# IntroCV Assignment 4-2D Classification

```r
library(tidyverse)
```

```
## -- Attaching packages --------
```

```
## v ggplot2 3.3.0     v purrr   0.3.3
## v tibble  2.1.3     v dplyr   0.8.5
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts -----------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
library(class) ## for knn
```

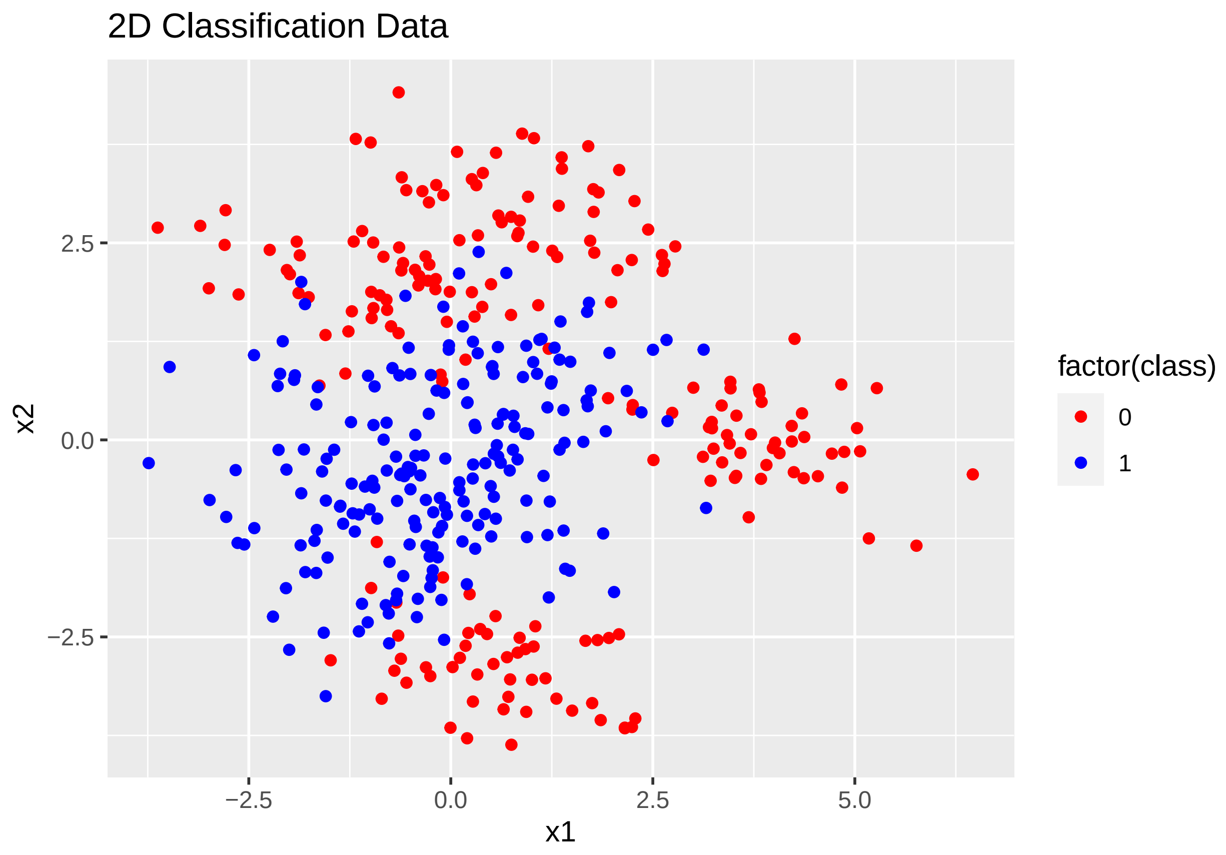
## Assignment 4: Classification 2

Consider the data set in "ClassificationData2D.csv", use 10-fold cross-validation to determine the best possible KNN predictive model.

```r
data2D.df <- read_csv( "ClassificationData2D.csv")
```

```
## Parsed with column specification:
## cols(
##   x1 = col_double(),
##   x2 = col_double(),
##   class = col_double()
## )
```

```r
data2D.df %>%
  ggplot()+
  geom_point(aes(x1,x2,color=factor(class)))+
  scale_color_manual(values=c("red","blue"))+
  labs(title="2D Classification Data")
```



How does this result compare with Logistic regression, LDA, and QDA. In each case, just use the full predictor set. Use 10-fold cross-validation to estimate the error rate.

## Solution: KNN

As always, start with a practice calculation

```
N <- nrow(data2D.df)
p <- ncol(data2D.df)
names(data2D.df)
```

```
## [1] "x1"    "x2"    "class"
```

Practice run. . . .

```
kVal <- 10
numFolds <- 10
folds <- sample(1:numFolds,N,rep=T)
errs <- numeric(numFolds)
for(fold in 1:numFolds){

  train.x <- data.matrix(data2D.df[folds != fold, 1:2])
  test.x <- data.matrix(data2D.df[folds == fold, 1:2])
  train.y <- data.matrix(data2D.df[folds != fold, 3])
  test.y <- data.matrix(data2D.df[folds == fold, 3])
```

```r
  mod.knn <- knn(train.x,test.x,train.y,k=kVal)
  errs[fold] <- mean((test.y != mod.knn))
}
mean(errs)
```

```
## [1] 0.09425182
```

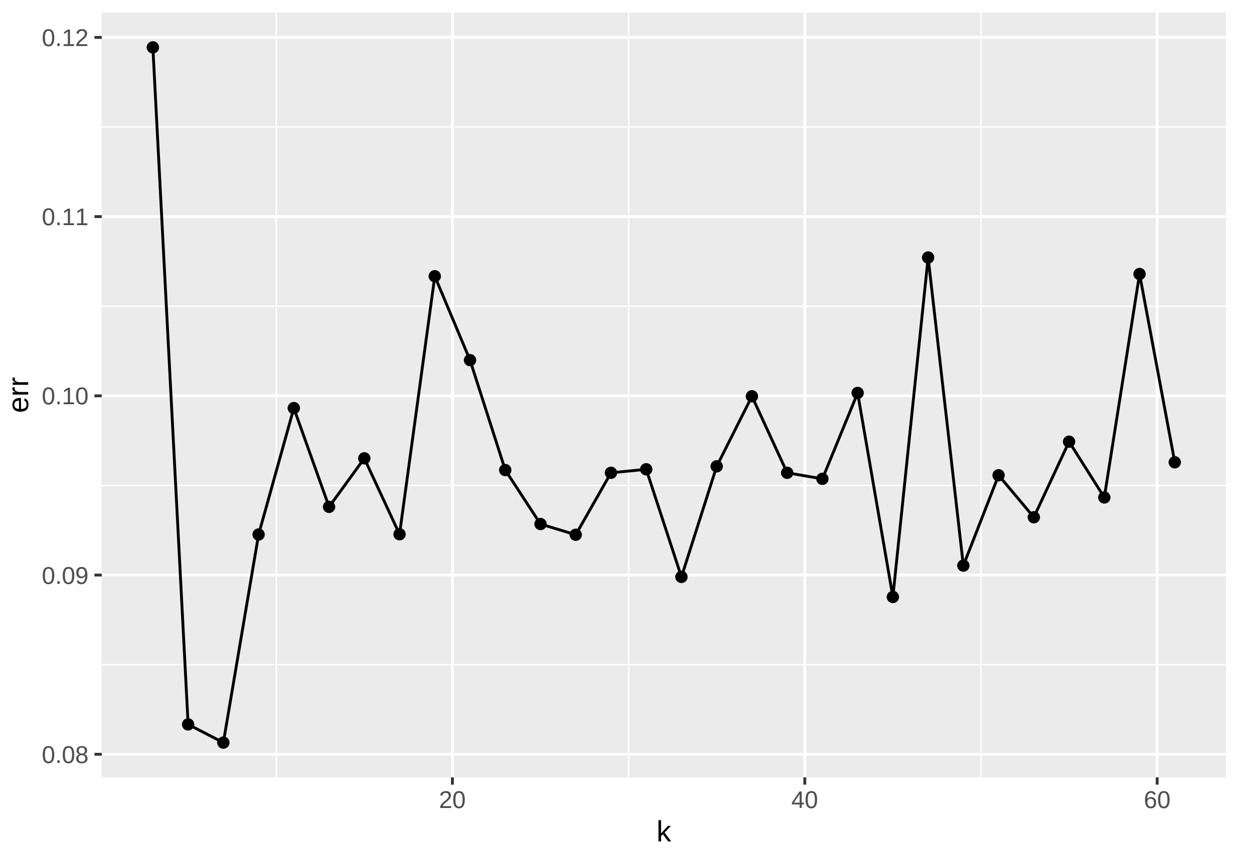
Make a function

```r
errCV.knn <- function(kVal){
  folds <- sample(1:numFolds,N,rep=T)
  errs <- numeric(numFolds)
  for(fold in 1:numFolds){

    train.x <- data.matrix(data2D.df[folds != fold, 1:2])
    test.x <- data.matrix(data2D.df[folds == fold, 1:2])
    train.y <- data.matrix(data2D.df[folds != fold, 3])
    test.y <- data.matrix(data2D.df[folds == fold, 3])
    mod.knn <- knn(train.x,test.x,train.y,k=kVal)
    errs[fold] <- mean((test.y != mod.knn))
  }
  mean(errs)
}
##test it out
errCV.knn(117)
```

```
## [1] 0.2319767
```

Run through a range of kVals. Might want to run this several times to get a feel for the behavior

```r
maxK <- 30
kVals <- 2*(1:maxK)+1
errs <- map_dbl(kVals,errCV.knn)
data.frame(k=kVals,err=errs) %>%
  ggplot()+
  geom_point(aes(k,err))+
  geom_line(aes(k,err))
```



Looking at the results, a value of k around 18-20 seems appropriate.

```
kOpt <- 20
(err.knn <- errCV.knn(kOpt))
```

```
## [1] 0.09883088
```

An error rate of about .10.

## Solution: LDA/QDA

```
library(MASS) ## for lda
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```r
  folds <- sample(1:numFolds,N,rep=T)
  errs <- numeric(numFolds)
  for(fold in 1:numFolds){
    train.df <- data2D.df[folds != fold,]
    test.df <- data2D.df[folds != fold, ]
    mod.lda <- lda(class ~ x1+x2,data=train.df)
    pred <- predict(mod.lda,newdata=test.df)
    pred$class
    errs[fold] <- with(test.df,mean((class!=pred$class)))
  }
  (err.lda <- mean(errs))
```

```
## [1] 0.3546617
```

How about QDA.

```r
 folds <- sample(1:numFolds,N,rep=T)
  errs <- numeric(numFolds)
  for(fold in 1:numFolds){
    train.df <- data2D.df[folds != fold,]
    test.df <- data2D.df[folds != fold, ]
    mod.qda <- qda(class ~ x1+x2,data=train.df)
    pred <- predict(mod.qda,newdata=test.df)
    pred$class
    errs[fold] <- with(test.df,mean((class!=pred$class)))
  }
  (err.qda <- mean(errs))
```

```
## [1] 0.1250786
```

```r
c(err.knn,err.lda,err.qda)
```

```
## [1] 0.09883088 0.35466169 0.12507865
```

KNN Wins.

## Visua, l

Just for fun, build a grid and include visualization of these models.

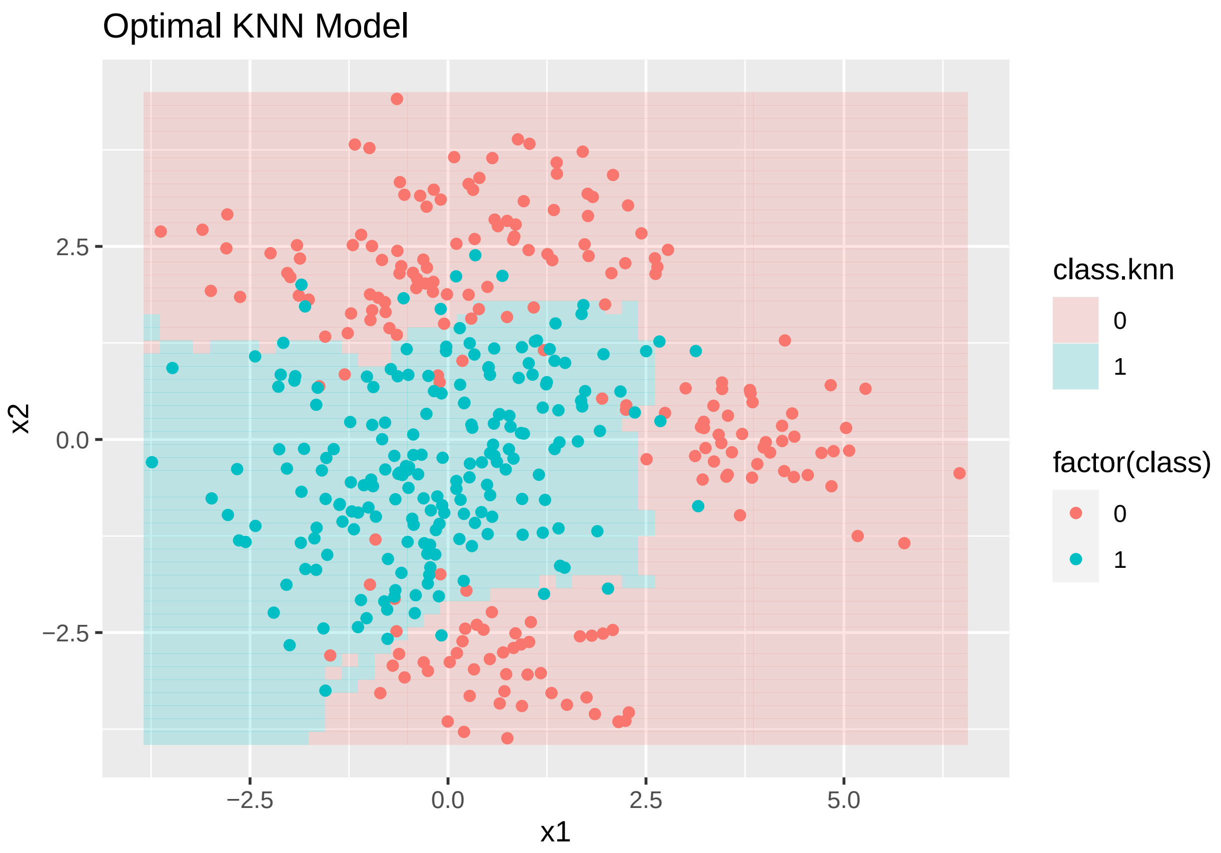
```r
rng1 <- with(data2D.df,range(x1))
rng2 <- with(data2D.df,range(x2))
gridSize <- 50
grid.df <- data.frame(expand.grid(x1=seq(rng1[1],rng1[2],length=gridSize),
                       x2=seq(rng2[1],rng2[2],length=gridSize)))
head(grid.df)
```

```
##          x1        x2
## 1 -3.738588 -3.868058
## 2 -3.530456 -3.868058
## 3 -3.322323 -3.868058
## 4 -3.114191 -3.868058
## 5 -2.906058 -3.868058
## 6 -2.697926 -3.868058
```

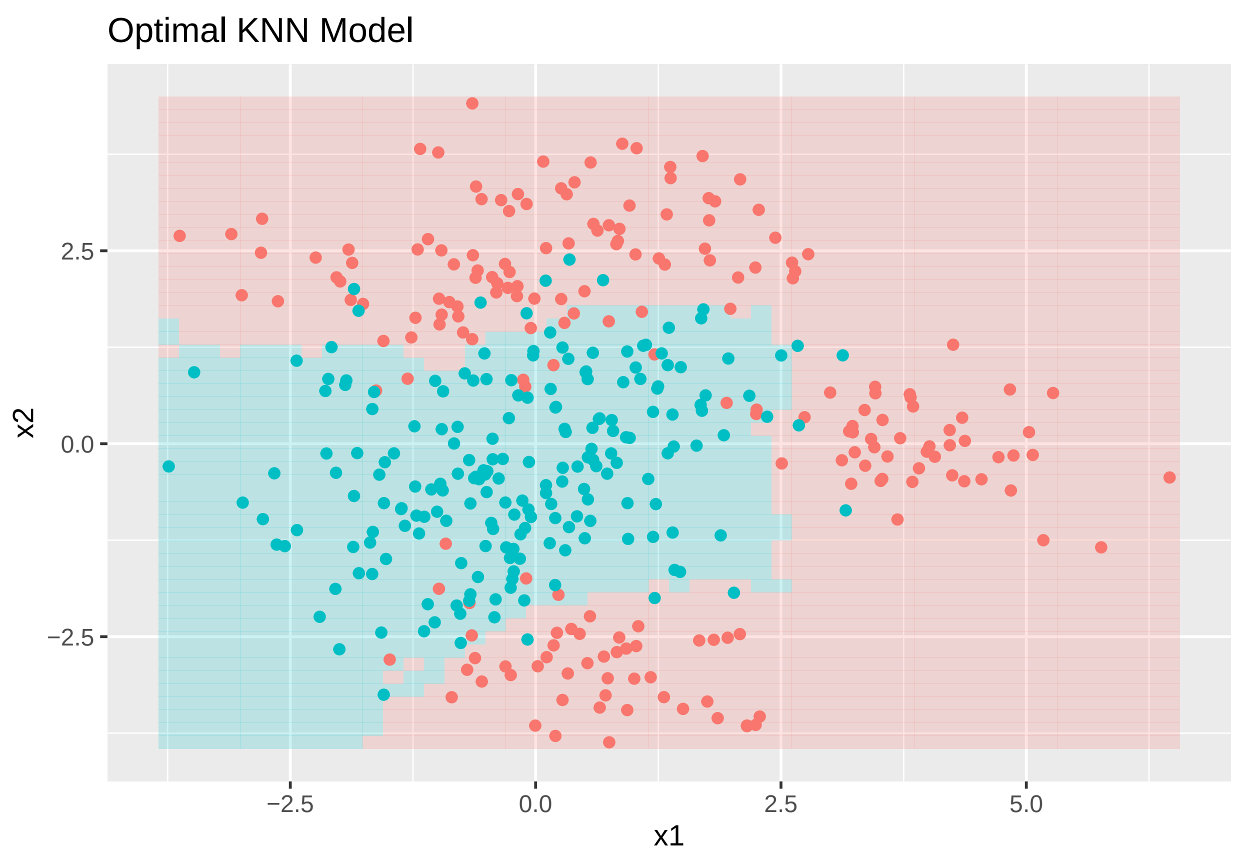
```
grid.xy <- data.matrix(grid.df[c("x1","x2")])
head(grid.xy)
```

```
##             x1        x2
## [1,] -3.738588 -3.868058
## [2,] -3.530456 -3.868058
## [3,] -3.322323 -3.868058
## [4,] -3.114191 -3.868058
## [5,] -2.906058 -3.868058
## [6,] -2.697926 -3.868058
```

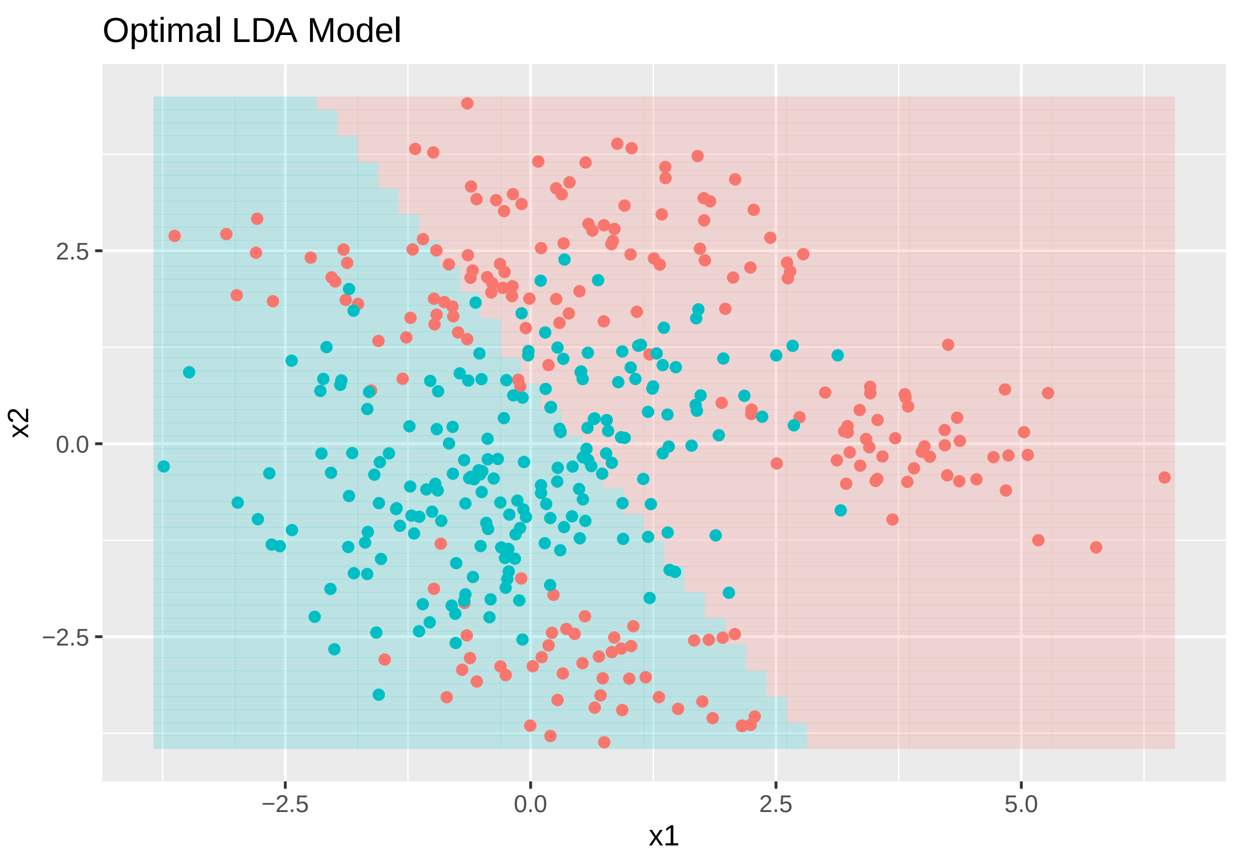
```
train.x <- data.matrix(data2D.df[, 1:2])
train.y <- data.matrix(data2D.df[, 3])


####
mod.knn <- knn(train.x,grid.xy,train.y,k=kOpt)
grid.df$class.knn <- mod.knn

knn.gg <- grid.df %>%
  ggplot()+
  geom_tile(aes(x1,x2,fill=class.knn),alpha=.2)+
  geom_point(data=data2D.df,
             aes(x1,x2,color=factor(class)))+
  labs(title="Optimal KNN Model")
knn.gg
```



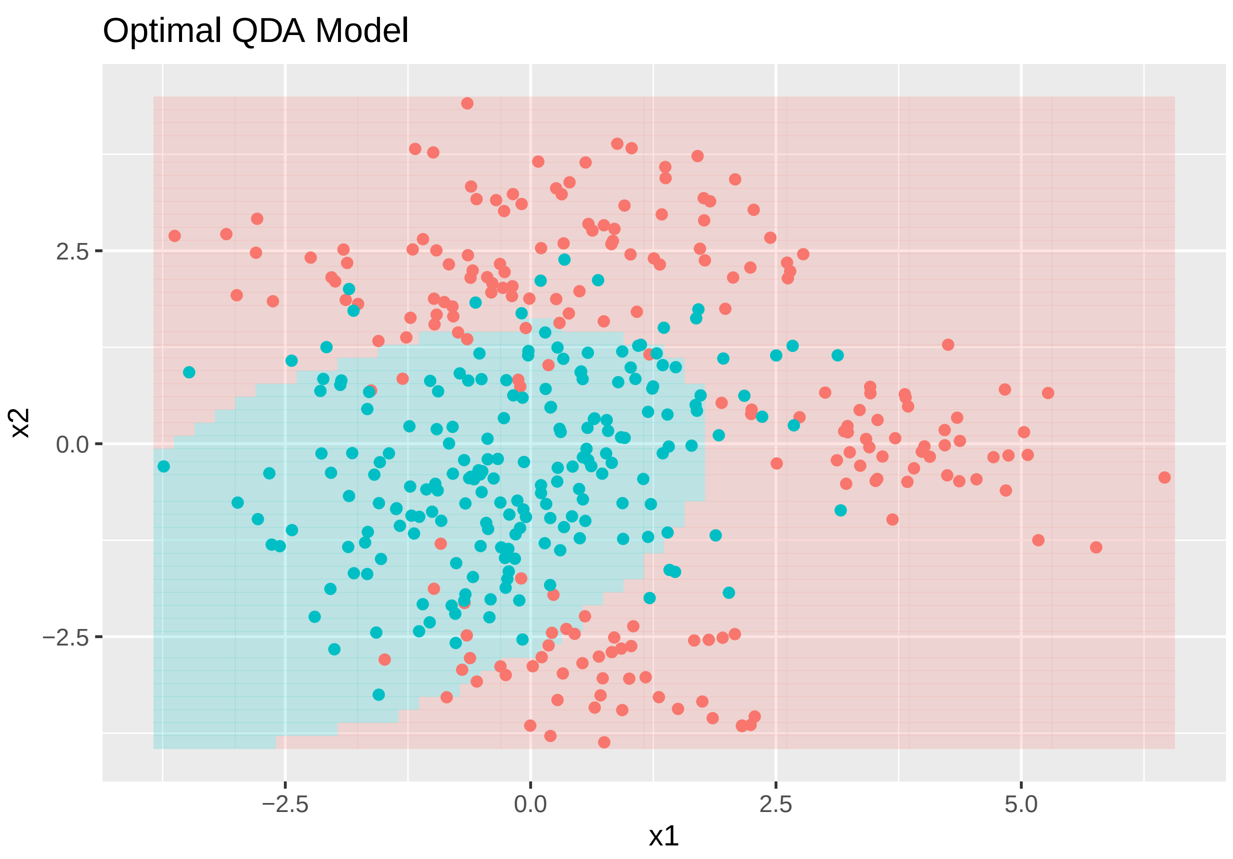
```
mod.lda <- lda(class ~  x1+x2,data=data2D.df)
mod.qda <- qda(class ~  x1+x2,data=data2D.df)
pred.lda <- predict(mod.lda,newdata=grid.df)
pred.qda <- predict(mod.qda,newdata=grid.df)
grid.df$class.lda <- pred.lda$class
grid.df$class.qda <- pred.qda$class

##
knn.gg <- grid.df %>%
  ggplot()+
  geom_tile(aes(x1,x2,fill=class.knn),alpha=.2)+
  geom_point(data=data2D.df,
             aes(x1,x2,color=factor(class)))+
  labs(title="Optimal KNN Model")+
  guides(color=FALSE,fill=FALSE)
knn.gg
```



```
lda.gg <- grid.df %>%
  ggplot()+
  geom_tile(aes(x1,x2,fill=class.lda),alpha=.2)+
  geom_point(data=data2D.df,
             aes(x1,x2,color=factor(class)))+
  labs(title="Optimal LDA Model")+
  guides(color=FALSE,fill=FALSE)
lda.gg
```



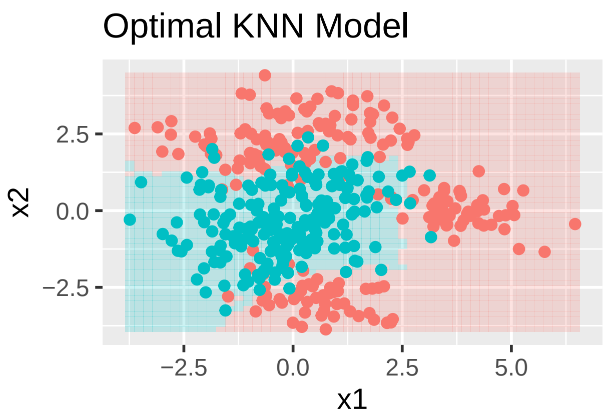
```r
qda.gg <- grid.df %>%
  ggplot()+
  geom_tile(aes(x1,x2,fill=class.qda),alpha=.2)+
  geom_point(data=data2D.df,
             aes(x1,x2,color=factor(class)))+
  labs(title="Optimal QDA Model")+
  guides(color=FALSE,fill=FALSE)
qda.gg
```

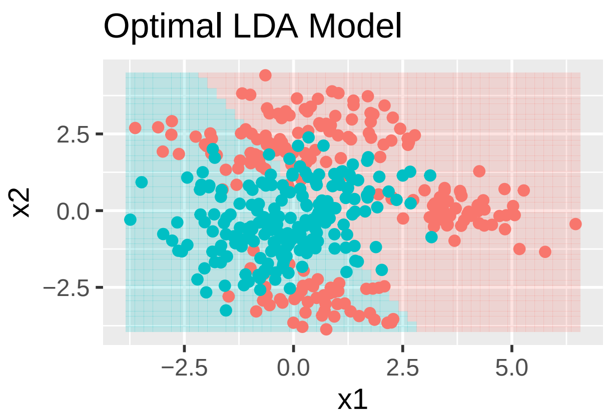


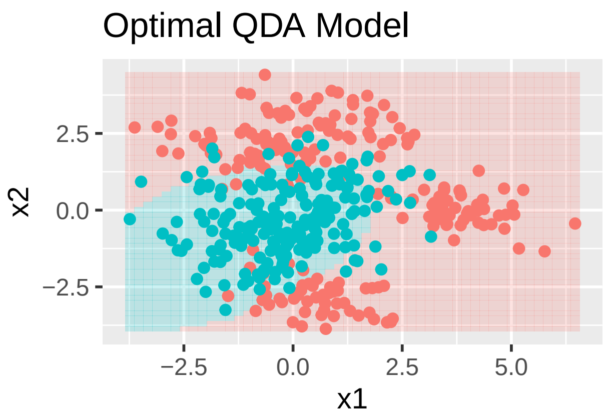
```r
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
grid.arrange(knn.gg,lda.gg,qda.gg,nrow=2)
```


Optimal KNN Model
x2
2.5
0.0
−2.5
−2.5
0.0
2.5
5.0
x1


Optimal LDA Model
x2
2.5
0.0
−2.5
−2.5
0.0
2.5
5.0
x1


Optimal QDA Model
x2
2.5
0.0
−2.5
−2.5
0.0
2.5
5.0
x1