used to predict the class of Y based on $X_1, \ldots, X_p$.

- How do we measure accuracy of our model? RMSE is no longer appropriate (why?)
- Compute error rate (proportion of incorrect predictions) on training data:

$$\text{Training Error} = \frac{1}{n}\sum_{i=1}^{n} I\big(y_i \neq \hat{g}(x_i)\big)$$

where $I\big(y_i \neq \hat{g}(x_i)\big)$ equals 1 if $y_i \neq \hat{g}(x_i)$ and 0 otherwise.

The Task

Suppose Y is categorical response variable with several levels $A_1, \ldots, A_k$, and that $X_1, \ldots, X_p$ are predictors (either categorical or quantitative).

- Suppose we want to predict which of 4 presidential candidates that a voter will select, based on their age, race, gender and annual income.

Goal: Build a model $\hat{g}(X_1, \ldots, X_p)$, which outputs a level $A_1, \ldots, A_p$, that can be used to predict the class of Y based on $X_1, \ldots, X_p$.

- How do we measure accuracy of our model? RMSE is no longer appropriate (why?)
- Compute error rate (proportion of incorrect predictions) on training data:

$$\text{Training Error} = \frac{1}{n} \sum_{i=1}^{n} I\big(y_i \neq \hat{g}(x_i)\big)$$

  where $I\big(y_i \neq \hat{g}(x_i)\big)$ equals 1 if $y_i \neq \hat{g}(x_i)$ and 0 otherwise.

- Compute average error rate on test data

$$\text{Test Error} = \text{Avg. } I\big(y_i \neq \hat{g}(x_0)\big)$$

  with the average taken across many test observations x_0.

The Best Possible Model

In general, the value of a response Y may depend on more than just the values of the predictors $X_1, \ldots, X_p$ in a model. That is, the value of the response y_0 is random.

The Best Possible Model

In general, the value of a response Y may depend on more than just the values of the predictors $X_1, \ldots, X_p$ in a model. That is, the value of the response y_0 is random.

- The **conditional probability** of Event 1 given Event 2 is written $P(\text{Event 1}|\text{Event 2})$. It is the probability that Event 1 occurs, given that we know Event 1 has occurred.

The Best Possible Model

In general, the value of a response Y may depend on more than just the values of the predictors $X_1, \ldots, X_p$ in a model. That is, the value of the response y_0 is random.

- The **conditional probability** of Event 1 given Event 2 is written $P(\text{Event 1}|\text{Event 2})$. It is the probability that Event 1 occurs, given that we know Event 1 has occurred.
  - For example, $P(Y = A_j|X = x_0)$ is the conditional probability that an observation has class A_j, given that it has predictor values x_0.

The Best Possible Model

In general, the value of a response Y may depend on more than just the values of the predictors $X_1, \ldots, X_p$ in a model. That is, the value of the response y_0 is random.

- The **conditional probability** of Event 1 given Event 2 is written $P(\text{Event 1}|\text{Event 2})$. It is the probability that Event 1 occurs, given that we know Event 1 has occurred.
  - For example, $P(Y = A_j | X = x_0)$ is the conditional probability that an observation has class A_j, given that it has predictor values x_0.
- A reasonable model for predicting Y would be the one that assigns each observation to the most likely class, given the values of its predictors.

The Best Possible Model

In general, the value of a response Y may depend on more than just the values of the predictors $X_1, \ldots, X_p$ in a model. That is, the value of the response y_0 is random.

- The **conditional probability** of $\text{Event } 1$ given $\text{Event } 2$ is written $P(\text{Event } 1|\text{Event } 2)$. It is the probability that $\text{Event } 1$ occurs, given that we know $\text{Event } 1$ has occurred.
  - For example, $P(Y = A_j|X = x_0)$ is the conditional probability that an observation has class A_j, given that it has predictor values x_0.
- A reasonable model for predicting Y would be the one that assigns each observation to the most likely class, given the values of its predictors.
  - Suppose Y is binary with two levels: A_1 and A_2. If we know that $P(Y = A_1|X = x_0) > 0.5$, then we should predict A_1. Otherwise, we predict A_2.

The Best Possible Model

In general, the value of a response Y may depend on more than just the values of the predictors $X_1, \ldots, X_p$ in a model. That is, the value of the response y_0 is random.

- The **conditional probability** of Event 1 given Event 2 is written $P(\text{Event 1}|\text{Event 2})$. It is the probability that Event 1 occurs, given that we know Event 1 has occurred.
  - For example, $P(Y = A_j|X = x_0)$ is the conditional probability that an observation has class A_j, given that it has predictor values x_0.
- A reasonable model for predicting Y would be the one that assigns each observation to the most likely class, given the values of its predictors.
  - Suppose Y is binary with two levels: A_1 and A_2. If we know that $P(Y = A_1|X = x_0) > 0.5$, then we should predict A_1. Otherwise, we predict A_2.
- This model is called the **Bayes Classifier** and can be written as:

$$g(x_0) = \text{argmax}_{A_j} P(Y = A_j \,|\, X = x_0)$$

The Best Possible Model

In general, the value of a response Y may depend on more than just the values of the predictors $X_1, \ldots, X_p$ in a model. That is, the value of the response y_0 is random.

- The **conditional probability** of Event 1 given Event 2 is written $P(\text{Event 1}|\text{Event 2})$. It is the probability that Event 1 occurs, given that we know Event 1 has occurred.
  - For example, $P(Y = A_j|X = x_0)$ is the conditional probability that an observation has class A_j, given that it has predictor values x_0.
- A reasonable model for predicting Y would be the one that assigns each observation to the most likely class, given the values of its predictors.
  - Suppose Y is binary with two levels: A_1 and A_2. If we know that $P(Y = A_1|X = x_0) > 0.5$, then we should predict A_1. Otherwise, we predict A_2.
- This model is called the **Bayes Classifier** and can be written as:

$$g(x_0) = \text{argmax}_{A_j} P(Y = A_j \,|\, X = x_0)$$

- This model actually minimizes average test error rate among all possible models.

The Best Possible Model

In general, the value of a response Y may depend on more than just the values of the predictors $X_1, \ldots, X_p$ in a model. That is, the value of the response y_0 is random.

- The **conditional probability** of Event 1 given Event 2 is written $P(\text{Event 1}|\text{Event 2})$. It is the probability that Event 1 occurs, given that we know Event 1 has occurred.
  - For example, $P(Y = A_j|X = x_0)$ is the conditional probability that an observation has class A_j, given that it has predictor values x_0.
- A reasonable model for predicting Y would be the one that assigns each observation to the most likely class, given the values of its predictors.
  - Suppose Y is binary with two levels: A_1 and A_2. If we know that $P(Y = A_1|X = x_0) > 0.5$, then we should predict A_1. Otherwise, we predict A_2.
- This model is called the **Bayes Classifier** and can be written as:

$$g(x_0) = \text{argmax}_{A_j} P(Y = A_j \,|\, X = x_0)$$

- This model actually minimizes average test error rate among all possible models.
- In practice, we cannot build this optimal model, since we don't know know the formula for $P(Y = A_j \,|\, X = x_0)$. Instead, we will try to estimate it.

Simulation

- Suppose Y takes values A or B, and X_1 and X_2 are predictors taking values in $[0, 1]$.

Simulation

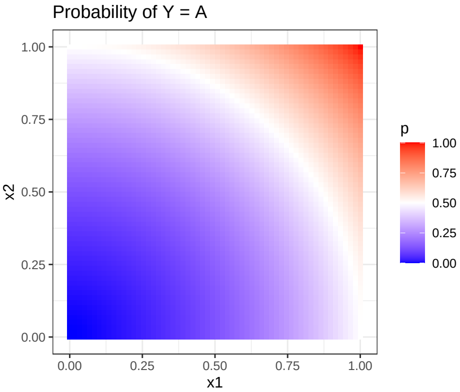
- Suppose Y takes values A or B, and X_1 and X_2 are predictors taking values in $[0, 1]$.
- Additionally, suppose that if $X_1 = x_1$ and $X_2 = x_2$, then $Y = A$ with probability

$$p = (x_1^2 + x_2^2)/2$$

Simulation

- Suppose Y takes values A or B, and X_1 and X_2 are predictors taking values in $[0, 1]$.
- Additionally, suppose that if $X_1 = x_1$ and $X_2 = x_2$, then $Y = A$ with probability

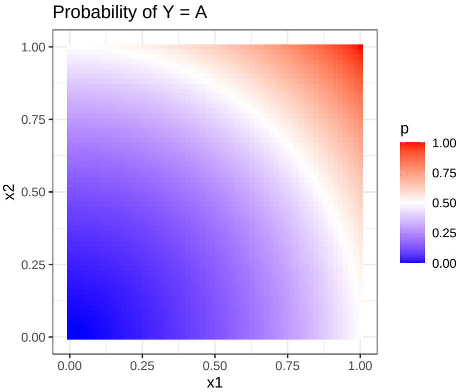
$$p = (x_1^2 + x_2^2)/2$$



Simulation

- Suppose Y takes values A or B, and X_1 and X_2 are predictors taking values in $[0, 1]$.
- Additionally, suppose that if $X_1 = x_1$ and $X_2 = x_2$, then $Y = A$ with probability

$$p = (x_1^2 + x_2^2)/2$$

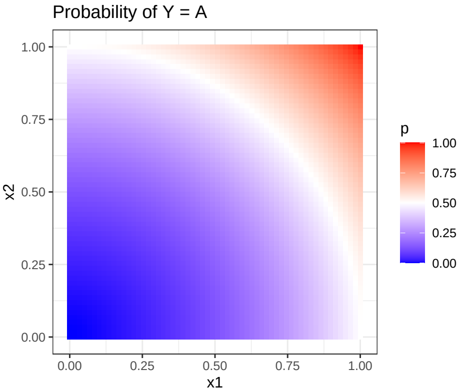


What is the Bayes Classifier g in this case?

Simulation

- Suppose Y takes values A or B, and X_1 and X_2 are predictors taking values in $[0, 1]$.
- Additionally, suppose that if $X_1 = x_1$ and $X_2 = x_2$, then $Y = A$ with probability

$$p = (x_1^2 + x_2^2)/2$$



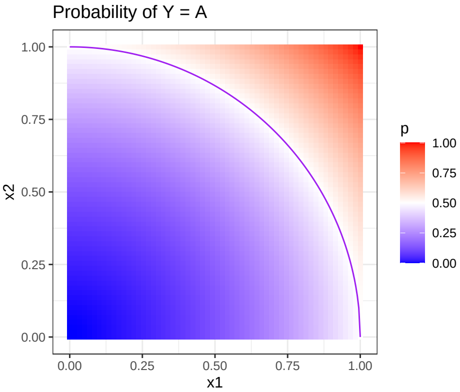
What is the Bayes Classifier g in this case?

$$g(x_0) = \text{argmax}_{A_j} P(Y = A_j \mid X = x_0)$$
$$= \begin{cases} A, & \text{if } x_1^2 + x_2^2 \geq 1 \\ B, & \text{if } x_1^2 + x_2^2 < 1 \end{cases}$$

Simulation

- Suppose Y takes values A or B, and X_1 and X_2 are predictors taking values in $[0, 1]$.
- Additionally, suppose that if $X_1 = x_1$ and $X_2 = x_2$, then $Y = A$ with probability

$$p = (x_1^2 + x_2^2)/2$$



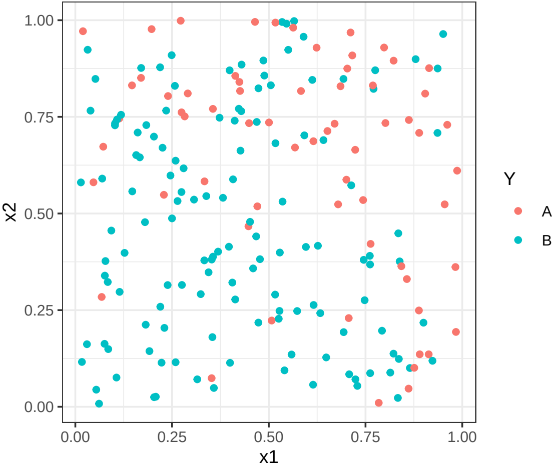
What is the Bayes Classifier g in this case?

$$g(x_0) = \mathrm{argmax}_{A_j} P(Y = A_j \mid X = x_0)$$

$$= \begin{cases} A, & \text{if } x_1^2 + x_2^2 \geq 1 \\ B, & \text{if } x_1^2 + x_2^2 < 1 \end{cases}$$

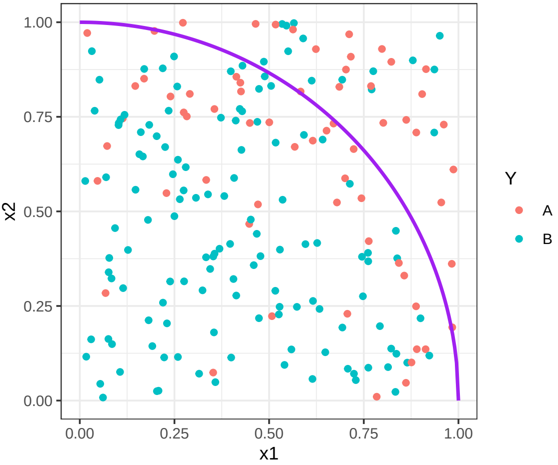
Simulate Data

Let's simulate 200 data points from this model.



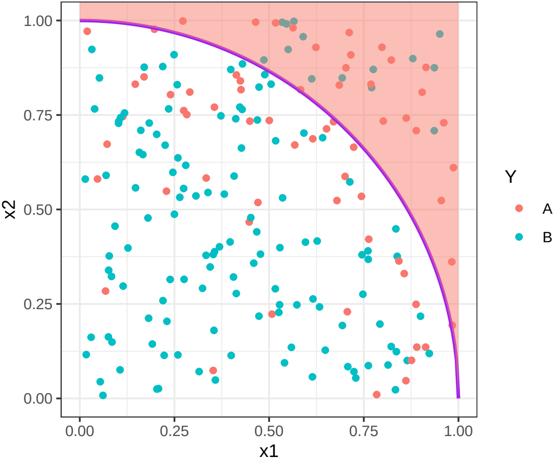
The Bayes Classifier

The purple arc represents the Bayes Classifier boundary



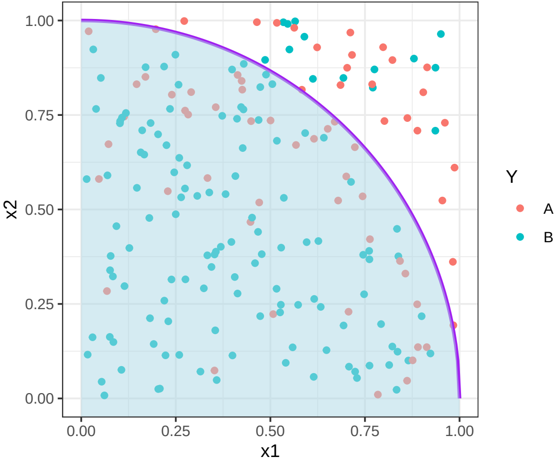
The Bayes Classifier

Any test point outside the circle should be classified as A (red)



The Bayes Classifier

Any test point inside the circle should be classified as B (blue)



Expected Error Rate

In general, using the Bayes Classifier produces an expected error rate of

$$1 - \mathrm{Avg.}\left(\max_j \mathrm{P}(Y = A_j \mid X = x_0)\right)$$

Expected Error Rate

In general, using the Bayes Classifier produces an expected error rate of

$$1 - \text{Avg.}\left(\max_j \mathrm{P}(Y = A_j \mid X = x_0)\right)$$

- For our study, our Bayes Classifier has a theoretical error rate of $\frac{2}{3} - \frac{\pi}{8} \approx 0.274$.

Expected Error Rate

In general, using the Bayes Classifier produces an expected error rate of

$$1 - \mathrm{Avg.}\left(\max_j \mathrm{P}(Y = A_j \mid X = x_0)\right)$$

- For our study, our Bayes Classifier has a theoretical error rate of $\frac{2}{3} - \frac{\pi}{8} \approx 0.274$.
  - Can verify using multivariate calculus or by sampling a large number of times.

Expected Error Rate

In general, using the Bayes Classifier produces an expected error rate of

$$1 - \text{Avg.}\left(\max_j \mathrm{P}(Y = A_j \mid X = x_0)\right)$$

- For our study, our Bayes Classifier has a theoretical error rate of $\frac{2}{3} - \frac{\pi}{8} \approx 0.274$.
  - Can verify using multivariate calculus or by sampling a large number of times.
- Why won't the Bayes Classifier give an error rate of 0?

Expected Error Rate

In general, using the Bayes Classifier produces an expected error rate of

$$1 - \text{Avg.}\left(\max_j \mathrm{P}(Y = A_j \mid X = x_0)\right)$$

- For our study, our Bayes Classifier has a theoretical error rate of $\frac{2}{3} - \frac{\pi}{8} \approx 0.274$.
  - Can verify using multivariate calculus or by sampling a large number of times.
- Why won't the Bayes Classifier give an error rate of 0?
  - The response Y is random and may depend on variables beyond the predictors used in the model. Even if an observation has high probability of being in a class, it isn't guaranteed to be in that class.

Expected Error Rate

In general, using the Bayes Classifier produces an expected error rate of

$$1 - \text{Avg.}\left(\max_j \mathrm{P}(Y = A_j \,|\, X = x_0)\right)$$

- For our study, our Bayes Classifier has a theoretical error rate of $\frac{2}{3} - \frac{\pi}{8} \approx 0.274$.
    - Can verify using multivariate calculus or by sampling a large number of times.
- Why won't the Bayes Classifier give an error rate of 0?
    - The response Y is random and may depend on variables beyond the predictors used in the model. Even if an observation has high probability of being in a class, it isn't guaranteed to be in that class.
- This is the theoretical lower bound on average test error for this classification problem.

Expected Error Rate

In general, using the Bayes Classifier produces an expected error rate of

$$1 - \text{Avg.}\left(\max_j \mathrm{P}(Y = A_j \mid X = x_0)\right)$$

- For our study, our Bayes Classifier has a theoretical error rate of $\frac{2}{3} - \frac{\pi}{8} \approx 0.274$.
  - Can verify using multivariate calculus or by sampling a large number of times.
- Why won't the Bayes Classifier give an error rate of 0?
  - The response Y is random and may depend on variables beyond the predictors used in the model. Even if an observation has high probability of being in a class, it isn't guaranteed to be in that class.
- This is the theoretical lower bound on average test error for this classification problem.
  - This is analogous to the irreducible error in regression problems

Section 2

K-Nearest Neighbors

From Bayes Classifier to KNN

In theory, the Bayes Classifier is our best model for classification.

From Bayes Classifier to KNN

In theory, the Bayes Classifier is our best model for classification.

- In practice, we don't know the conditional probability of Y given X, and so cannot build a Bayes Classifier model.

From Bayes Classifier to KNN

In theory, the Bayes Classifier is our best model for classification.

- In practice, we don't know the conditional probability of Y given X, and so cannot build a Bayes Classifier model.
- But given sufficient data, we can *estimate* the conditional probabilities

From Bayes Classifier to KNN

In theory, the Bayes Classifier is our best model for classification.

- In practice, we don't know the conditional probability of Y given X, and so cannot build a Bayes Classifier model.
- But given sufficient data, we can *estimate* the conditional probabilities

Given a positive integer K and a test observation x_0, let N_0 denote the K nearest training observations to x_0. Then our model for the conditional probability $P(Y = A_j|X = x_0)$ is

$$P(Y = A_j \,|\, X = x_0) \approx \frac{1}{K} \sum_{i \in N_0} I(y_i = A_j)$$

From Bayes Classifier to KNN

In theory, the Bayes Classifier is our best model for classification.

- In practice, we don't know the conditional probability of Y given X, and so cannot build a Bayes Classifier model.
- But given sufficient data, we can *estimate* the conditional probabilities

Given a positive integer K and a test observation x_0, let N_0 denote the K nearest training observations to x_0. Then our model for the conditional probability $P(Y = A_j | X = x_0)$ is

$$P(Y = A_j \,|\, X = x_0) \approx \frac{1}{K} \sum_{i \in N_0} I(y_i = A_j)$$

- Our classifier model is

$$\hat{g}(x_0) = \text{argmax}_{A_j} \left\{ \frac{1}{K} \sum_{i \in N_0} I(y_i = A_j) \right\}$$

From Bayes Classifier to KNN

In theory, the Bayes Classifier is our best model for classification.

- In practice, we don't know the conditional probability of Y given X, and so cannot build a Bayes Classifier model.
- But given sufficient data, we can *estimate* the conditional probabilities

Given a positive integer K and a test observation x_0, let N_0 denote the K nearest training observations to x_0. Then our model for the conditional probability $P(Y = A_j|X = x_0)$ is

$$P(Y = A_j \,|\, X = x_0) \approx \frac{1}{K} \sum_{i \in N_0} I(y_i = A_j)$$

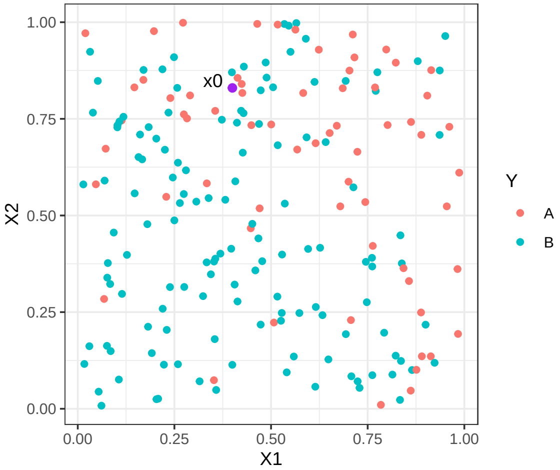
- Our classifier model is

$$\hat{g}(x_0) = \mathrm{argmax}_{A_j} \left\{ \frac{1}{K} \sum_{i \in N_0} I(y_i = A_j) \right\}$$

- An alternative formulation is that $\hat{g}(x_0)$ predicts the class with greatest frequency among the K neighbors of x_0.

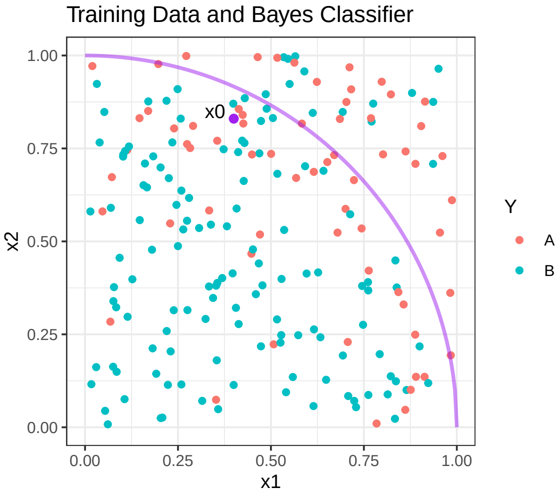
Classify Points

Classify x_0 for $K = 1, 2, 3, 5, 10, 200$.



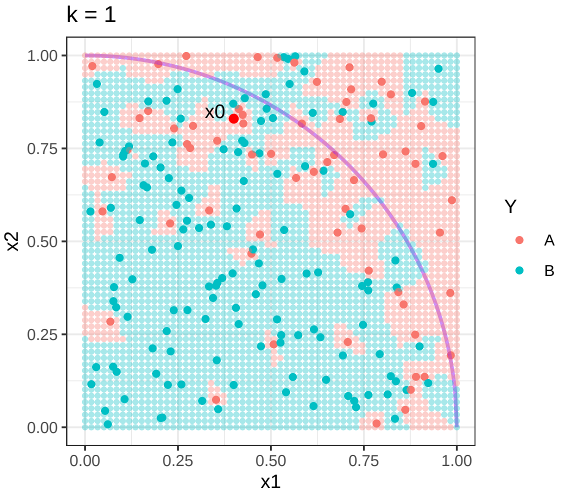
Classification Boundaries

Here are the classification boundaries for a variety of values of K.



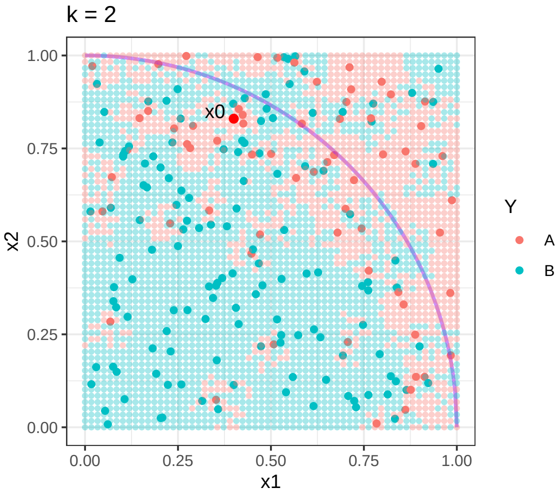
k=1

Here are the classification boundaries for a variety of values of K.



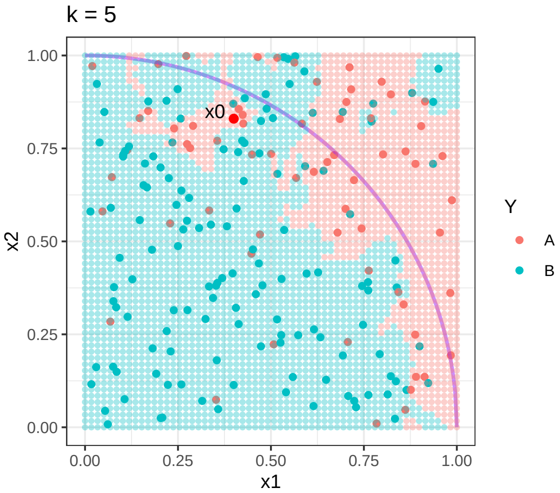
k=2

Here are the classification boundaries for a variety of values of K.



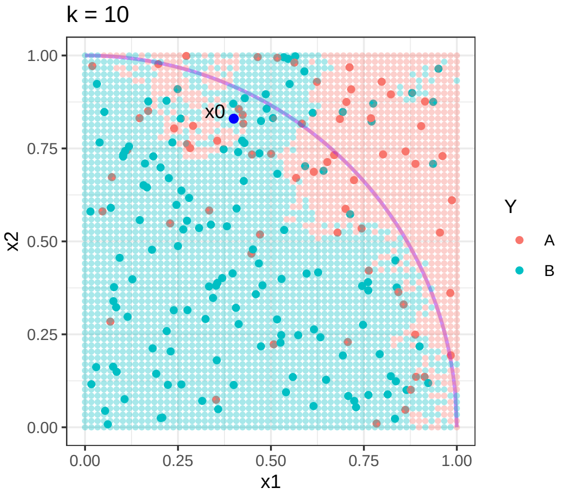
k=5

Here are the classification boundaries for a variety of values of K.



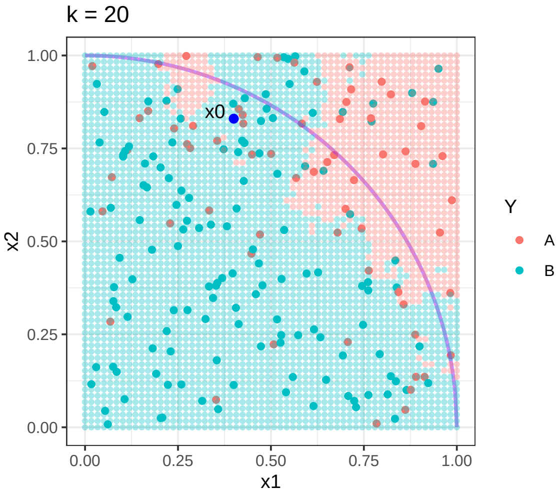
k=10

Here are the classification boundaries for a variety of values of K.



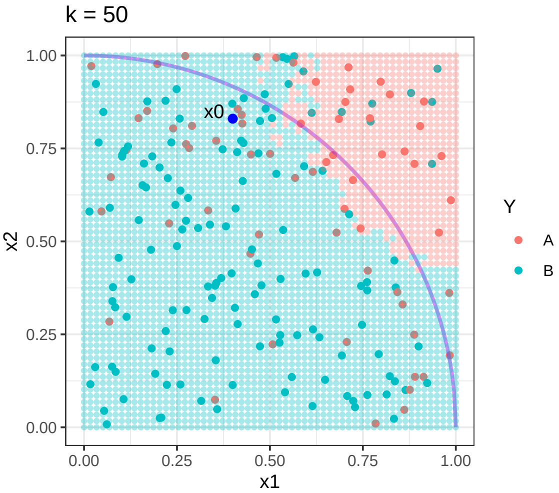
k=20

Here are the classification boundaries for a variety of values of K.



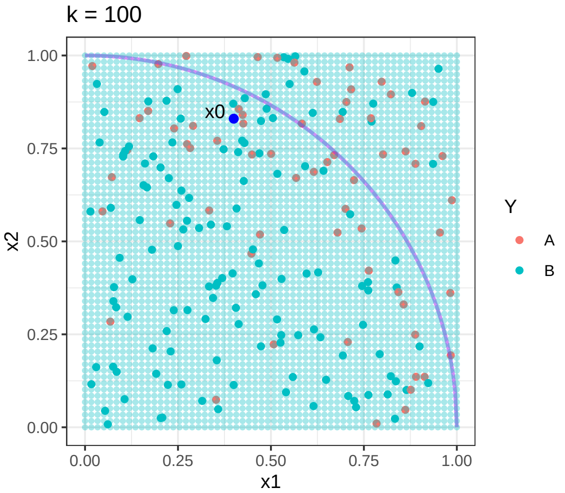
k=50

Here are the classification boundaries for a variety of values of K.



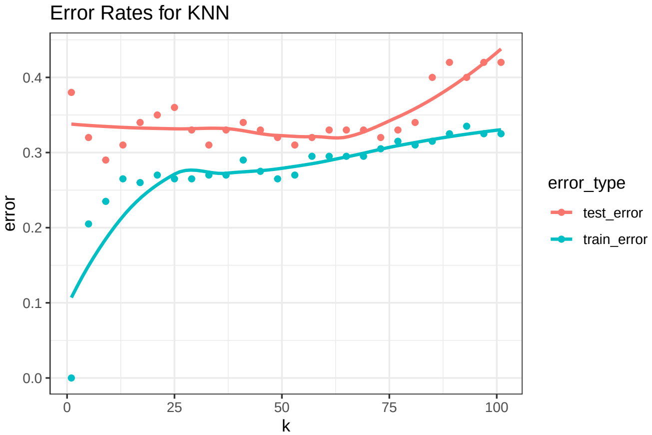
k=100

Here are the classification boundaries for a variety of values of K.



Error Rates

The graph below shows error rates for the training set, as well as a test set of 100 points.



Section 3

Assessing Classification Models

KNN Classification in R

- To create `knn` models in R, we use the `kknn` function from the `kknn` package.
  - Previously, we used `kknn` to build regression models, but classification models are possible as well

KNN Classification in R

- To create `knn` models in R, we use the `kknn` function from the `kknn` package.
  - Previously, we used `kknn` to build regression models, but classification models are possible as well
- The output of `kknn` is a list with several components:

KNN Classification in R

- To create `knn` models in R, we use the `kknn` function from the `kknn` package.
  - Previously, we used `kknn` to build regression models, but classification models are possible as well
- The output of `kknn` is a list with several components:
  - `fitted.values`, a vector of predicted *classes*
  - `prob`, a matrix of predicted class probabilities
  - `CL`, a matrix of the classes of the `k` nearest neighbors
  - `D`, a matrix of the distances from each point to the `k` nearest neighbors

Code for KNN

As an example, we fit KNN with $k = 30$

Code for KNN

As an example, we fit KNN with $k = 30$

- Let's inspect the structure of the training and testing data:

```
glimpse(train_data)
```

```
## Rows: 200
## Columns: 3
## $ x1 <dbl> 0.50747820, 0.30676851, 0.42690767, 0.69310208, 0.08513597, 0.22543~
## $ x2 <dbl> 0.2230884, 0.5358950, 0.6625291, 0.8480705, 0.1491831, 0.6700994, 0~
## $ Y  <fct> A, B, B, B, B, B, A, A, B, B, A, A, A, B, B, B, B, B, B, B, B, A, B~
```

```
glimpse(test_data)
```

```
## Rows: 100
## Columns: 3
## $ x1 <dbl> 0.89760792, 0.71213586, 0.32742617, 0.76785585, 0.68311176, 0.37160~
## $ x2 <dbl> 0.72979928, 0.60380908, 0.87182280, 0.34015532, 0.63769834, 0.33938~
## $ Y  <fct> A, A, B, B, B, B, B, B, A, B, B, B, B, B, B, A, B, B, A, A, B, B, A~
```

Code for KNN

As an example, we fit KNN with $k = 30$

- Let's inspect the structure of the training and testing data:

```
glimpse(train_data)
```

```
## Rows: 200
## Columns: 3
## $ x1 <dbl> 0.50747820, 0.30676851, 0.42690767, 0.69310208, 0.08513597, 0.22543~
## $ x2 <dbl> 0.2230884, 0.5358950, 0.6625291, 0.8480705, 0.1491831, 0.6700994, 0~
## $ Y  <fct> A, B, B, B, B, B, A, A, B, B, A, A, A, B, B, B, B, B, B, B, B, A, B~
```

```
glimpse(test_data)
```

```
## Rows: 100
## Columns: 3
## $ x1 <dbl> 0.89760792, 0.71213586, 0.32742617, 0.76785585, 0.68311176, 0.37160~
## $ x2 <dbl> 0.72979928, 0.60380908, 0.87182280, 0.34015532, 0.63769834, 0.33938~
## $ Y  <fct> A, A, B, B, B, B, B, B, A, B, B, B, B, B, B, A, B, B, A, A, B, B, A~
```

- Now we build the knn object:

```
library(kknn)
sim_fit_30 <- kknn(Y ~ x1 + x2, train = train_data, test = test_data,
                   k = 30, kernel = "rectangular")
```

Code for KNN

Let's look at the `fitted.values`

```
head(sim_fit_30$fitted.values)
```

```
## [1] A A B B A B
## Levels: A B
```

Code for KNN

Let's look at the `fitted.values`

```
head(sim_fit_30$fitted.values)
```

```
## [1] A A B B A B
## Levels: A B
```

- And the matrix of class probabilities

```
head(sim_fit_30$prob)
```

```
##                A         B
## [1,] 0.6666667 0.3333333
## [2,] 0.5333333 0.4666667
## [3,] 0.4666667 0.5333333
## [4,] 0.3333333 0.6666667
## [5,] 0.5666667 0.4333333
## [6,] 0.1000000 0.9000000
```

Code for KNN

Let's look at the `fitted.values`

```
head(sim_fit_30$fitted.values)
```

```
## [1] A A B B A B
## Levels: A B
```

- And the matrix of class probabilities

```
head(sim_fit_30$prob)
```

```
##              A         B
## [1,] 0.6666667 0.3333333
## [2,] 0.5333333 0.4666667
## [3,] 0.4666667 0.5333333
## [4,] 0.3333333 0.6666667
## [5,] 0.5666667 0.4333333
## [6,] 0.1000000 0.9000000
```

- Create a new data frame containing the true response values, the predicted response values, and the class probabilities

```
sim_results <- data.frame(obs = test_data$Y,
                          preds = sim_fit_30$fitted.values,
                          probs = sim_fit_30$prob)
```

Confusion Matrix

The `yardstick` package contain several functions for measuring model performance.

Confusion Matrix

The `yardstick` package contain several functions for measuring model performance.

- A confusion matrix is a two-way table comparing the model predictions to true response values

Confusion Matrix

The `yardstick` package contain several functions for measuring model performance.

- A confusion matrix is a two-way table comparing the model predictions to true response values
    - We create the confusion matrix using `conf_mat` from `yardstick`

Confusion Matrix

The `yardstick` package contain several functions for measuring model performance.

- A confusion matrix is a two-way table comparing the model predictions to true response values
    - We create the confusion matrix using `conf_mat` from `yardstick`

```
library(yardstick)
conf_mat(sim_results, truth = obs, estimate = preds)
```

```
##           Truth
## Prediction  A  B
##          A 14  7
##          B 28 51
```

Confusion Matrix

The `yardstick` package contain several functions for measuring model performance.

- A confusion matrix is a two-way table comparing the model predictions to true response values
  - We create the confusion matrix using `conf_mat` from `yardstick`

```
library(yardstick)
conf_mat(sim_results, truth = obs, estimate = preds)
```

```
##           Truth
## Prediction  A  B
##          A 14  7
##          B 28 51
```

- Correct predictions are represented along the diagonal of the confusion matrix

Confusion Matrix

The `yardstick` package contain several functions for measuring model performance.

- A confusion matrix is a two-way table comparing the model predictions to true response values
  - We create the confusion matrix using `conf_mat` from `yardstick`

```
library(yardstick)
conf_mat(sim_results, truth = obs, estimate = preds)
```

```
##           Truth
## Prediction  A  B
##          A 14  7
##          B 28 51
```

- Correct predictions are represented along the diagonal of the confusion matrix
- A model's **accuracy** is its proportion of correct guesses.

Confusion Matrix

The `yardstick` package contain several functions for measuring model performance.

- A confusion matrix is a two-way table comparing the model predictions to true response values
  - We create the confusion matrix using `conf_mat` from `yardstick`

```
library(yardstick)
conf_mat(sim_results, truth = obs, estimate = preds)
```

```
##           Truth
## Prediction  A  B
##          A 14  7
##          B 28 51
```

- Correct predictions are represented along the diagonal of the confusion matrix
- A model's **accuracy** is its proportion of correct guesses.
  - There were $14 + 51 = 65$ correct guesses, out of 100 attempts, for an accuracy of 0.65.

Error Rate

- Instead of computing by hand, we can use `yardstick`'s `accuracy` function.

Error Rate

- Instead of computing by hand, we can use `yardstick`'s `accuracy` function.

```
accuracy(sim_results, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.65
```

Error Rate

- Instead of computing by hand, we can use `yardstick`'s `accuracy` function.

```
accuracy(sim_results, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.65
```

- The **Error Rate** of a model is the proportion of incorrect guesses. Note that every guess is either correct or incorrect, so

$$\text{error} + \text{accuracy} = 1 \quad \Longrightarrow \quad \text{error} = 1 - \text{accuracy}$$

Error Rate

- Instead of computing by hand, we can use yardstick's accuracy function.

```
accuracy(sim_results, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.65
```

- The **Error Rate** of a model is the proportion of incorrect guesses. Note that every guess is either correct or incorrect, so

$$\text{error} + \text{accuracy} = 1 \quad \Longrightarrow \quad \text{error} = 1 - \text{accuracy}$$

- There is no yardstick function for `error`, but we can compute it by pulling the estimate value from the `accuracy` data frame, and subtracting from 1:

Error Rate

- Instead of computing by hand, we can use `yardstick`'s `accuracy` function.

```
accuracy(sim_results, truth = obs, estimate = preds)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.65
```

- The **Error Rate** of a model is the proportion of incorrect guesses. Note that every guess is either correct or incorrect, so

$$\text{error} + \text{accuracy} = 1 \quad \Longrightarrow \quad \text{error} = 1 - \text{accuracy}$$

- There is no yardstick function for `error`, but we can compute it by pulling the `estimate` value from the `accuracy` data frame, and subtracting from 1:

```
acc <- accuracy(sim_results, truth = obs, estimate = preds) %>% pull(.estimate)
error <- 1 - acc
error
```

```
## [1] 0.35
```

Sensitivity and Specificity

Sensitivity: Proportion of true positives correctly predicted

- Type II Error rate: $1 - \text{Sensitivity}$ (false negatives)

Specificity: Proportion of true negatives correctly predicted

- Type I Error rate: $1 - \text{Specificity}$ (false positives)

Sensitivity and Specificity

Sensitivity: Proportion of true positives correctly predicted

- Type II Error rate: $1 - \text{Sensitivity}$ (false negatives)

Specificity: Proportion of true negatives correctly predicted

- Type I Error rate: $1 - \text{Specificity}$ (false positives)

Usually, we predict class A if $P(Y = A|X = x_0) > 0.5$. But we could change our cut-off from 0.5 to another proportion.

Sensitivity and Specificity

Sensitivity: Proportion of true positives correctly predicted

- Type II Error rate: $1 - \text{Sensitivity}$ (false negatives)

Specificity: Proportion of true negatives correctly predicted

- Type I Error rate: $1 - \text{Specificity}$ (false positives)

Usually, we predict class A if $P(Y = A|X = x_0) > 0.5$. But we could change our cut-off from 0.5 to another proportion.

- For example, we could instead use predict A if $P(Y = A|X = x_0) > 0.1$

Sensitivity and Specificity

Sensitivity: Proportion of true positives correctly predicted

- Type II Error rate: $1 - \text{Sensitivity}$ (false negatives)

Specificity: Proportion of true negatives correctly predicted

- Type I Error rate: $1 - \text{Specificity}$ (false positives)

Usually, we predict class A if $P(Y = A|X = x_0) > 0.5$. But we could change our cut-off from 0.5 to another proportion.

- For example, we could instead use predict A if $P(Y = A|X = x_0) > 0.1$
- By doing so, we can increase sensitivity to the detriment of specificity (or vice versa). But the tradeoff is non-linear

Sensitivity and Specificity

Sensitivity: Proportion of true positives correctly predicted

- Type II Error rate: $1 - \text{Sensitivity}$ (false negatives)

Specificity: Proportion of true negatives correctly predicted

- Type I Error rate: $1 - \text{Specificity}$ (false positives)

Usually, we predict class A if $P(Y = A|X = x_0) > 0.5$. But we could change our cut-off from 0.5 to another proportion.

- For example, we could instead use predict A if $P(Y = A|X = x_0) > 0.1$
- By doing so, we can increase sensitivity to the detriment of specificity (or vice versa). But the tradeoff is non-linear
  - Decreasing specificity by 0.1 may increase sensitivity by 0.15 **when specificity is** 0.6, but may only increase sensitivity by 0.05 **when specificity is** 0.3.

Sensitivity and Specificity

Sensitivity: Proportion of true positives correctly predicted

- Type II Error rate: $1 - \text{Sensitivity}$ (false negatives)

Specificity: Proportion of true negatives correctly predicted

- Type I Error rate: $1 - \text{Specificity}$ (false positives)

Usually, we predict class A if $P(Y = A|X = x_0) > 0.5$. But we could change our cut-off from 0.5 to another proportion.

- For example, we could instead use predict A if $P(Y = A|X = x_0) > 0.1$
- By doing so, we can increase sensitivity to the detriment of specificity (or vice versa). But the tradeoff is non-linear
  - Decreasing specificity by 0.1 may increase sensitivity by 0.15 **when specificity is** 0.6, but may only increase sensitivity by 0.05 **when specificity is** 0.3.
- When might we want high specificity? High sensitivity?

Sensitivity and Specificity

Sensitivity: Proportion of true positives correctly predicted

- Type II Error rate: $1 - \text{Sensitivity}$ (false negatives)

Specificity: Proportion of true negatives correctly predicted

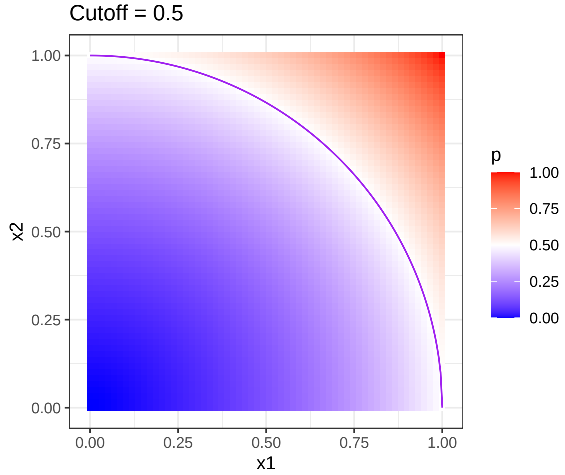
- Type I Error rate: $1 - \text{Specificity}$ (false positives)

Usually, we predict class A if $P(Y = A|X = x_0) > 0.5$. But we could change our cut-off from 0.5 to another proportion.

- For example, we could instead use predict A if $P(Y = A|X = x_0) > 0.1$
- By doing so, we can increase sensitivity to the detriment of specificity (or vice versa). But the tradeoff is non-linear
  - Decreasing specificity by 0.1 may increase sensitivity by 0.15 **when specificity is** 0.6, but may only increase sensitivity by 0.05 **when specificity is** 0.3**.**
- When might we want high specificity? High sensitivity?
- What are the ramifications of changing the classification cutoff?

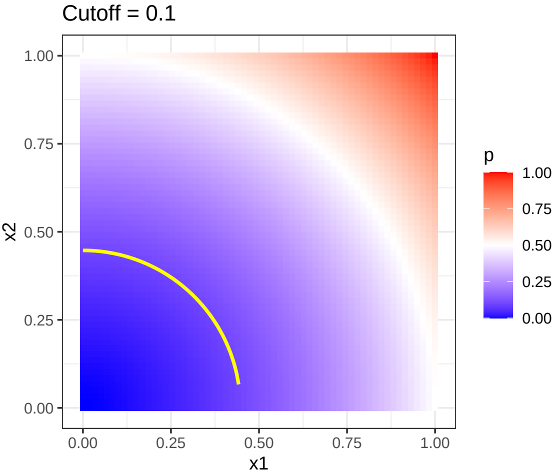
Decision Boundary

Changing the cutoff corresponds to changing our decision boundary:



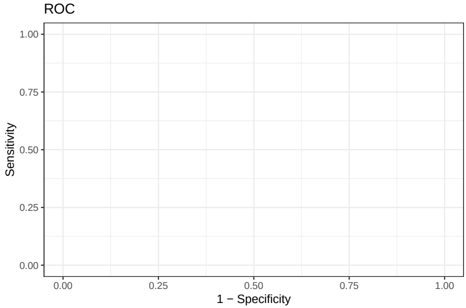
Decision Boundary

Changing the cutoff corresponds to changing our decision boundary:



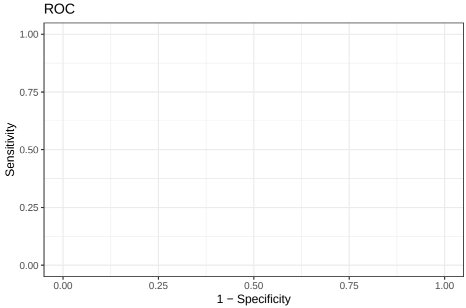
ROC Curves

A Receiver Operating Characteristic (ROC) curve is a plot of sensitivity vs. type I error rate, based on classification probabilities.



ROC Curves

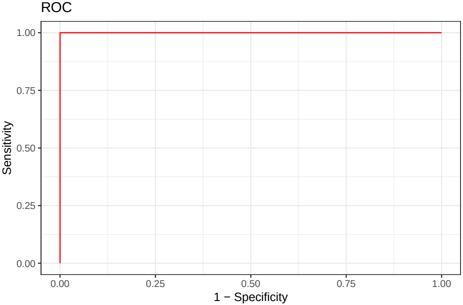
A Receiver Operating Characteristic (ROC) curve is a plot of sensitivity vs. type I error rate, based on classification probabilities.



- What does the ROC curve look like for a perfectly accurate model?

ROC Curves

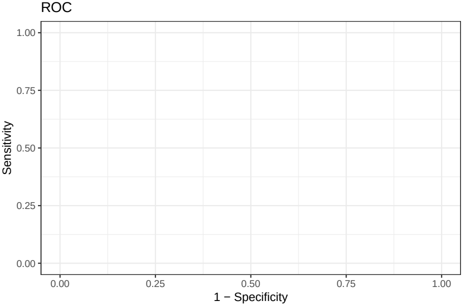
A Receiver Operating Characteristic (ROC) curve is a plot of sensitivity vs. type I error rate, based on classification probabilities.



- What does the ROC curve look like for a perfectly accurate model?

ROC Curves

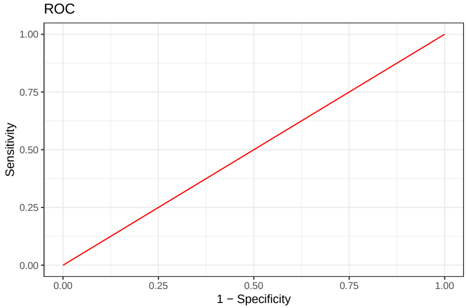
A Receiver Operating Characteristic (ROC) curve is a plot of sensitivity vs. type I error rate, based on classification probabilities.



- What does the ROC curve look like for a model that just randomly guesses?

ROC Curves

A Receiver Operating Characteristic (ROC) curve is a plot of sensitivity vs. type I error rate, based on classification probabilities.



- What does the ROC curve look like for a model that just randomly guesses?

AUC

- The area under the ROC curve (AUC) is a method for assessing how well a model performs:
    - The perfect model has AUC of 1, while the random guessing model has AUC of 0.5.

AUC

- The area under the ROC curve (AUC) is a method for assessing how well a model performs:
    - The perfect model has AUC of 1, while the random guessing model has AUC of 0.5.
- Models with higher AUC (closer to 1) are better than models with lower AUC (closer to 0.5)

AUC

- The area under the ROC curve (AUC) is a method for assessing how well a model performs:
    - The perfect model has AUC of 1, while the random guessing model has AUC of 0.5.
- Models with higher AUC (closer to 1) are better than models with lower AUC (closer to 0.5)
- The `roc_auc` function in `yardstick` will compute the area under the ROC curve

AUC

- The area under the ROC curve (AUC) is a method for assessing how well a model performs:
  - The perfect model has AUC of 1, while the random guessing model has AUC of 0.5.
- Models with higher AUC (closer to 1) are better than models with lower AUC (closer to 0.5)
- The `roc_auc` function in `yardstick` will compute the area under the ROC curve

```
roc_auc(sim_results, truth = obs, probs.A)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.750
```

AUC

- The area under the ROC curve (AUC) is a method for assessing how well a model performs:
  - The perfect model has AUC of 1, while the random guessing model has AUC of 0.5.
- Models with higher AUC (closer to 1) are better than models with lower AUC (closer to 0.5)
- The `roc_auc` function in `yardstick` will compute the area under the ROC curve

```
roc_auc(sim_results, truth = obs, probs.A)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc binary         0.750
```

- Here, `probs.A` is the column name for the column containing the estimated probabilities each observation is of class A.

Creating ROC Curves

The `roc_curve` function in `yardstick` will create an ROC curve:

Creating ROC Curves

The `roc_curve` function in `yardstick` will create an ROC curve: