Duncan McKinnon

West

W 8

QA

```r
# Load R Packages
suppressPackageStartupMessages({
  library(tidyverse)
  library(NHANES)
  library(class)
  library(mosaic)
})
```

1,2,3).

```r
# create 1, 0 factors for RegularMarij and Gender, limit data to 11 variables
dNH <- NHANES %>%
  mutate(RM1 = ifelse(RegularMarij=='Yes', 1, 0), Sex1 = ifelse(Gender=='male', 1, 0)) %>%
  select(HardDrugs, RM1, Age, AlcoholYear, BMI, Sex1, HHIncomeMid, Weight, Height, SexNumPartnLife, Tot

# summarize new fields
dNH %>% select(RM1, Sex1) %>% summary()
```

```
##       RM1              Sex1      
##  Min.   :0.000   Min.   :0.000  
##  1st Qu.:0.000   1st Qu.:0.000  
##  Median :0.000   Median :0.000  
##  Mean   :0.276   Mean   :0.498  
##  3rd Qu.:1.000   3rd Qu.:1.000  
##  Max.   :1.000   Max.   :1.000  
##  NA's   :5059
```

```r
dNH %>% glimpse()
```

```
## Observations: 10,000
## Variables: 11
## $ HardDrugs       <fct> Yes, Yes, Yes, NA, Yes, NA, NA, No, No, No, No...
## $ RM1             <dbl> 0, 0, 0, NA, 0, NA, NA, 0, 0, 0, NA, 1, 1, NA,...
## $ Age             <int> 34, 34, 34, 4, 49, 9, 8, 45, 45, 45, 66, 58, 5...
## $ AlcoholYear     <int> 0, 0, 0, NA, 20, NA, NA, 52, 52, 52, 100, 104,...
## $ BMI             <dbl> 32.22, 32.22, 32.22, 15.30, 30.57, 16.82, 20.6...
## $ Sex1            <dbl> 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1...
## $ HHIncomeMid     <int> 30000, 30000, 30000, 22500, 40000, 87500, 6000...
## $ Weight          <dbl> 87.4, 87.4, 87.4, 17.0, 86.7, 29.8, 35.2, 75.7...
## $ Height          <dbl> 164.7, 164.7, 164.7, 105.4, 168.4, 133.1, 130....
## $ SexNumPartnLife <int> 8, 8, 8, NA, 10, NA, NA, 20, 20, 20, 15, 7, 10...
## $ TotChol         <dbl> 3.49, 3.49, 3.49, NA, 6.70, 4.86, 4.09, 5.82, ...
```

```r
# remove all NA's and show num left
dNH <- dNH %>% na.omit()

m <- dim(dNH)
m[1]
```

```
## [1] 3940
```

4).

```r
# set random seed
set.seed(1847)

# create function for splitting out data into train and test
split_data <- function(data, m, test_p) {
  idata <- list()
  idata$train_indices <- sample.int(m, (1 - test_p) * m)
  idata$train <- data[idata$train_indices, ]
  idata$test <- data[-idata$train_indices, ]
  return(idata)
}

# create train and test data
DNH <- split_data(dNH, m[1], 0.2)

# glimpse training and tests sets
glimpse(DNH$train)
```

```
## Observations: 3,152
## Variables: 11
## $ HardDrugs       <fct> No, No, No, No, No, No, Yes, Yes, Yes, No, No,...
## $ RM1             <dbl> 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 0...
## $ Age             <int> 52, 39, 31, 34, 21, 44, 54, 39, 51, 20, 59, 41...
## $ AlcoholYear     <int> 364, 52, 52, 260, 24, 104, 0, 52, 208, 24, 24,...
## $ BMI             <dbl> 33.52, 34.30, 29.70, 30.50, 40.00, 19.99, 23.5...
## $ Sex1            <dbl> 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0...
## $ HHIncomeMid     <int> 2500, 100000, 87500, 40000, 2500, 100000, 1250...
## $ Weight          <dbl> 82.0, 92.6, 77.6, 97.1, 103.7, 52.6, 66.6, 92....
## $ Height          <dbl> 156.4, 164.2, 161.6, 178.5, 161.0, 162.2, 168....
## $ SexNumPartnLife <int> 5, 5, 10, 100, 2, 50, 5, 30, 15, 20, 4, 10, 10...
## $ TotChol         <dbl> 6.03, 5.64, 4.24, 5.04, 4.06, 6.47, 6.59, 4.40...
```

```r
glimpse(DNH$test)
```

```
## Observations: 788
## Variables: 11
## $ HardDrugs       <fct> Yes, Yes, No, No, No, No, Yes, Yes, Yes, No, N...
## $ RM1             <dbl> 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1...
## $ Age             <int> 34, 49, 56, 56, 26, 51, 25, 21, 21, 43, 22, 48...
## $ AlcoholYear     <int> 0, 20, 12, 12, 104, 24, 104, 20, 20, 260, 52, ...
## $ BMI             <dbl> 32.22, 30.57, 19.73, 19.73, 21.00, 30.60, 27.0...
## $ Sex1            <dbl> 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1...
## $ HHIncomeMid     <int> 30000, 40000, 87500, 87500, 87500, 70000, 1000...
## $ Weight          <dbl> 87.4, 86.7, 57.5, 57.5, 64.9, 82.1, 86.3, 103....
## $ Height          <dbl> 164.7, 168.4, 170.7, 170.7, 175.8, 163.8, 178....
## $ SexNumPartnLife <int> 8, 10, 2, 2, 9, 8, 6, 4, 4, 3, 2, 6, 3, 14, 4,...
## $ TotChol         <dbl> 3.49, 6.70, 5.79, 5.79, 5.95, 4.53, 5.30, 4.24...
```

5 a-d).

```
# create function to re-run knn on the same data for many values of k
# return : list() with list() of the model test result values
#                       list() of the confusion matrices and prediction accuracy of each model
reKNN <- function(dtrain, dtest, train_labels, test_labels, ks){
  mods <- list()
  confs <- list()
  pcts <- list()

  for(i in ks){
    # re-run the modeling step on each value of k and generate confusion matrix
    model_predictions <- knn(train = dtrain, test = dtest, cl = train_labels, k = i)
    confusion_matrix <- table(test_labels, model_predictions)

    # get model accuracy from the confusion matrix
    acc_pct <- (confusion_matrix[1,1] + confusion_matrix[2,2]) / sum(confusion_matrix)

    # store results of modeling and confusion matrix/accuracy in list labeled by k used in model
    mods[[paste(i)]] <- model_predictions
    confs[[paste(i)]] <- list( 'confusion_matrix' = confusion_matrix, 'accuracy' = acc_pct )
  }
  return(list('model' = mods, 'performance' = confs))
}

# run reKNN for values of k
dtrn <- DNH$train %>% select(-HardDrugs)
dcltrn <- as.factor(DNH$train$HardDrugs)
dtst <- DNH$test %>% select(-HardDrugs)
dcltst <- as.factor(DNH$test$HardDrugs)
knns <- c(1, 5, 25, 100)

iknn <- reKNN(dtrn, dtst, dcltrn, dcltst, knns)

iknn$performance
```

```
## $`1`
## $`1`$confusion_matrix
##            model_predictions
## test_labels  No Yes
##         No  554  50
##         Yes  47 137
##
## $`1`$accuracy
## [1] 0.8769036
##
##
## $`5`
## $`5`$confusion_matrix
##            model_predictions
## test_labels  No Yes
##         No  545  59
##         Yes 105  79
##
## $`5`$accuracy
## [1] 0.7918782
##
##
## $`25`
## $`25`$confusion_matrix
##            model_predictions
## test_labels  No Yes
##         No  581  23
##         Yes 162  22
##
## $`25`$accuracy
## [1] 0.7652284
##
##
## $`100`
## $`100`$confusion_matrix
##            model_predictions
## test_labels  No Yes
##         No  603   1
##         Yes 183   1
##
## $`100`$accuracy
## [1] 0.7664975
```

6).

In the test of different k values (1,5,25,100) I found that k = 1 produced the most accurate predictions on this data set. The model produced with k=1 was almost 88% accurate in predictions on the test set, while k=5, 25 and 100 were all less than 80% accurate. This implies that the data may be so mixed in N dimensional space that only the most immediate neighbors are reasonable predictors of the overall class of the observation.

QB

1, 2).

```
# set random seed
set.seed(1847)

# create null model data
nullmod <- data.frame( y = sample(c(1,0), 500, replace = T),
                       x1 = runif(500, 0, 10),
                       x2 = runif(500, 0, 10))
glimpse(nullmod)
```

```
## Observations: 500
## Variables: 3
## $ y  <dbl> 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,...
## $ x1 <dbl> 8.3793318, 4.4838668, 2.8321497, 2.5175372, 8.5770707, 7.51...
## $ x2 <dbl> 4.67711044, 9.17958498, 5.94734839, 2.56939501, 7.70956941,...
```

```
# set random seed
set.seed(1847)

# create train and test data
p <- 0.2
m <- 500

train_indices_null <- sample.int(m, (1-p)*m)
train_null <- nullmod[train_indices_null, ]
test_null <- nullmod[-train_indices_null, ]

glimpse(train_null)
```

```
## Observations: 400
## Variables: 3
## $ y  <dbl> 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0,...
## $ x1 <dbl> 8.59285014, 9.37109700, 6.36942114, 8.83725895, 4.83748983,...
## $ x2 <dbl> 1.2488676, 9.7732755, 2.7639338, 6.9972891, 5.7779546, 9.05...
```

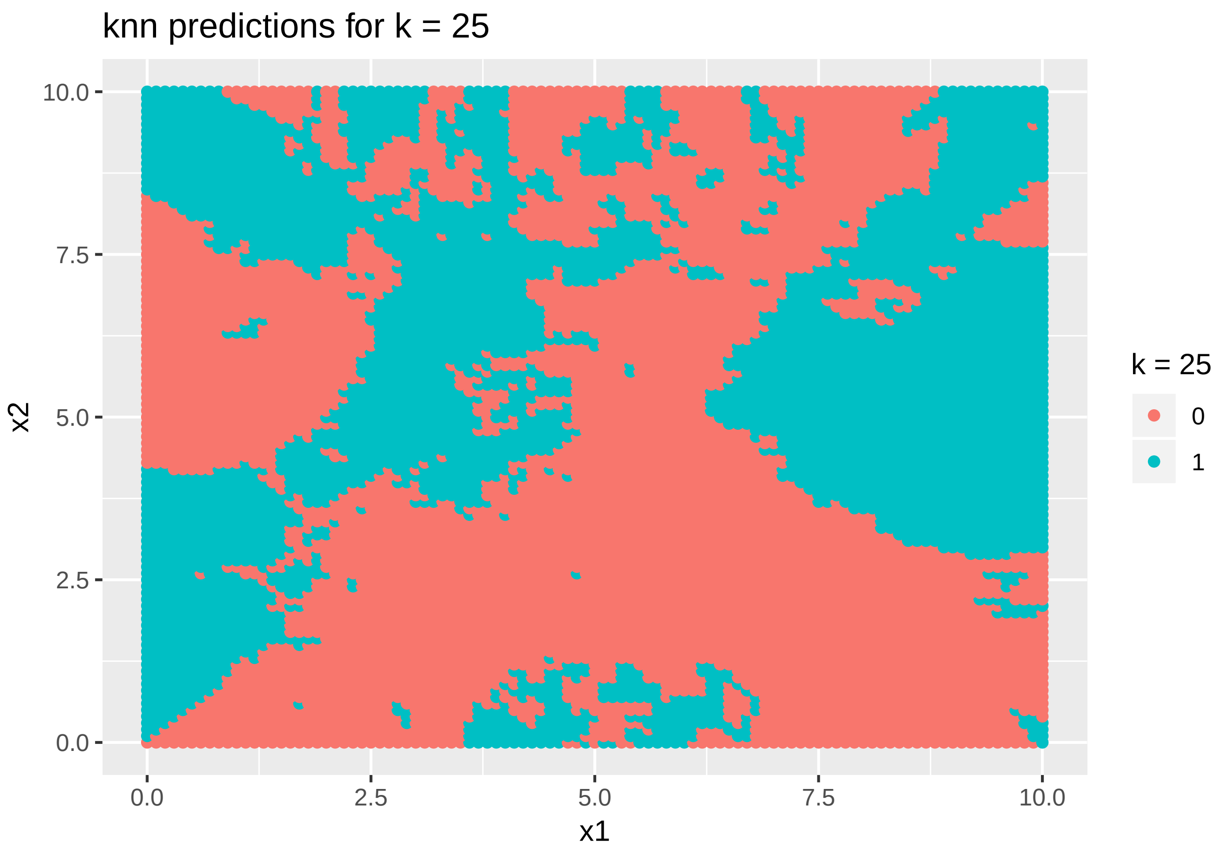
3).

```
# create grid data
grid_null <- expand.grid(x1 = seq(0, 10, 0.1), x2 = seq(0, 10, 0.1))

glimpse(grid_null)
```

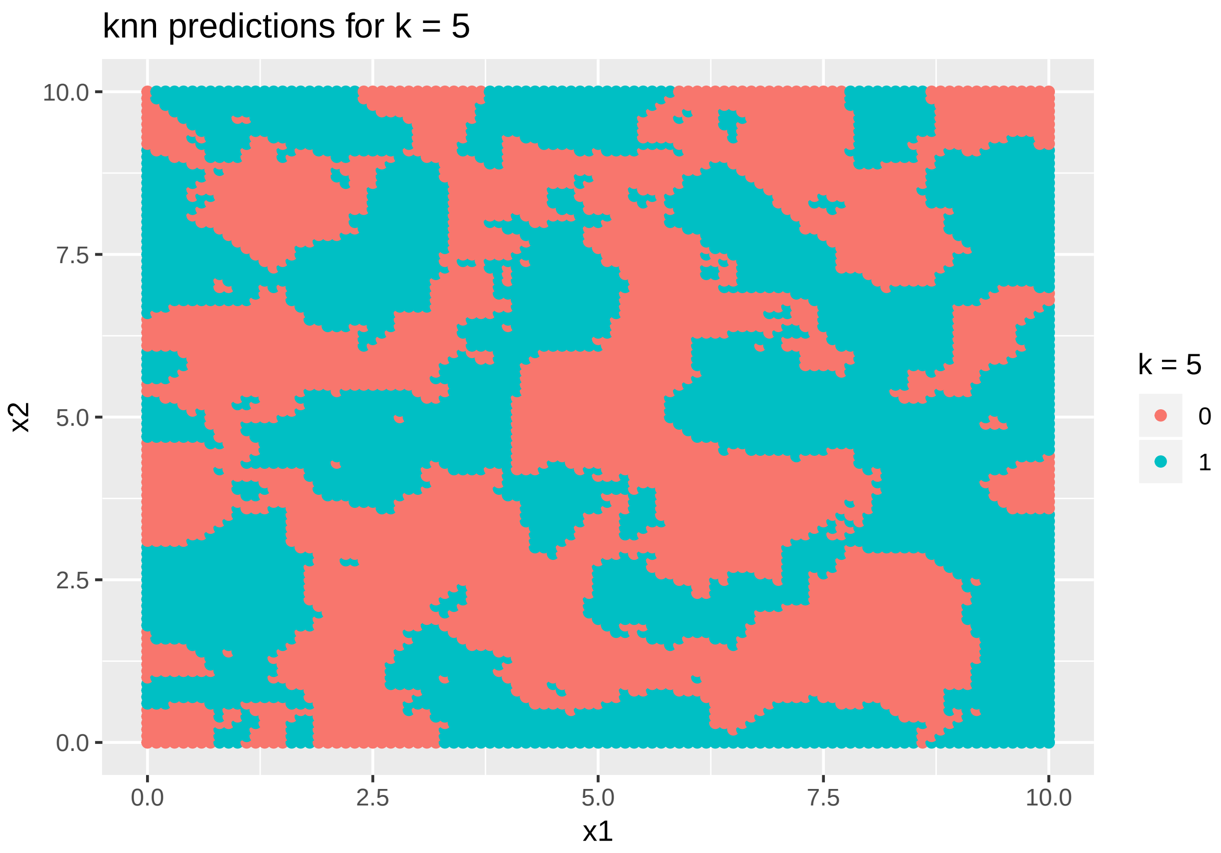
```
## Observations: 10,201
## Variables: 2
## $ x1 <dbl> 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1,...
## $ x2 <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...
```

4, 5, 6, (7)).

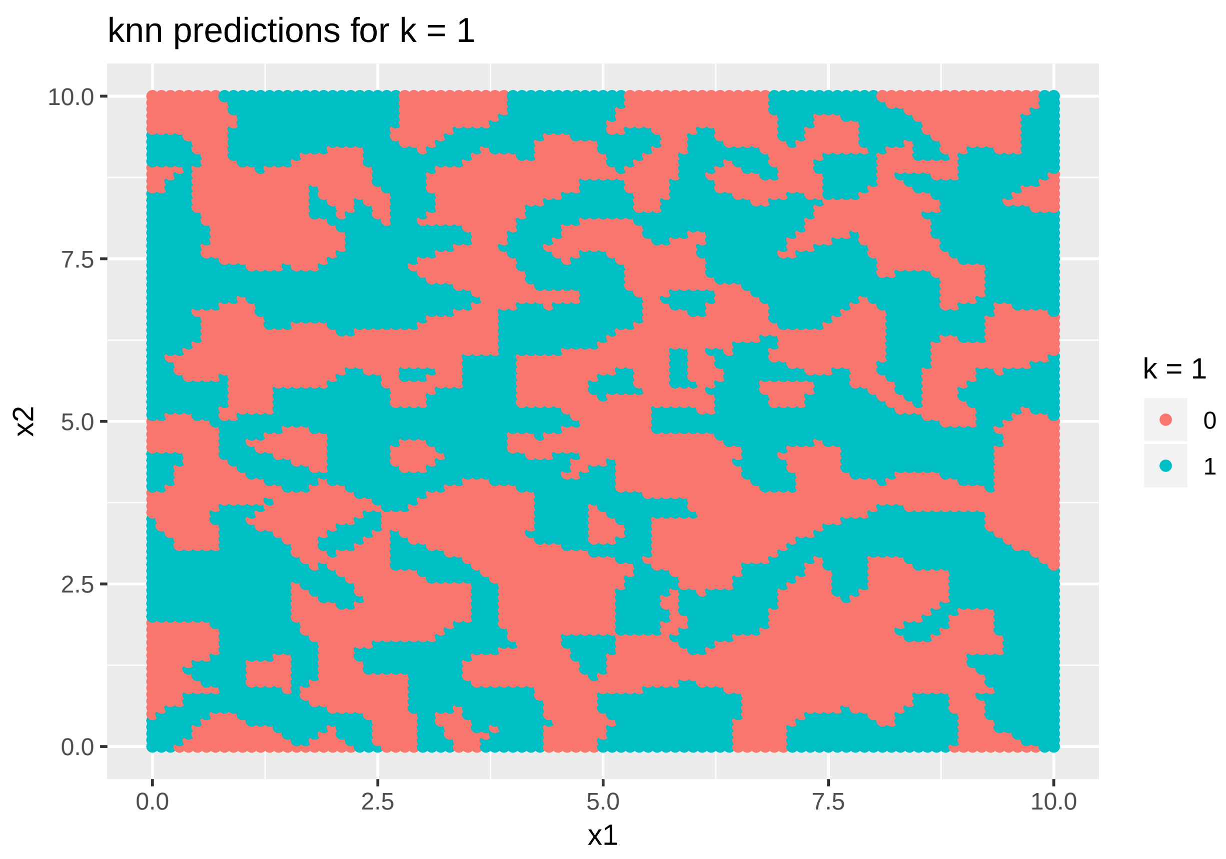
```
# split training data and labels
trn <- train_null %>% select(-y)
trn_labels <- factor(train_null$y)
kvs <- c(25,5,1,400)

# re-run the modeling step on each value of k and save the predictions to a list
mods <- list()
for(i in kvs){
  model_predictions <- knn(train = trn, test = grid_null, cl = trn_labels, k = i)
  mods[[i]] <- model_predictions
}

# plot the colors in the color grid using each set of model predictions
p <- ggplot(grid_null) + geom_point(aes(x = x1, y=x2))
p + aes( color = mods[[25]] ) + labs(title = 'knn predictions for k = 25', color = 'k = 25')
```



```
p + aes( color = mods[[5]] ) + labs(title = 'knn predictions for k = 5',color = 'k = 5')
```



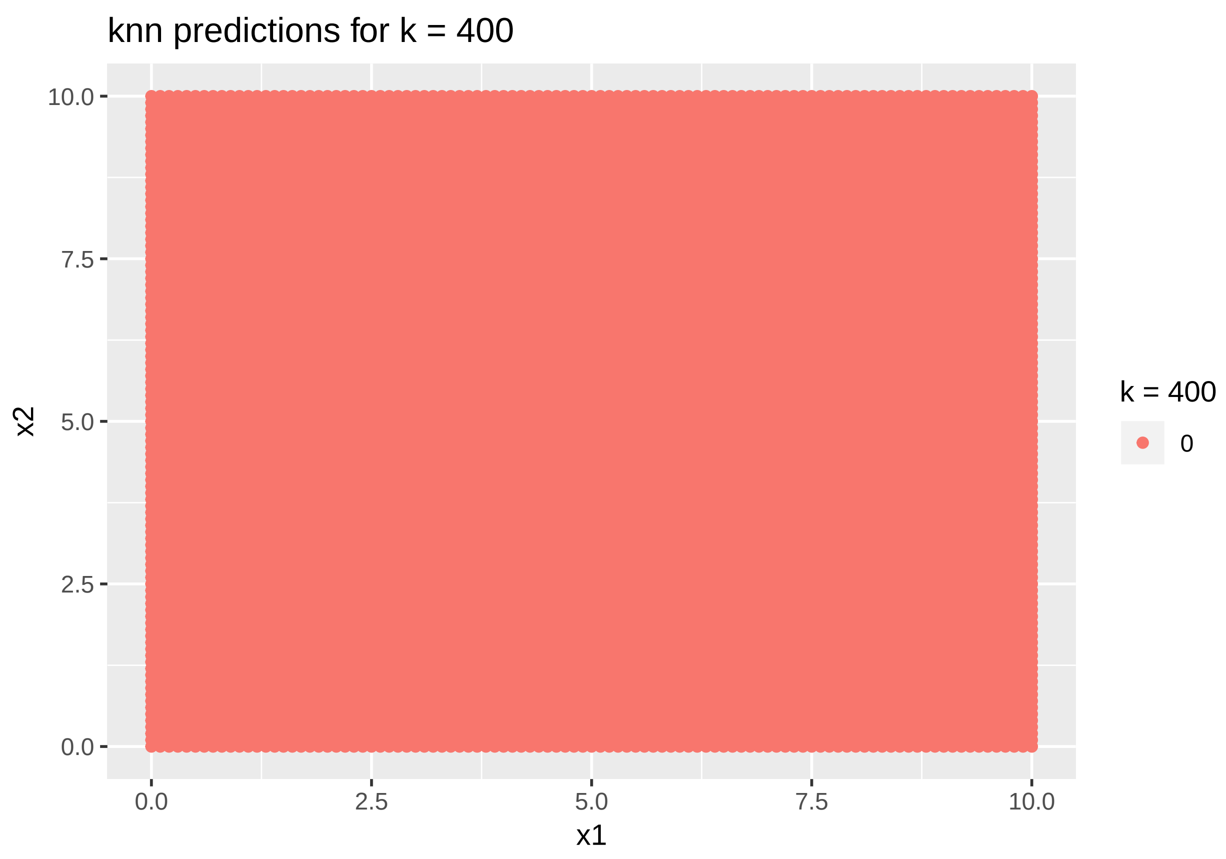
```
p + aes( color = mods[[1]] ) + labs(title = 'knn predictions for k = 1',color = 'k = 1')
```



7).

The plot using k = n will always be uninformative because every prediction for each observation will just be an aggregation of the labels of every point in the training data, since the nearest k = n training points will include all n points in the training data.

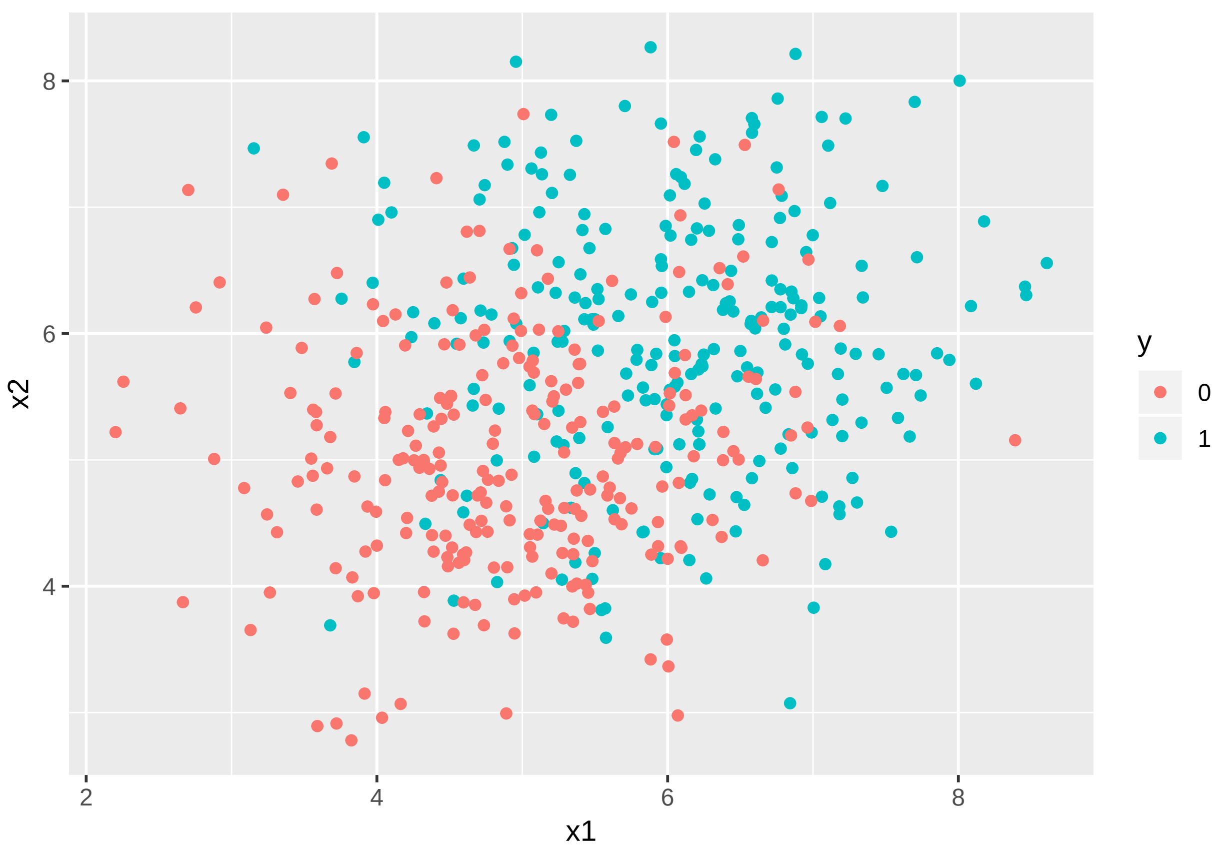
```
# plot the grid predictions for k = 400
p + aes( color = mods[[400]] ) + labs(title = 'knn predictions for k = 400',color = 'k = 400')
```



QC

1).

```
# set random seed
set.seed(1847)

# create x's and y in data frame
x1 <- c( rnorm(250, 6, 1), rnorm(250, 5, 1) )
x2 <- c( rnorm(250, 6, 1), rnorm(250, 5, 1) )
y <- as.factor(c( rep(1, 250), rep(0, 250) ))
dat_norm <- cbind.data.frame(x1, x2, y)

# plot data form
ggplot(dat_norm) + geom_point(aes(x = x1, y = x2, color = y))
```



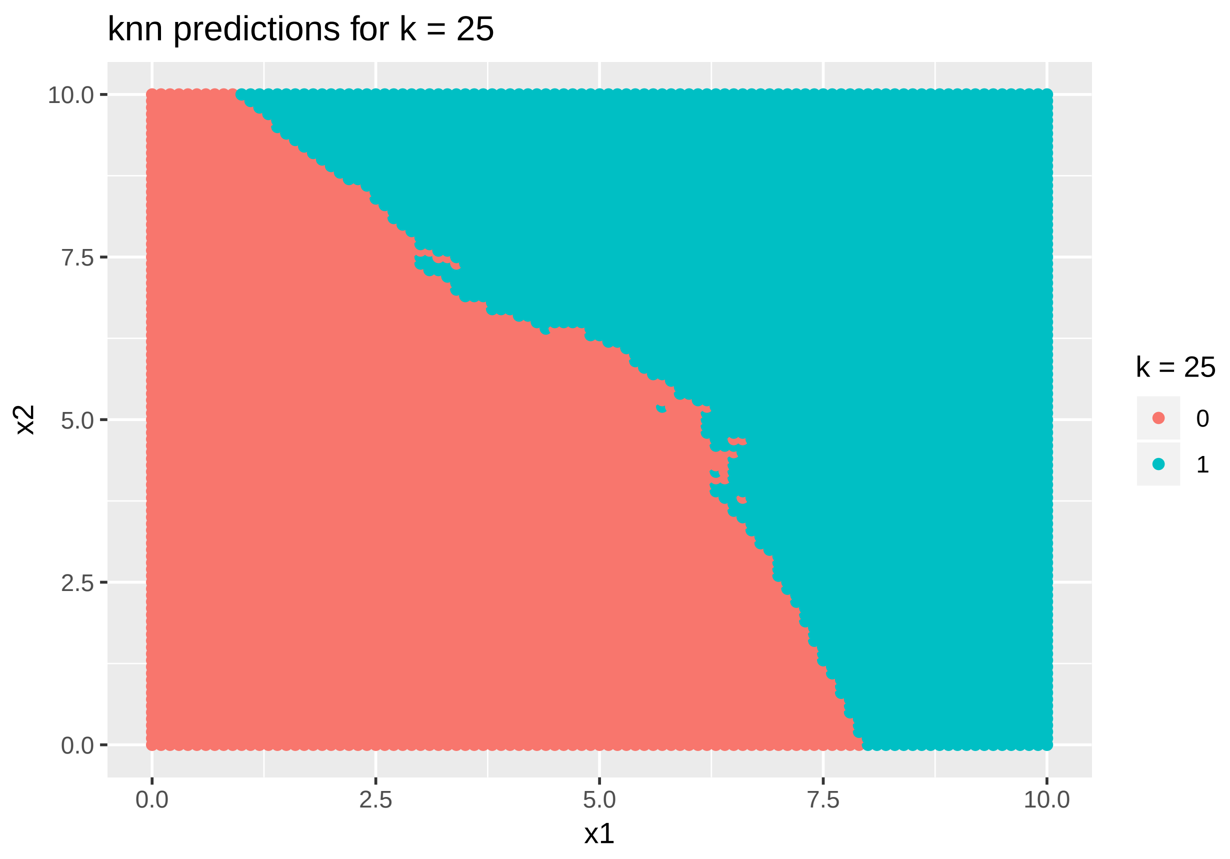
2).

```
# set random seed
set.seed(1847)
m <- 500
p <- 0.2

# split training and test set
indices_norm <- sample.int(m, (1-p)*m)
train_norm <- dat_norm[indices_norm, ]
test_norm <- dat_norm[-indices_norm, ]

# show training data
glimpse(train_norm)
```

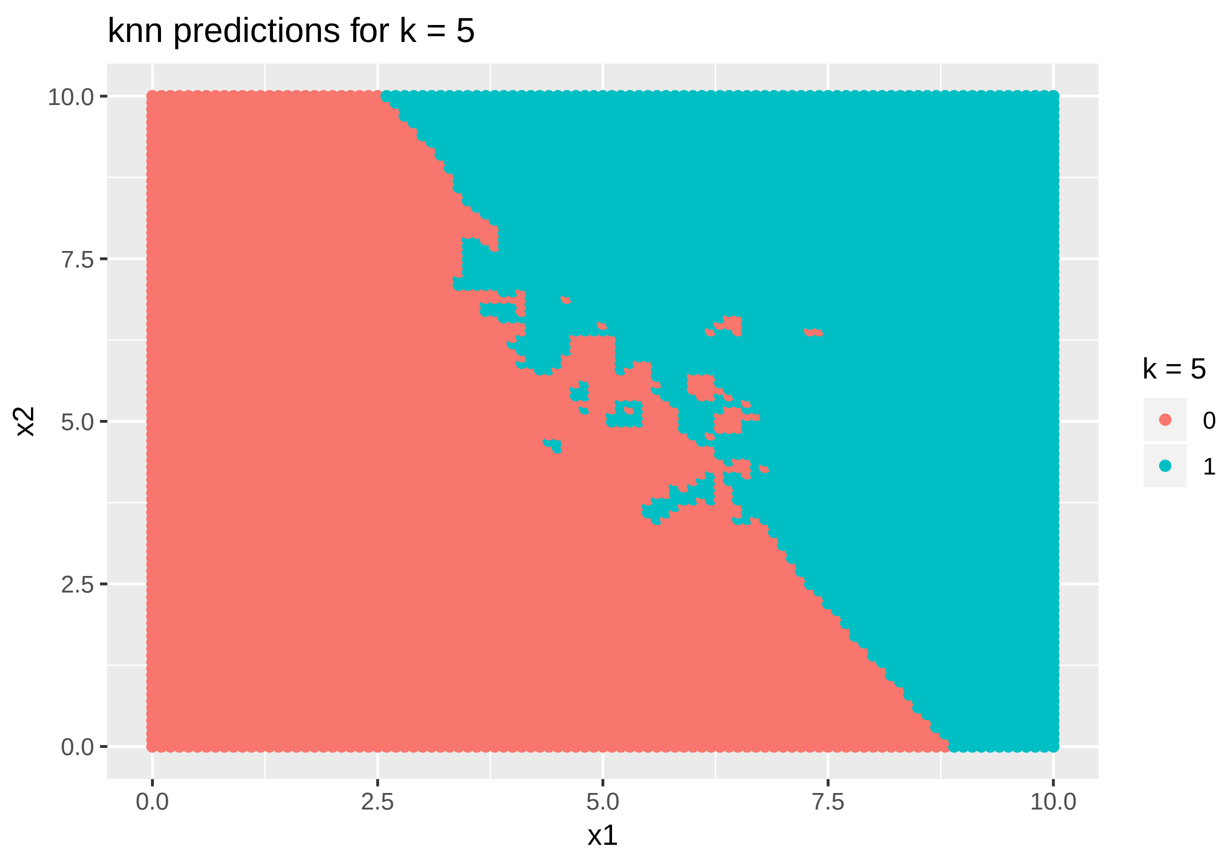
```
## Observations: 400
## Variables: 3
## $ x1 <dbl> 7.708587, 5.397889, 5.617803, 3.264038, 5.104902, 5.715045,...
## $ x2 <dbl> 5.671228, 5.760676, 6.416842, 3.950046, 4.408397, 5.683394,...
## $ y  <fct> 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0,...
```

3, 4, 5).

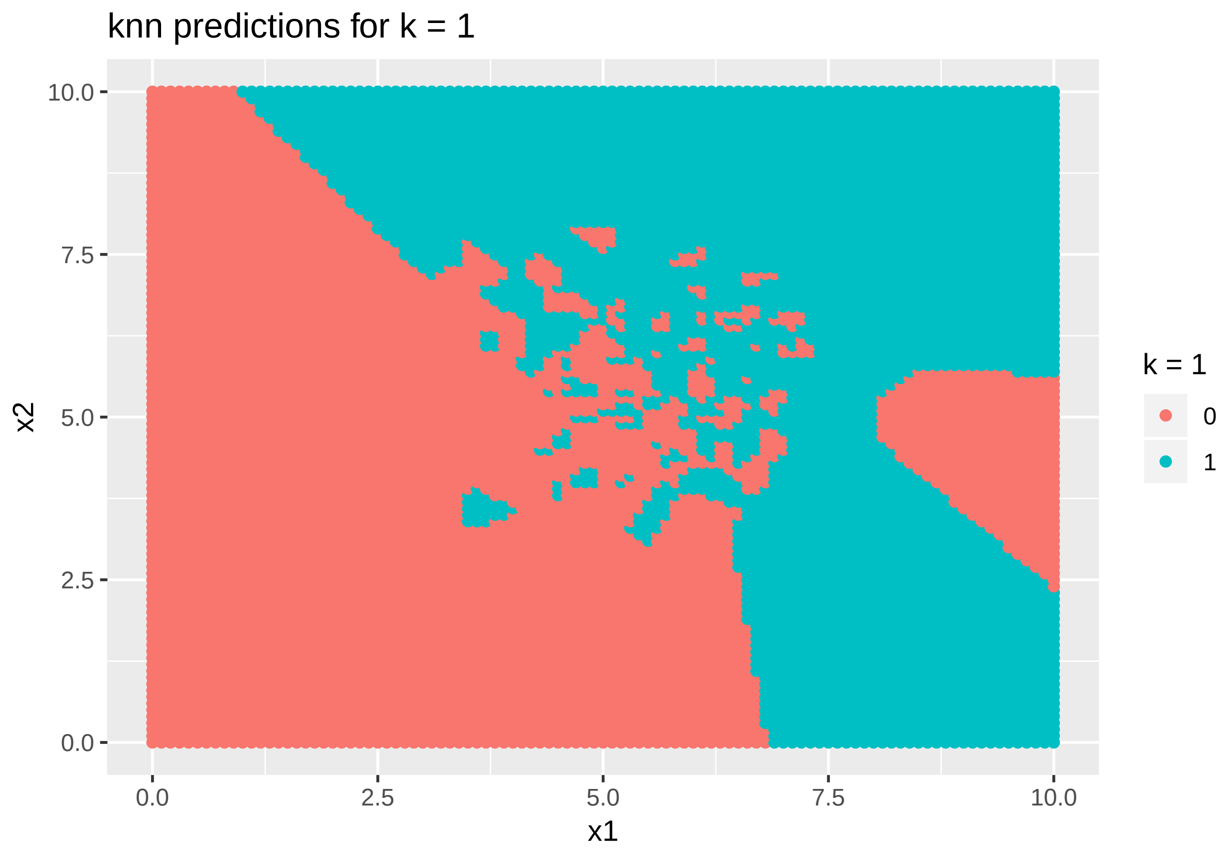
```
# split training data and labels
trn <- train_norm %>% select(-y)
trn_labels <- factor(train_norm$y)
kvs <- c(25,5,1)

# re-run the modeling step on each value of k and save the predictions to a list
mods <- list()
for(i in kvs){
  model_predictions <- knn(train = trn, test = grid_null, cl = trn_labels, k = i)
  mods[[i]] <- model_predictions
}

# plot the colors in the color grid using each set of model predictions
p <- ggplot(grid_null) + geom_point(aes(x = x1, y=x2))
p + aes( color = mods[[25]] ) + labs(title = 'knn predictions for k = 25', color = 'k = 25')
```



```
p + aes( color = mods[[5]] ) + labs(title = 'knn predictions for k = 5',color = 'k = 5')
```



```r
p + aes( color = mods[[1]] ) + labs(title = 'knn predictions for k = 1',color = 'k = 1')
```



6).

As the values of k changed from 25 to 5 to 1 in the knn analysis, the influence of only the closest observations from the training data increased, causing the values close to the boundary to be increasingly influenced by the random mixing of the two labeled distributions in the data. At higher values of k, a greater portion of the distribution was accounted for in labeling each observation and so more information capturing the overall form of the distribution was integrated into the decision making process.

7).

It seems that the classifications for k = 25 more closely matched the boundary we would expect given the distributions used to create the training data. While k = 5 produced similar classifications at the extremes, the classifications closer to the boundary were clearly influenced more by noise in the distributions, as there were small disconnected pockets embedded across the boundary between distributions. At n = 1, values at the extremes also started looking random, as outlying examples from the training set exhibited hi influence over classifications in regions where there were few examples to be near.