Lab0 Team ROCstars

Ryan Morreale, Emery Berry, Jacob Krol

Lab0: About Your Team, About Individuals, and Team Collaboration

Due: 11:59PM Sunday, January 25 on Canvas as a knitted pdf file of a Quarto document

1. You will determine your team's name and goals.
2. You will add to the document your individual sections.
3. You will practice collaborating on an applied statistical learning "project."

Instructions for Lab0

1. Using Quarto, you will complete the first team section "About Team Teamname." See further instructions below.
2. Each team member will add their own individual section to the team Quarto document. How do you want to collaborate on your document this semester? Github? Google Colab? Something else? See instructions for individual sections below.
3. As a team, you will complete an applied "project". There will be some individual components (so that everyone gets practice with implementing the stat learning methods) and team components.

About Your Team

In this team section, include a team photo (all of you), your team name, and your team's main goal for this semester for this course (i.e., what does your team want to accomplish by the end of the semester?). Throughout the semester you will be giving feedback to your teammates about how well they are helping your team accomplish its goal(s).

Individual Sections

Each individual must complete their own subsection, which must include:

- a photo of yourself with a caption explaining the context
- at least one (non-statistics) question you would like to know the answer to that could potentially be answered by applying statistical learning methods to data
- what you would love to be doing six months after graduation and then five years after that (what would make you excited to be doing?)
- what you hope your greatest career accomplishment will be
- and given these hopes and goals, what you are hoping to learn/accomplish/do in this course.
- You must also include something of your own choosing not described above. Anything. Be creative!

Team Applied Section

Find the misclassification rate for K-nearest neighbors (KNN) for a range of k values, given a complicated true generating model. Do this individually. As a team, compare answers and then plot the decision boundary for the team's "best" value of k.

Also, individually practice walking through the steps of Q1, Q2, and Q3 for this "project". Within this section will be a team section (what is the "best" k and what is the decision boundary) and individual sections. In your individual section, describe a plausible story for Q1, Q2, and Q3. Specifically, for Q1: What is Y, X1, and X2, and why should we care?. Make up something plausible that you actually care about. Y could be whether a kitten is adopted from a shelter given X1=age and X2=health of the cat. That's just one of countless plausible examples.

For Q2: Make predictions given X1 and X2 using KNN and k=(1 and 5) or (2 and 6) or (3 and 7) or (4 and 8). That is, each teammate fits KNN for two different values of k. Compute the misclassification rates for your values of k.

For Q3: Using this model and your plausible Q1 scenario, what is the answer for X1=0.5 and X2=0.5? For your scenario, what are some ethical implications of your stat learning modeling?

Generating Model

We have a logistic regression generating model. Given $x_1 \in [0,1]$ and $x_2 \in [0,1], Y \sim Ber(p)$, where p is related to x_1 and x_2 through the logistic link function: $\log(\frac{p}{(1-p)}) = x_1 - 2x_2 - 2{x_1}^2 + {x_2}^2 + 3x_1x_2$, where log is the natural log, sometimes written as ln.

The code for this is below.

```
library(class)
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.6
v forcats   1.0.1     v stringr   1.6.0
v ggplot2   4.0.1     v tibble    3.3.1
v lubridate 1.9.4     v tidyr     1.3.2
v purrr     1.2.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to becor
```

```
#Generative model
set.seed(116) #setting a random seed so that we can reproduce everything exactly if we want t

generate_y <- function(x1,x2) { #two input parameters to generate the output y
  logit <- x1 -2*x2 -2*x1^2 + x2^2 + 3*x1*x2
  p <- exp(logit)/(1+exp(logit)) #apply the inverse logit function
  y <- rbinom(1,1,p) #y becomes a 0 (with prob 1-p) or a 1 with probability p
}
```

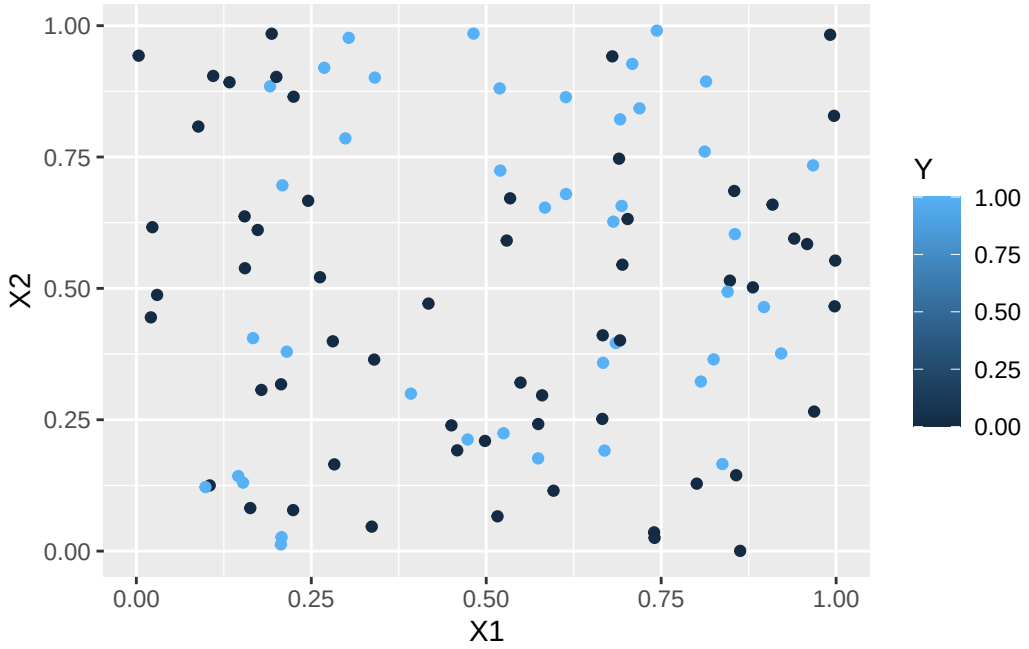
Example code

We are going to use our generating model to create a test set of 100 predictors (x1, x2), and then 100 outcomes. Then we plot all three variables just to see what the generating model is doing.

```
# Generate a dataset with 100 points
set.seed(116)
n = 100
X1 <- runif(n,0,1)
X2 <- runif(n,0,1)

#I'm going to use a for loop to generate 100 y's
Y <- rep(0,n) #initializing my Y to be a vector of 0's
for (i in 1:n) {
  Y[i] <- generate_y(X1[i],X2[i])
}

sum(Y) #How many 0's and 1's were predicted? In this training set, 42% were 1's. However, alr
```

```
[1] 42
```

```
training <- cbind(X1,X2,Y) #combining all of my variables into a training dataset
ggplot(data=training, aes(x=X1, y=X2, color=Y)) +
  geom_point()
```



This generating function seems to produce Y's with some spatial pattern in the X1, X2 parameter space, but the regions of 0's and 1's are not very well separated. How well will KNN do to classify/predict Y given new x1 and x2 values?

Test dataset

We are going to generate a new set of 100 predictors (x1, x2) and outcomes (y) that we will use as our "ground truth".

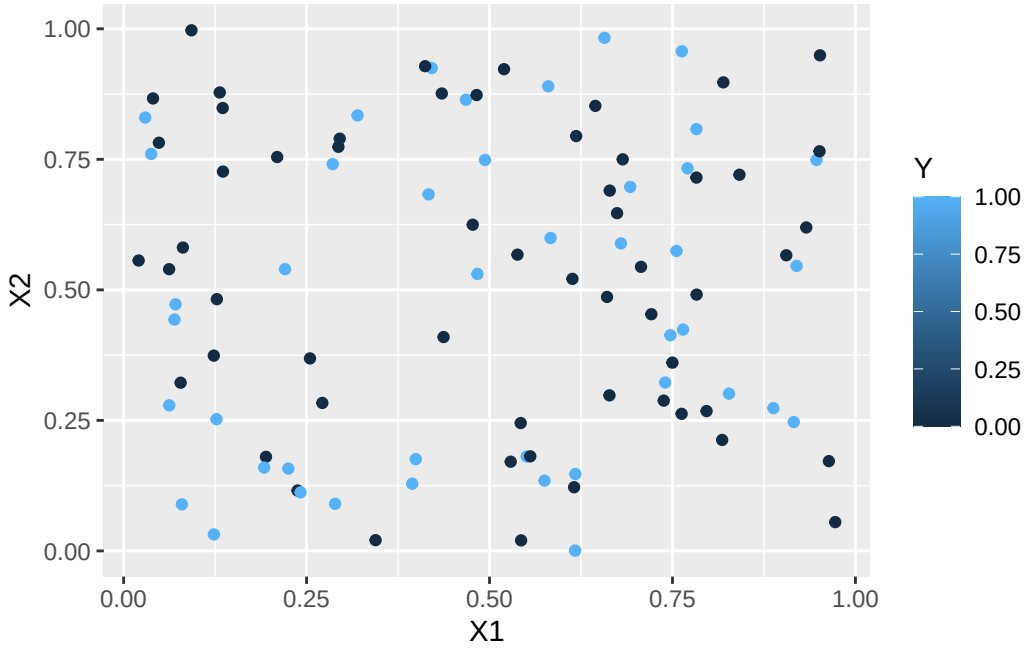
So, create the test dataset (using random seed=121) first.

```
# Create the training dataset as above using seed=116
# Create a testing dataset using seed=121

set.seed(121)
n = 100
X1 <- runif(n,0,1)
```

```r
X2 <- runif(n,0,1)

#I'm going to use a for loop to generate 100 y's
Y <- rep(0,n) #initializing my Y to be a vector of 0's
for (i in 1:n) {
  Y[i] <- generate_y(X1[i],X2[i])
}

sum(Y) #43 1's, which is much closer to the 51.5% true rate
```

```
[1] 43
```

```r
testing <- cbind(X1,X2,Y)

#Let's plot the test set. Does it look like the training set? Yeah, looks similar.
ggplot(data=testing, aes(x=X1, y=X2, color=Y)) +
  geom_point()
```



What individuals need to do

1. Given the training set (seed=116) and the testing set (seed=121), fit KNN on two different values of k.

2. Calculate the misclassification rate for each k. If you don't know how to do this, ask a teammate or the professor.
3. If possible, plot the decision boundaries for your k values.
4. Summarize the Q1, Q2, and Q3 aspects of this "project." Use your imagination. Everyone should have a different scenario.
5. Think about and discuss with your team some of the bonus questions below.

What teams need to do

1. Find the best k. Plot the decision boundary for that k.
2. Answer as many of the bonus questions as you can.

Bonus questions

Ultimately, we would like to know more about this problem, but we don't necessarily have all the tools yet to answer all of our questions. Here are some bonus questions:

- What is the misclassification rate for k=1…2j, where j is the "optimal" k. That is, for a whole range of k values?
- What is the Bayes decision boundary for the optimal k?
- How is the misclassification rate broken down into variance, bias, and irreducible error of the KNN estimator? Related to these questions are how the misclassification rate changes when we use different training or test sets.
- What happens to the misclassification rate if we go outside the $[0, 1]^2$ parameter space? I.e., if x_1 and x_2 are < 0 or > 1?

Note: the intended audience for your team document is your teammates, the professor, and your future self.

Some intended outcomes from this assignment:

- You and your team will learn how to effectively edit a document collaboratively
- You will think about what you want to accomplish in life and how this course relates to that
- Your teammates, the professor, and the TA will learn something more about you
- You will get more experience applying statistical learning methods
- You will gain experience collaborating with your teammates on an applied problem
- Get practice evaluating a method, specifically KNN
- Practice thinking about the whole problem (i.e., Q1Q2Q3, not just the Q2 quantitative parts)

Individual answers

Jacob Krol

Non-technical answers section (Jake)

Photo of myself (Jake)

Jake visiting a temple in Thailand.

What non-statistic question would I like to answer using statistical learning? (Jake)

The non-statistics question I would like to answer is ...

Are there genefusion mutations driving human cancers that have yet to be discovered, and can we find them using sequencing data from thousands of people?

What would I like to do after graduation? (Jake)

Six months after graduation, I would love to have both stable income, start acquiring more financial assets, and start a family with my partner.

Five years after graduation, I would love to own a home, and I hope to balance my family and work time well.

What is my most desired career accomplishment? (Jake)

My most desired career accomplishment would be impactful publish papers or patents that directly improve diagnosis/treatment of genetic diseases.

How do my hopes and goals relate to the course? (Jake)

I am hoping to better understand the underlying math of statistical learning methods. Particularly, I am interested in models with high interpretability that I can apply to my Ph.D. research focused on human genetic diseases.

Additional facts about me (Jake)

I love to cook, but I am not very good at it.

Technical answers section (Jake)

Plausible story for KNN data (Jake)

- The outcome, Y, is a binary indicator of whether a patient has a disease. 0 means the patient does not have the disease, and 1 means the patient does have the disease.
- The variable X1 is the outcome of a diagnostic test (e.g., bloodwork) for a disease. The test score ranges from 0 to 1. 0 is the lowest confidence that the patient has the disease. 1 implies the highest confidence that the patient has the disease.
- The variable X2 is doctor's belief that the patient has the disease. Belief also ranges from 0 to 1 with 0 being the lowest confidence and 1 being the highest confidence.

A hosptial is considering using their historical electronic health records (EHR) data on

Q1,Q2,Q3 for story (Jake)

Q1 (Jake)

The domain of the problem is healthcare, specifically disease diagnosis. The goal of the study is to develop an accurate, automated approach to predict whether a patient has a disease given the diagnostic test outcome and the doctor's belief about their disease state. An important context is that diseases often have serious health implications that could change the patient's life. Both type 1 and type 2 error of diagnosis could cause major issues. False positives (Type I error) may lead to a harsh treatment being assigned to a patient without a disease. While for false negatives (Type II error), a patient's disease state may worsen because they do not receive timely treatment. The data about test results, doctor's beliefs, and disease states are collected from a hosptial's electronic health records (EHR) system. Breifly, the plan of the study is to use historical EHR from the past few years to evaluate how diagnostic tests and doctor belief states are at predicting the true disease state of a patient. There are many ethical considerations in this study. Here, I discuss only a few. Will doctors and patients know their EHR data is being used to develop a diagnostic model? Will both doctors and patients know when a model is being used to assist in diagnosis? What is an acceptable accuracy of a diagnostic model? Who will take the blame for incorrect model predictions? For which diseases is it acceptable to use a model for diagnosis?

Q2 (Jake)

Hypothetically, the data would be carefully extracted form the EHR system and processed into a clean 3 column table with columns X1, X2, and Y. 50% of the data would be used for training and 50% for testing. A scatterplot visualization with axes X1 and X2 colored by Y would be used to explore the data. For modeling, a KNN model would be fit to the data with k=3 and k=7. The misclassification rate would be calculated for both k values. Finally, we are interested in inferring the disease state for a new patient where both the test and doctor belief are in disease is 0.5. In addition to the ethical considerations for Q1, another ethical consideration is the choice of KNN which is has no parametric interpretability. All data processing, model fitting, evaluation, and inference would be contained in a single script for reproducibility.

Q3 (Jake)

Interpreted findings include the misclassification rates for k=3 and k=7, the decision boundary plots for both k values, and the predicted disease state for a new patient with test and doctor belief both equal to 0.5. Conclusions will be drawn by considering the misclassification rates and decision boundaries. If the misclassification rates are low, then KNN may be a good choice for disease diagnosis. If the decision boundaries are smooth and reasonable, then KNN may be a good choice for disease diagnosis. The predicted disease state for the new patient will be used to illustrate how the model can be used for inference. A key ethical implications is the potential consequences of incorrect model predictions on patient health. Furthermore, if misclassificaiton rate is low, then we may recommend using KNN as a diagnostic tool. A call to action is only necessary if th model proves to be accurate enough for clinical use. A reasonable implementation may be adapting this quarto document into a modular pipeline to automate integration of new EHR data, model fitting, evaluation, and inference.

KNN misclassification rates for k=3 and k=7 (Jake)

Train data.

```
library(class)
library(tidyverse)

set.seed(116) #setting a random seed so that we can reproduce everything exactly if we want t

generate_y <- function(x1,x2) { #two input parameters to generate the output y
  logit <- x1 -2*x2 -2*x1^2 + x2^2 + 3*x1*x2
  p <- exp(logit)/(1+exp(logit)) #apply the inverse logit function
  y <- rbinom(1,1,p) #y becomes a 0 (with prob 1-p) or a 1 with probability p
}
# Generate a dataset with 100 points
set.seed(116)
n = 100
X1_tr <- runif(n,0,1)
X2_tr <- runif(n,0,1)

#I'm going to use a for loop to generate 100 y's
Y_tr <- rep(0,n) #initializing my Y to be a vector of 0's
for (i in 1:n) {
  Y_tr[i] <- generate_y(X1_tr[i],X2_tr[i])
}

training <- cbind(X1_tr,X2_tr,Y_tr) #combining all of my variables into a training dataset
```

Test data

```
set.seed(121)
n = 100
X1_te <- runif(n,0,1)
X2_te <- runif(n,0,1)

Y_te <- rep(0,n) #initializing my Y to be a vector of 0's
for (i in 1:n) {
  Y_te[i] <- generate_y(X1_te[i],X2_te[i])
}

sum(Y_te) #43 1's, which is much closer to the 51.5% true rate
```

```
[1] 43
```

```
testing <- cbind(X1_te,X2_te,Y_te)
```

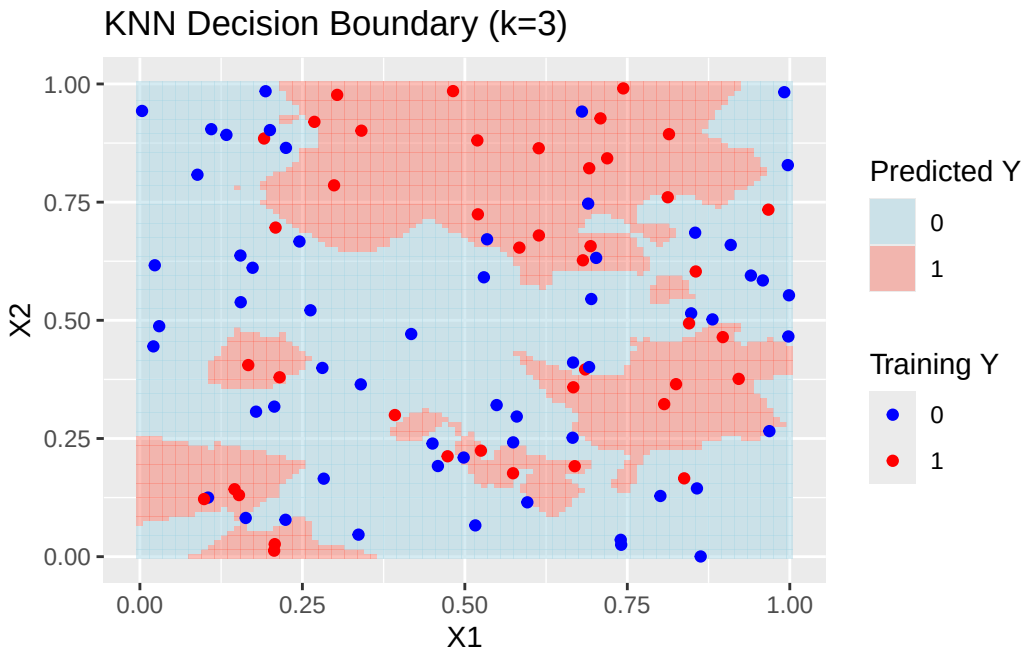
Evaluate KNN for k=3 and k=7.

```
ks=c(3,7)
mcrate <- c()

n_test <- nrow(testing)
for (k in ks) {
  y_pred <- knn(train = training[,1:2], test = testing[,1:2],
                        cl = training[,3], k = k)

  misclass_rate <- sum(y_pred != testing[,3]) / n_test
  mcrate <- c(mcrate, misclass_rate)
}
# print misclassification rates
for (i in 1:length(ks)) {
  cat("k =", ks[i], "misclassification rate:", mcrate[i], "\n")
}
```

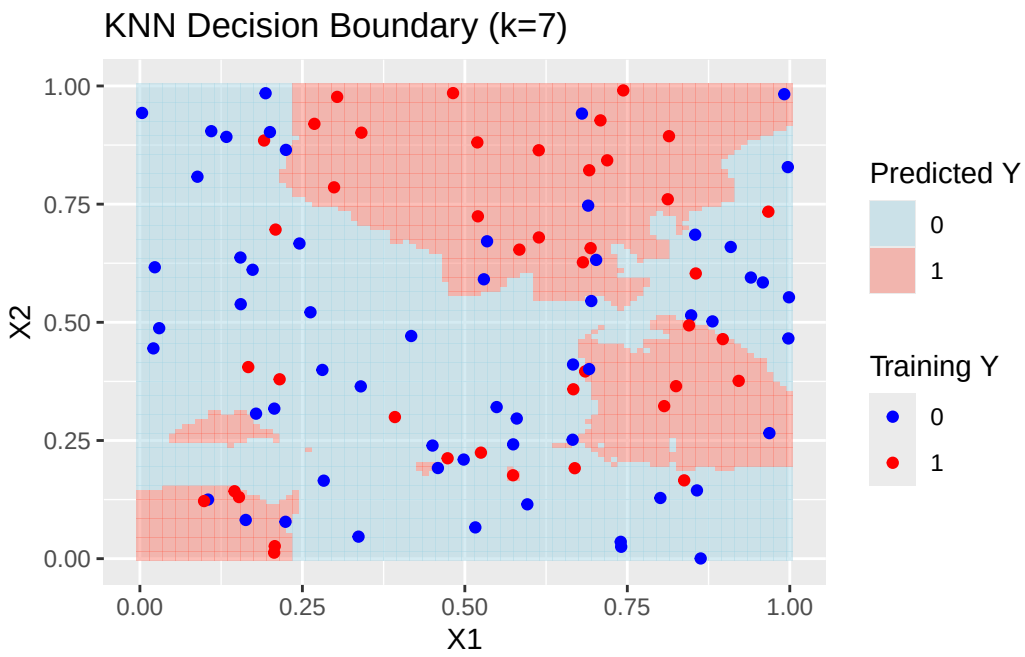
```
k = 3 misclassification rate: 0.47
k = 7 misclassification rate: 0.43
```

KNN decision boundary for k=3 and k=7

```
# grid unit square
test_grid <- expand.grid(
  X1 = seq(0, 1, by = 0.01),
  X2 = seq(0, 1, by = 0.01)
)

test_grid$y_pred_k3 <- knn(
  train= training[, 1:2],
  test= test_grid,
  cl= training[, 3],
  k=3
)
test_grid$y_pred_k7 <- knn(
  train= training[, 1:2],
  test= test_grid[, 1:2],
  cl= training[, 3],
  k=7
)

# plot db for k=3
ggplot() +
  geom_tile(data = test_grid, aes(x = X1, y = X2, fill = y_pred_k3), alpha = 0.5) +
  geom_point(data = as.data.frame(training), aes(x = X1_tr, y = X2_tr, color = as.factor(Y_t
  labs(title = "KNN Decision Boundary (k=3)", fill = "Predicted Y", color = "Training Y") +
  scale_fill_manual(values = c("0" = "lightblue", "1" = "salmon")) +
  scale_color_manual(values = c("0" = "blue", "1" = "red"))
```




```
# plot db for k=7
ggplot() +
  geom_tile(data = test_grid, aes(x = X1, y = X2, fill = y_pred_k7), alpha = 0.5) +
  geom_point(data = as.data.frame(training), aes(x = X1_tr, y = X2_tr, color = as.factor(Y_tr
  labs(title = "KNN Decision Boundary (k=7)", fill = "Predicted Y", color = "Training Y") +
  scale_fill_manual(values = c("0" = "lightblue", "1" = "salmon")) +
  scale_color_manual(values = c("0" = "blue", "1" = "red"))
```



KNN prediction for new data point (X1=0.5, X2=0.5)

```
new_data <- data.frame(X1 = 0.5, X2 = 0.5)
predicted_y_k3 <- knn(train = training[, 1:2],
                      test = new_data,
                      cl = training[, 3],
                      k = 3)
predicted_y_k7 <- knn(train = training[, 1:2],
                      test = new_data,
                      cl = training[, 3],
                      k = 7)
cat("Predicted Y for (X1=0.5, X2=0.5) with k=3:", predicted_y_k3, "\n")
```

```
Predicted Y for (X1=0.5, X2=0.5) with k=3: 1
```

```
cat("Predicted Y for (X1=0.5, X2=0.5) with k=7:", predicted_y_k7, "\n")
```

```
Predicted Y for (X1=0.5, X2=0.5) with k=7: 1
```

```
# 1 for both k=3 and k=7
```

Cleanup (Jake)

```
rm(list = ls())
```

Summary (Jake)

I used KNN to predict a binary disease state based on a diagnostic test result and doctor's belief. Misclassification rates were 0.47 and 0.43 for k=3 and k=7, respectively. Since both classes' features were generated from a random uniform distribution, I did not expect much better performance than a random model: misclassification rate = 0.5. Qualitatively, decision boundary plots show that misclassification often occurs near a boundary. Finally, I evaluated a new patient test point with diagnostic test and doctor's belief both at 0.5, KNN predicted a disease state of 1 for both k values. This is bad because X1,X2=0.5 implies that both the test and the doctor were uncertain about the diagnosis, yet the model is forced to pick a definite outcome; in this case, the model chose to diagnose the disease. In my opinion, the misclassification performance being close to a baseline model is not sufficient for real clinical use. Furthermore, the ethical implications of incorrect model predictions on patient health are serious. Therefore, I would not recommend using KNN as a diagnostic tool in this scenario.

Ryan Morreale

Non-technical answers section (Ryan)

- a photo of yourself with a caption explaining the context

This is a photo of me with my mom and 2 sisters this winter break in Puerto Vallarta, MX, taking a sunset selfie before dinner. It was my first time in Mexico and I had a great time.

- at least one (non-statistics) question you would like to know the answer to that could potentially be answered by applying statistical learning methods to data

I would like to know what kind of music makes me the most productive and maybe on a different line what kind of music gives me the most energy….

- what you would love to be doing six months after graduation and then five years after that (what would make you excited to be doing?)

I would like to be starting my first full time job by at least six months after graduation. By five years after that I would like to be doing a job that I would really be interested in doing this might be something like a sports analyst.

- what you hope your greatest career accomplishment will be

I don't have the need or want to be known to have done some crazy achievement but I want to make a good salary and hopefully climb the ranks a little, I would just like to be known for

being a good and kind person in any of my jobs while obviously still bringing value and doing something that interests me.

- and given these hopes and goals, what you are hoping to learn/accomplish/do in this course.

I'm hoping to better my regression skills, my coding, and in general learn things that I can take into a internship. I am excited to learn about Boosting and Random Forests because those sound like interesting regression tools.

- You must also include something of your own choosing not described above. Anything. Be creative!

Ranking my favorite indoor Rock Climbing activities:

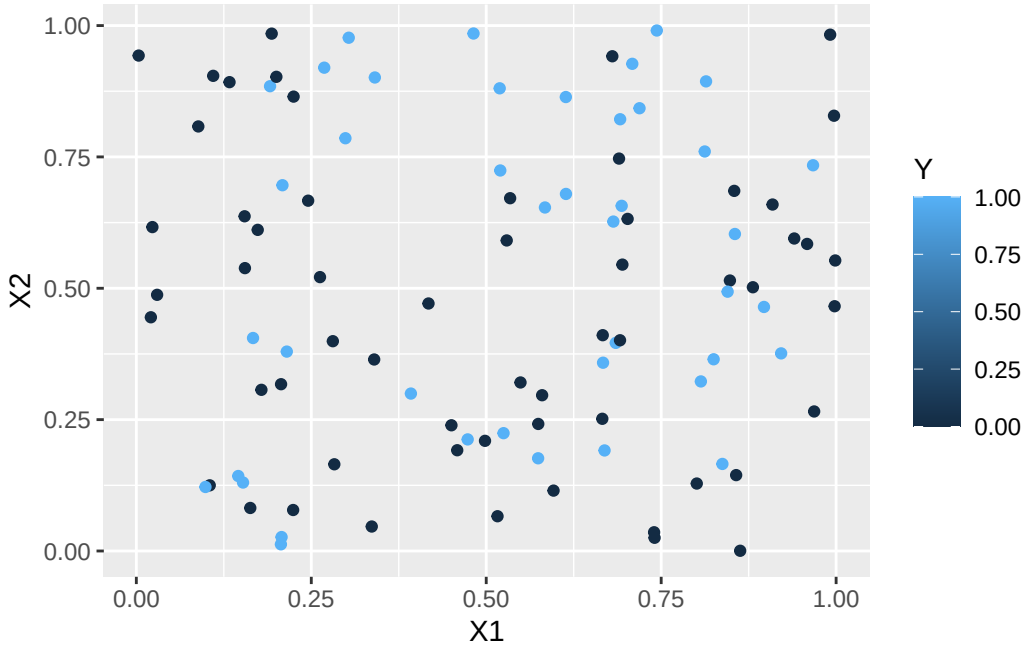
1. Top Rope
2. Bouldering
3. Strength Training, including things like hang boarding, grip strength, and pull ups (If this counts as a category. I think it does though)
4. Kilter/Moon Board
5. Lead Climbing (Only because I have never done it but hope to learn)

Technical answers section (Ryan)

```
set.seed(116)
n = 100
X1 <- runif(n,0,1)
X2 <- runif(n,0,1)

generate_y <- function(x1,x2) { #two input parameters to generate the output y
  logit <- x1 -2*x2 -2*x1^2 + x2^2 + 3*x1*x2
  p <- exp(logit)/(1+exp(logit)) #apply the inverse logit function
  y <- rbinom(1,1,p) #y becomes a 0 (with prob 1-p) or a 1 with probability p
}

#I'm going to use a for loop to generate 100 y's
Y <- rep(0,n) #initializing my Y to be a vector of 0's
for (i in 1:n) {
  Y[i] <- generate_y(X1[i],X2[i])
}

sum(Y) #How many 0's and 1's were predicted? In this training set, 42% were 1's. However, al
```

```
[1] 42
```

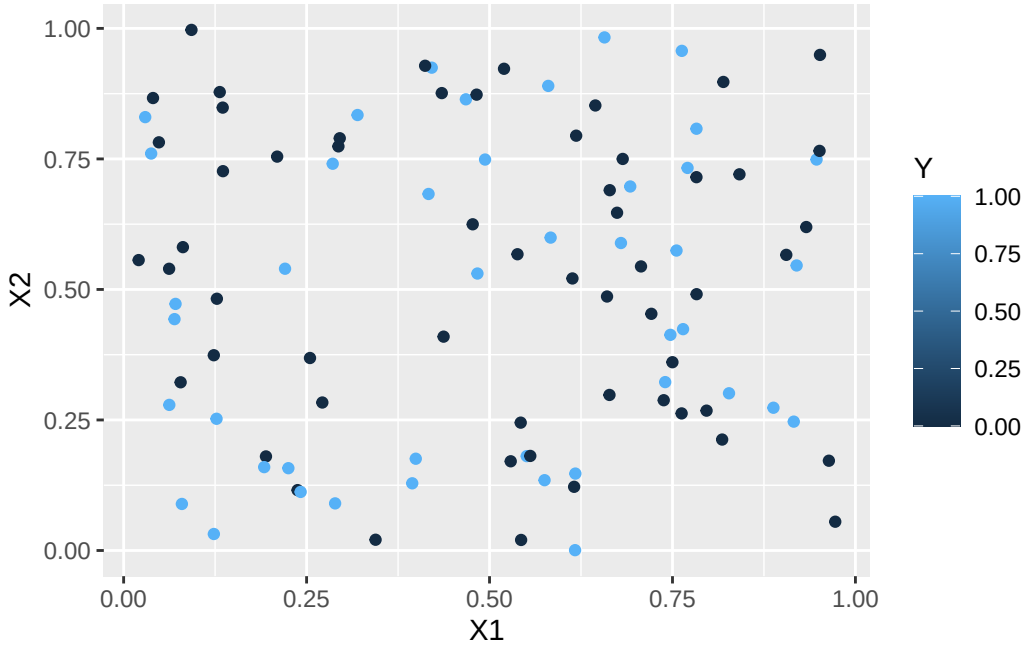
```
training <- cbind(X1,X2,Y) #combining all of my variables into a training dataset
ggplot(data=training, aes(x=X1, y=X2, color=Y)) +
  geom_point()
```




```
# Create a testing dataset using seed=121

set.seed(121)
n = 100
X1 <- runif(n,0,1)
X2 <- runif(n,0,1)

#I'm going to use a for loop to generate 100 y's
Y <- rep(0,n) #initializing my Y to be a vector of 0's
for (i in 1:n) {
  Y[i] <- generate_y(X1[i],X2[i])
}

sum(Y) #43 1's, which is much closer to the 51.5% true rate
```

```
[1] 43
```

```r
testing <- cbind(X1,X2,Y)

#Let's plot the test set. Does it look like the training set? Yeah, looks similar.
ggplot(data=testing, aes(x=X1, y=X2, color=Y)) +
  geom_point()
```




```r
predicted.ys1 <- knn(train = training[,1:2], test = testing[,1:2], cl=training[,3],k=1)
predicted.ys <- knn(train = training[,1:2], test = testing[,1:2], cl=training[,3],k=5)
```

Given the training set (seed=116) and the testing set (seed=121), fit KNN on two different values of k.

```r
testing.grid <- expand.grid(X1 = seq(0, 1, by = .01),
                            X2 = seq(0, 1, by = .01))

predicted.grid1 <- knn(train = training[,1:2],
                       test = testing.grid,
                       cl = training[,3],
                       k = 1)
predicted.grid5 <- knn(train = training[,1:2],
                       test = testing.grid,
                       cl = training[,3],
```

```
                      k = 5)

predicted.gridxy <- testing.grid
predicted.gridxy$Y1 <- as.numeric(predicted.grid1) -1
predicted.gridxy$Y5 <- as.numeric(predicted.grid5) -1
```

Calculate the misclassification rate for each k. If you don't know how to do this, ask a teammate or the professor.

```
misclass_rate <- function(predictions, truth) {
  1 - sum(predictions == truth) / length(predictions)
}

pred_k1 <- knn(train = training[,1:2],
               test = testing[,1:2],
               cl = training[,3],
               k = 1)

pred_k5 <- knn(train = training[,1:2],
               test = testing[,1:2],
               cl = training[,3],
               k = 5)

mis_k1 <- misclass_rate(pred_k1, testing[,3])
mis_k5 <- misclass_rate(pred_k5, testing[,3])

mis_k1
```

```
[1] 0.5
```

```
mis_k5
```

```
[1] 0.42
```

If possible, plot the decision boundaries for your k values.

```
ggplot() +
  # Background: Show the decision regions
  geom_raster(data = predicted.gridxy, aes(x = X1, y = X2, fill = as.factor(Y1)), alpha = 0.3
  # The Boundary: The contour line where Y changes
```

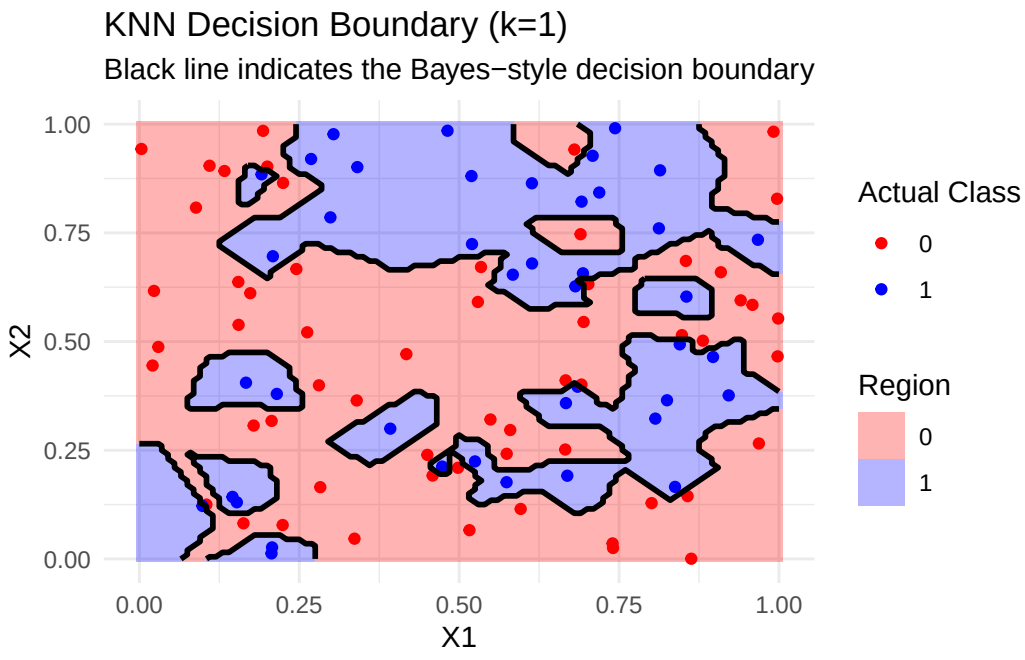
```
  geom_contour(data = predicted.gridxy, aes(x = X1, y = X2, z = Y1),
               breaks = 1.5, color = "black", size = 1) +
  # The actual data points
  geom_point(data = training, aes(x = X1, y = X2, color = as.factor(Y))) +
  scale_fill_manual(values = c("red", "blue"), name = "Region") +
  scale_color_manual(values = c("red", "blue"), name = "Actual Class") +
  labs(title = "KNN Decision Boundary (k=1)",
       subtitle = "Black line indicates the Bayes-style decision boundary") +
    geom_contour(data = predicted.gridxy, aes(x = X1, y = X2, z = Y1 +1),
               breaks = 1.5, color = "black", size = 1) +
theme_minimal()
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
i Please use `linewidth` instead.
```

```
Warning: `stat_contour()`: Zero contours were generated
```

```
Warning in min(x): no non-missing arguments to min; returning Inf
```

```
Warning in max(x): no non-missing arguments to max; returning -Inf
```

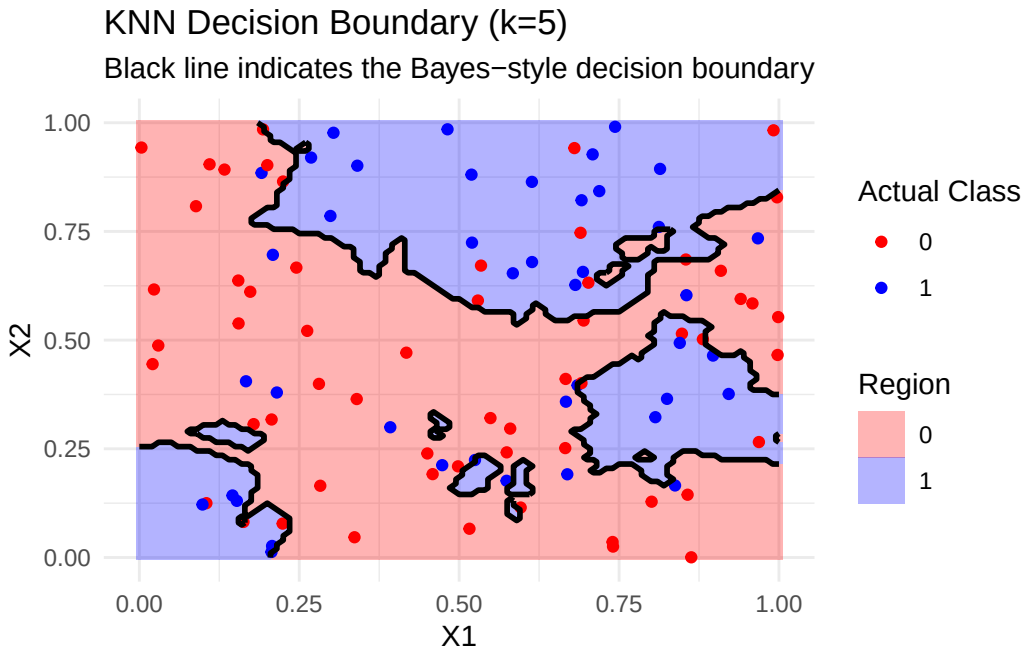


```r
ggplot() +
  # Background: Show the decision regions
  geom_raster(data = predicted.gridxy, aes(x = X1, y = X2, fill = as.factor(Y5)), alpha = 0.3
  # The Boundary: The contour line where Y changes
  geom_contour(data = predicted.gridxy, aes(x = X1, y = X2, z = Y5),
               breaks = 1.5, color = "black", size = 1) +
  # The actual data points
  geom_point(data = training, aes(x = X1, y = X2, color = as.factor(Y))) +
  scale_fill_manual(values = c("red", "blue"), name = "Region") +
  scale_color_manual(values = c("red", "blue"), name = "Actual Class") +
  labs(title = "KNN Decision Boundary (k=5)",
       subtitle = "Black line indicates the Bayes-style decision boundary") +
    geom_contour(data = predicted.gridxy, aes(x = X1, y = X2, z = Y5 +1),
               breaks = 1.5, color = "black", size = 1) +
theme_minimal()
```

```
Warning: `stat_contour()`: Zero contours were generated
```

```
Warning in min(x): no non-missing arguments to min; returning Inf
```

```
Warning in max(x): no non-missing arguments to max; returning -Inf
```



cleanup

```
rm(list = ls())
```

Summarize the Q1, Q2, and Q3 aspects of this "project." Use your imagination. Everyone should have a different scenario.

Q1. Y can represent whether a soccer shot results in a goal 1=goal, 0=no goal). X1= Shot Distance(0-1 where larger value indicates a farther shot) and X2= Shot Power (0-1 where larger value indicates harder shots). This problem is interesting because teams and analysts often want to know how shot location and shot power influence scoring probability. If we can predict the likelihood of a shot going in, we could evaluate player decision making. An example of this would be if a player shot from 18 yards out (X1=0.18) with a shot power of 50mph (X2=0.5), can we estimate whether it results in a goal.

Q2. To model the relationship between distance, power, and scoring, we applied a K Neares Neighbors classifer. Using the simulated data as training examples, we predicted outcomes for a test set using two values of k, k=1 and k=5. The resulting misclassification rate for k=1 is 0.50 and k=5 is 0.42. For this scenario, increasing k improved predictive accuracy, likely due to smoothing the decision boundary aka. reducing sensitivity to noise. This suggests that performance may benefit from moderate levels of regularization.

Q3. Given X1=0.5 and X2=0.5 the model predicts for both k values 0, which indicates that they would not score. However there are ethical considerations. A model based only on measurable attributes like shot speed and distance ignores contextual variables such as other context like defender pressure or player fatigue. It can also undervalue players strengths.

Emery Berry

Technical answers section (Emery)

```
library(class)
library(tidyverse)
```

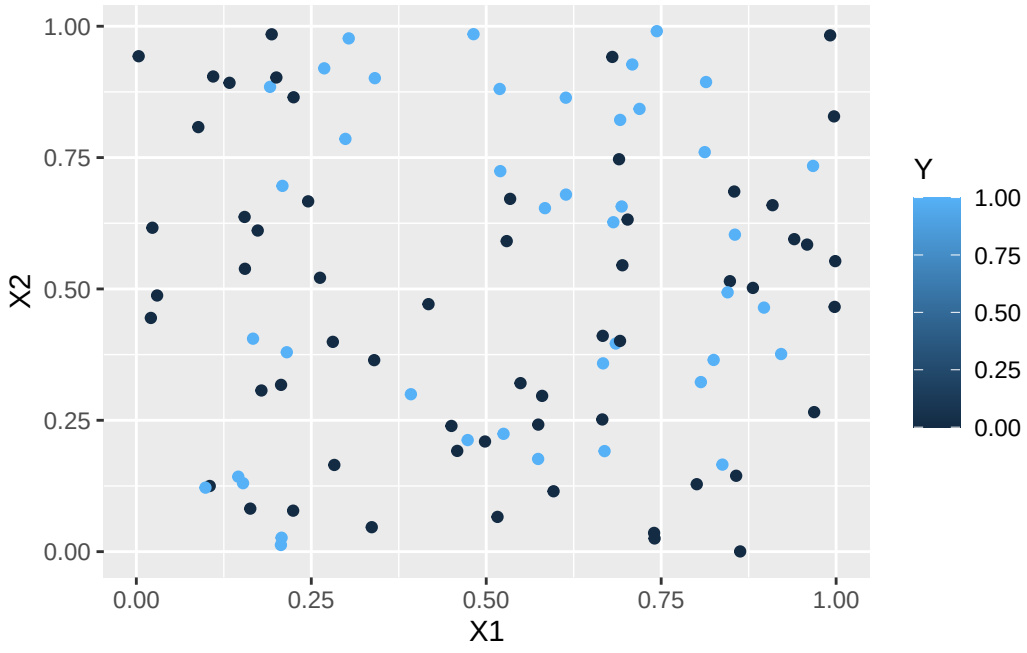
```
#Generative model
set.seed(116) #setting a random seed so that we can reproduce everything exactly if we want t

generate_y <- function(x1,x2) { #two input parameters to generate the output y
  logit <- x1 -2*x2 -2*x1^2 + x2^2 + 3*x1*x2
  p <- exp(logit)/(1+exp(logit)) #apply the inverse logit function
  y <- rbinom(1,1,p) #y becomes a 0 (with prob 1-p) or a 1 with probability p
}
```

Generate the training dataset.

```
# Generate a dataset with 100 points
set.seed(116)
n = 100
X1 <- runif(n,0,1)
X2 <- runif(n,0,1)

#I'm going to use a for loop to generate 100 y's
Y <- rep(0,n) #initializing my Y to be a vector of 0's
for (i in 1:n) {
  Y[i] <- generate_y(X1[i],X2[i])
}

sum(Y) #How many 0's and 1's were predicted? In this training set, 42% were 1's. However, alr
```

```
[1] 42
```

```
training <- cbind(X1,X2,Y) #combining all of my variables into a training dataset
ggplot(data=training, aes(x=X1, y=X2, color=Y)) +
  geom_point()
```

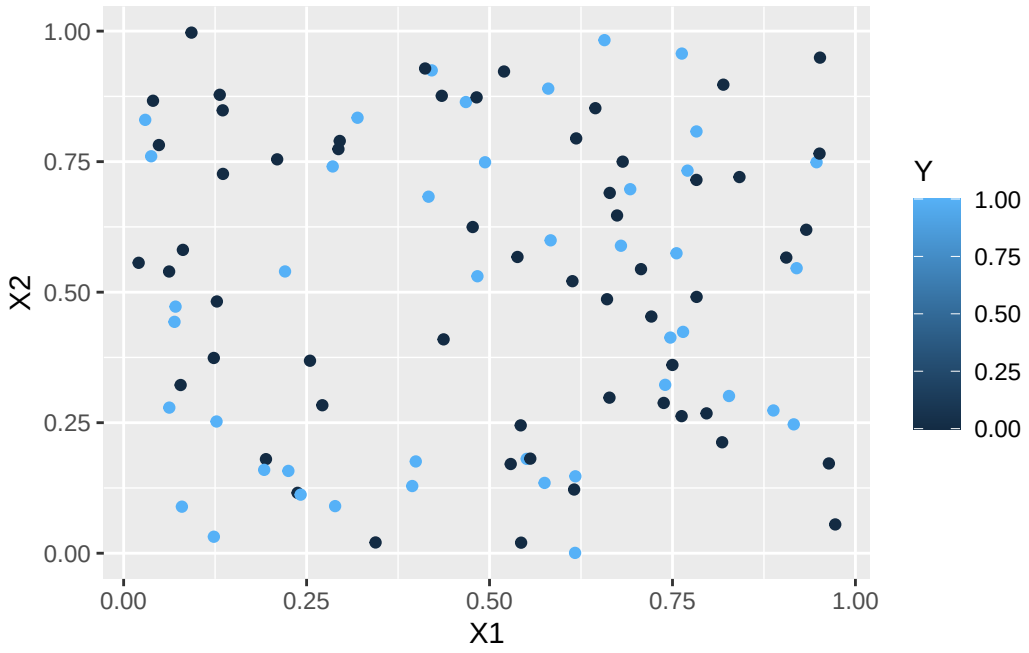


Create the test dataset (using random seed=121)

```r
# Create the training dataset as above using seed=116
# Create a testing dataset using seed=121

set.seed(121)
n = 100
X1 <- runif(n,0,1)
X2 <- runif(n,0,1)

#I'm going to use a for loop to generate 100 y's
Y <- rep(0,n) #initializing my Y to be a vector of 0's
for (i in 1:n) {
  Y[i] <- generate_y(X1[i],X2[i])
}

sum(Y) #43 1's, which is much closer to the 51.5% true rate
```

```
[1] 43
```

```r
testing <- cbind(X1,X2,Y)

#Let's plot the test set. Does it look like the training set? Yeah, looks similar.
ggplot(data=testing, aes(x=X1, y=X2, color=Y)) +
  geom_point()
```



Fit KNN for k=4 and k=8.

```
train_X <- training[,1:2]
test_X  <- testing[,1:2]
train_Y <- training[,3]


KNN_k4 <- knn(train = train_X, test = test_X, cl = train_Y, k = 4)
KNN_k8 <- knn(train = train_X, test = test_X, cl = train_Y, k = 8)
```

Evaluate misclassification rate for each k.

```
true_Y <- testing[,3]

misclass_rate <- function(predictions, test.truth) { #Given the model predictions and the tru
  1-(sum(predictions==test.truth)/length(predictions))
}

misclass_rate(KNN_k4, true_Y)
```

```
[1] 0.41
```

```r
misclass_rate(KNN_k8, true_Y)
```

```
[1] 0.38
```

Plot decision boundary.

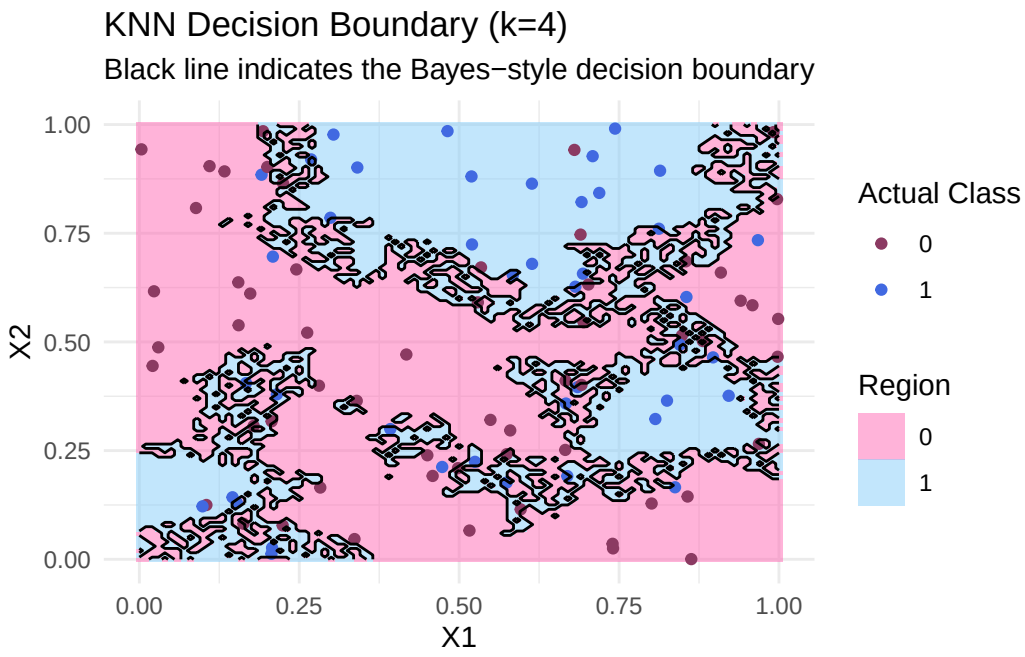
```r
# 1. Setup the Grid correctly
# Ensure the grid names match your training column names (X1, X2)
testing.grid <- expand.grid(X1 = seq(0, 1, by = .01),
                            X2 = seq(0, 1, by = .01))

# 2. Run KNN on the grid
# We convert the output to numeric so geom_contour can read it
predicted.grid.4 <- knn(train = train_X,
                        test = testing.grid,
                        cl = train_Y,
                        k = 4)
predicted.grid.8 <- knn(train = train_X,
                          test = testing.grid,
                          cl = train_Y,
                          k = 8)

# 3. Combine grid coordinates with predictions
predicted.gridxy.4 <- testing.grid
predicted.gridxy.4$Y <- as.numeric(predicted.grid.4) -1

predicted.gridxy.8 <- testing.grid
predicted.gridxy.8$Y <- as.numeric(predicted.grid.8) -1


# Plotting K=4
ggplot() +
  # Background: Show the decision regions
  geom_raster(data = predicted.gridxy.4, aes(x = X1, y = X2, fill = as.factor(Y)), alpha = 0
  # The Boundary: The contour line where Y changes
  geom_contour(data = predicted.gridxy.4, aes(x = X1, y = X2, z = Y),
               breaks = 1.5, color = "black", size = 0.5) +
  # The actual data points
  geom_point(data = training, aes(x = X1, y = X2, color = as.factor(Y))) +
  scale_fill_manual(values = c("hotpink", "lightskyblue"), name = "Region") +
  scale_color_manual(values = c("hotpink4", "royalblue"), name = "Actual Class") +
```

```
  labs(title = "KNN Decision Boundary (k=4)",
       subtitle = "Black line indicates the Bayes-style decision boundary") +
    geom_contour(data = predicted.gridxy.4, aes(x = X1, y = X2, z = Y +1),
                 breaks = 1.5, color = "black", size = 0.5) + # Add 1 to Y to make the contour
theme_minimal()
```

```
Warning: `stat_contour()`: Zero contours were generated
```

```
Warning in min(x): no non-missing arguments to min; returning Inf
```

```
Warning in max(x): no non-missing arguments to max; returning -Inf
```



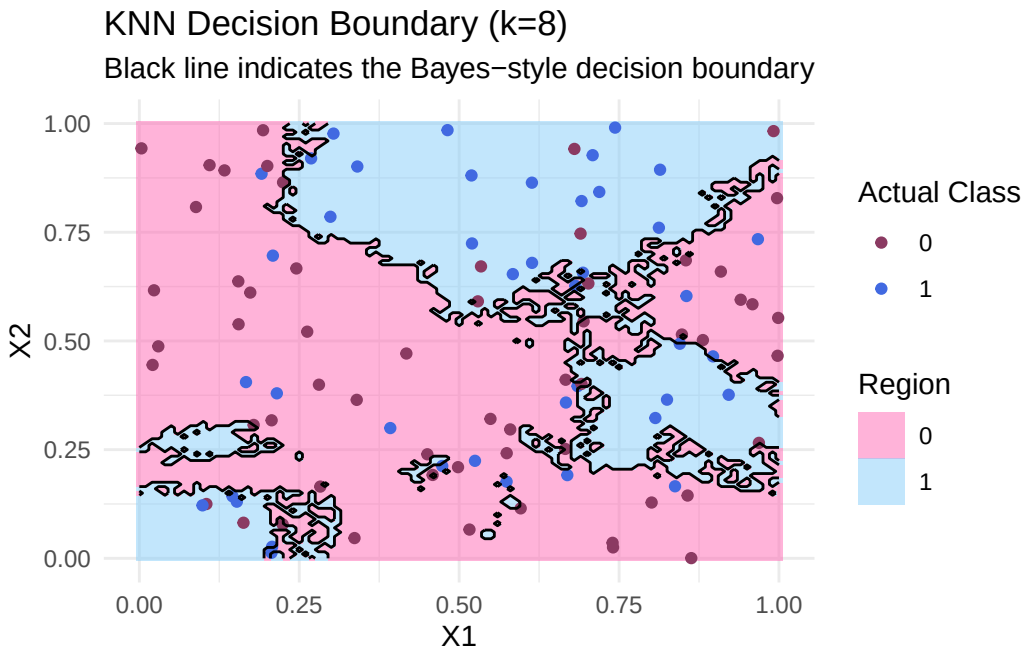

```
#Plotting K=8
ggplot() +
  # Background: Show the decision regions
  geom_raster(data = predicted.gridxy.8, aes(x = X1, y = X2, fill = as.factor(Y)), alpha = 0
  # The Boundary: The contour line where Y changes
  geom_contour(data = predicted.gridxy.8, aes(x = X1, y = X2, z = Y),
               breaks = 1.5, color = "black", size = 0.5) +
  # The actual data points
  geom_point(data = training, aes(x = X1, y = X2, color = as.factor(Y))) +
```

```
  scale_fill_manual(values = c("hotpink", "lightskyblue"), name = "Region") +
  scale_color_manual(values = c("hotpink4", "royalblue"), name = "Actual Class") +
  labs(title = "KNN Decision Boundary (k=8)",
       subtitle = "Black line indicates the Bayes-style decision boundary") +
    geom_contour(data = predicted.gridxy.8, aes(x = X1, y = X2, z = Y +1),
                 breaks = 1.5, color = "black", size = 0.5) + # Add 1 to Y to make the contour
theme_minimal()
```

```
Warning: `stat_contour()`: Zero contours were generated
```

```
Warning in min(x): no non-missing arguments to min; returning Inf
```

```
Warning in max(x): no non-missing arguments to max; returning -Inf
```



Cleanup

```
rm(list = ls())
```

Team applied answers

The minimum misclassification rate (0.38) was acheived for k=8. Jake evaluated k=3,7 with misclassification rates of 0.47 and 0.43, respectively. Ryan evaluated k=1,5 with misclassification rates of 0.5 and 0.42, respectively. And, Emery evaluated k=4,8 with misclassification rates of 0.41 and 0.38. We plotted the decision boundary for k=8 below.

```
library(class)
library(tidyverse)
K=8
# train
set.seed(116)
n = 100
X1_tr <- runif(n,0,1)
X2_tr <- runif(n,0,1)
generate_y <- function(x1,x2) {
  logit <- x1 -2*x2 -2*x1^2 + x2^2 + 3*x1*x2
  p <- exp(logit)/(1+exp(logit))
  y <- rbinom(1,1,p)
}
Y_tr <- rep(0,n)
for (i in 1:n) {
  Y_tr[i] <- generate_y(X1_tr[i],X2_tr[i])
}
training <- cbind(X1_tr,X2_tr,Y_tr)

# test
set.seed(121)
n = 100
X1_te <- runif(n,0,1)
X2_te <- runif(n,0,1)
Y_te <- rep(0,n)
for (i in 1:n) {
  Y_te[i] <- generate_y(X1_te[i],X2_te[i])
}
testing <- cbind(X1_te,X2_te,Y_te)
# plot decision boundary for k=8

test_grid <- expand.grid(
  X1 = seq(0, 1, by = 0.01),
  X2 = seq(0, 1, by = 0.01)
)
```

```r
test_grid$y_pred_k8 <- knn(
  train= training[, 1:2],
  test= test_grid,
  cl= training[, 3],
  k=8
)
ggplot() +
  geom_tile(data = test_grid, aes(x = X1, y = X2, fill = y_pred_k8), alpha = 0.5) +
  geom_point(data = as.data.frame(training), aes(x = X1_tr, y = X2_tr, color = as.factor(Y_tr
  labs(title = "KNN Decision Boundary (k=8)", fill = "Predicted Y", color = "Training Y") +
  scale_fill_manual(values = c("0" = "lightblue", "1" = "salmon")) +
  scale_color_manual(values = c("0" = "blue", "1" = "red"))
```