Review

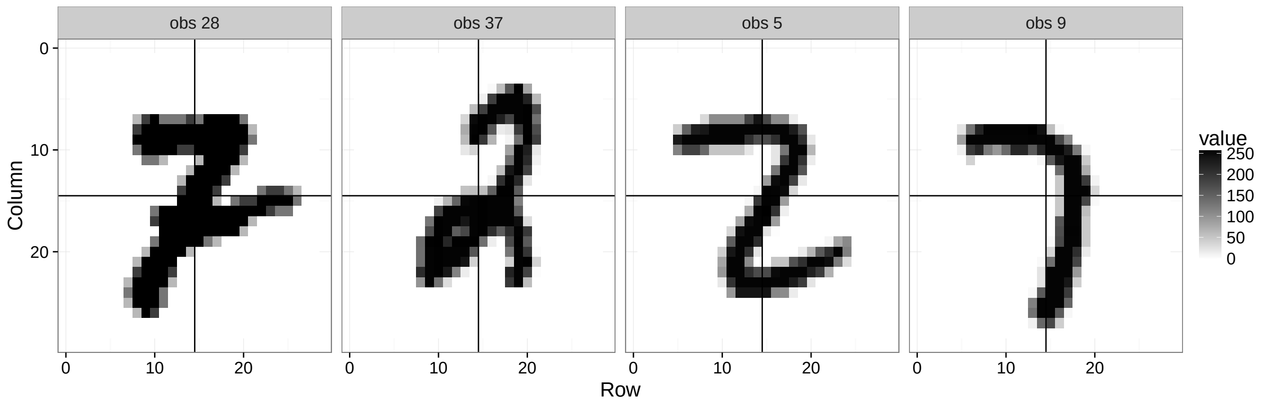
Let's continue with the digits data. We read-in the data:

```
url <- "https://raw.githubusercontent.com/datasciencelabs/data/master/hand-written-digits-train.csv"
if(!exists("digits")) digits <- read_csv(url)
```

To simplify the problem we will try to distinguish 2s from 7s. So we subset to only those digits

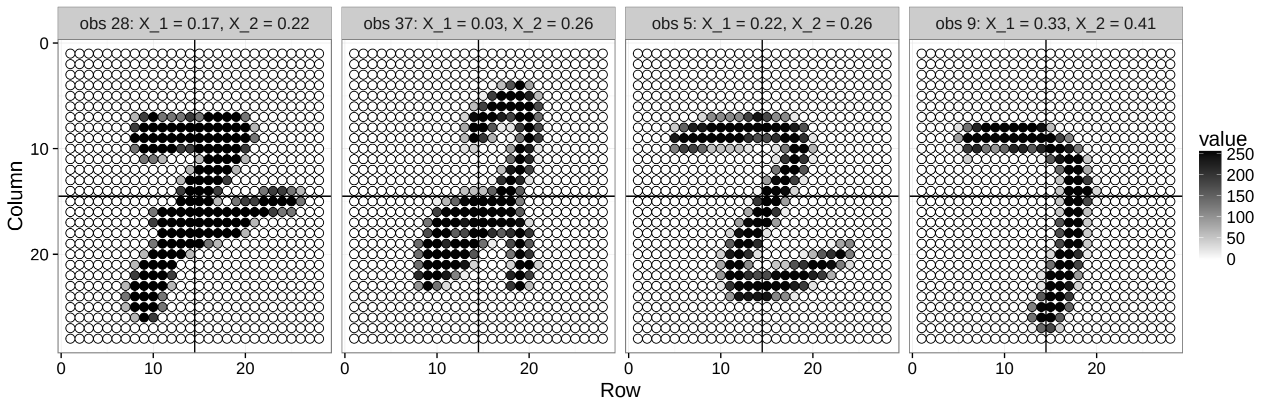
```
dat <- digits %>% filter(label%in%c(2,7))
```

For illustrative purposes we created two features: `X_1` is the percent of black pixels that are in the top left quadrant and `X_2` is the percent of black pixels that are in the bottom left quadrant:



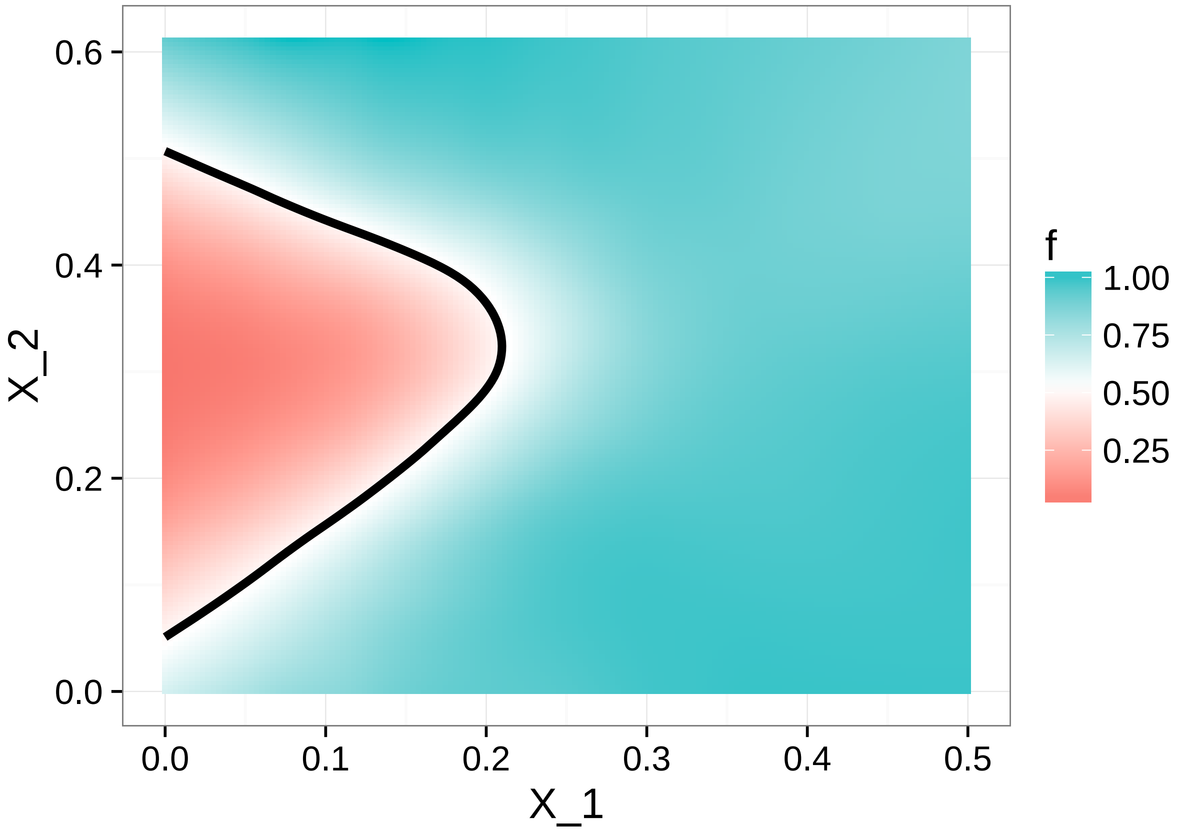
We can create these new predictors like this:

We can see some example of what these predictors are:



We act as if we know the truth:

```
## Loading required package: lattice
```



Quadratic and Linear Discriminant Analysis

For illustration purposes let's take a subset:

```
##create the training set with 1000 variabes
set.seed(1)
dat <- sample_n(dat, 1000) %>% select(y, X_1, X_2)
dat
```

```
## Source: local data frame [1,000 x 3]
##
##          y        X_1       X_2
##      (dbl)      (dbl)     (dbl)
## 1        1 0.28301887 0.2452830
## 2        0 0.07638889 0.2222222
## 3        0 0.36923077 0.3538462
## 4        1 0.26530612 0.3061224
## 5        0 0.08333333 0.3055556
## 6        0 0.23364486 0.2897196
## 7        0 0.11403509 0.2456140
## 8        1 0.12162162 0.2702703
## 9        0 0.20833333 0.3194444
## 10       0 0.11842105 0.3684211
## ..     ...        ...       ...
```

Now create train and test sets:

```r
library(caret)
inTrain <- createDataPartition(y = dat$y, p=0.5)
train_set <- slice(dat, inTrain$Resample1)
test_set <- slice(dat, -inTrain$Resample1)
```

Quadratic Discriminant Analysis (QDA) relates to the *Naive Bayes* approach we described earlier. We try to estimate $\Pr(Y = 1|X = x)$ using Bayes theorem.

$$f(x) = \Pr(Y = 1|\mathbf{X} = \mathbf{x}) = \frac{\pi p_{\mathbf{X}|Y=1}(\mathbf{x})}{(1-\pi)p_{\mathbf{X}|Y=0}(x) + \pi p_{\mathbf{X}|Y=1}(x)}$$

With QDA we assume that the distributions $p_{\mathbf{X}|Y=1}(\mathbf{x})$ and $p_{\mathbf{X}|Y=0}(\mathbf{x})$ are multivariate normal. In our case we have two predictors so we assume each one is bivariate normal. This implies we need to estimate two averages, two standard deviations, and a correlation for each case $Y = 1$ and $Y = 0$.

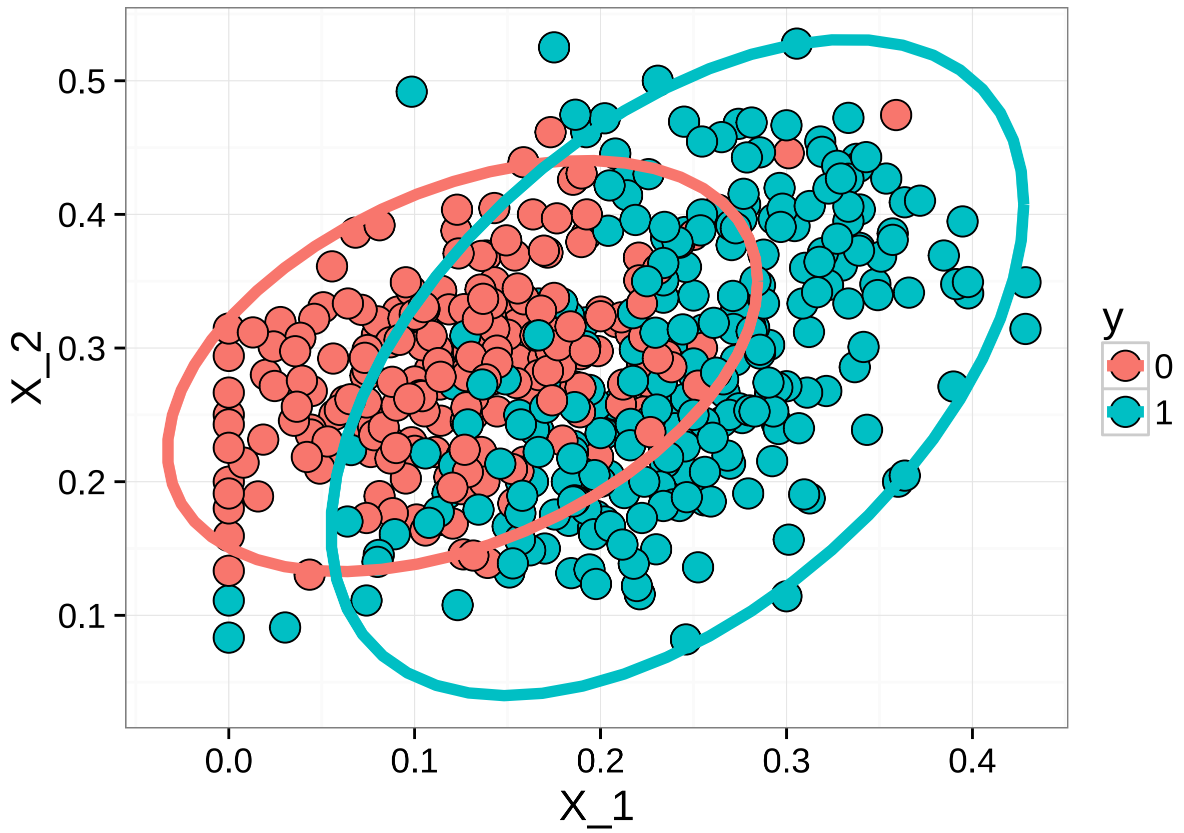
This implies that we can approximate the distributions $p_{X_1,X_2|Y=1}$ and $p_{X_1,X_2|Y=0}$. We can easily estimate parameters from the data:

```r
options(digits = 2)
params <- train_set %>% group_by(y) %>%
  summarize(avg_1 = mean(X_1), avg_2 = mean(X_2), sd_1= sd(X_1), sd_2 = sd(X_2), r = cor(X_1,X_2))
params
```

```
## Source: local data frame [2 x 6]
##
##       y avg_1 avg_2  sd_1  sd_2     r
##   (dbl) (dbl) (dbl) (dbl) (dbl) (dbl)
## 1     0  0.13  0.29 0.064 0.062  0.41
## 2     1  0.24  0.29 0.076 0.100  0.50
```

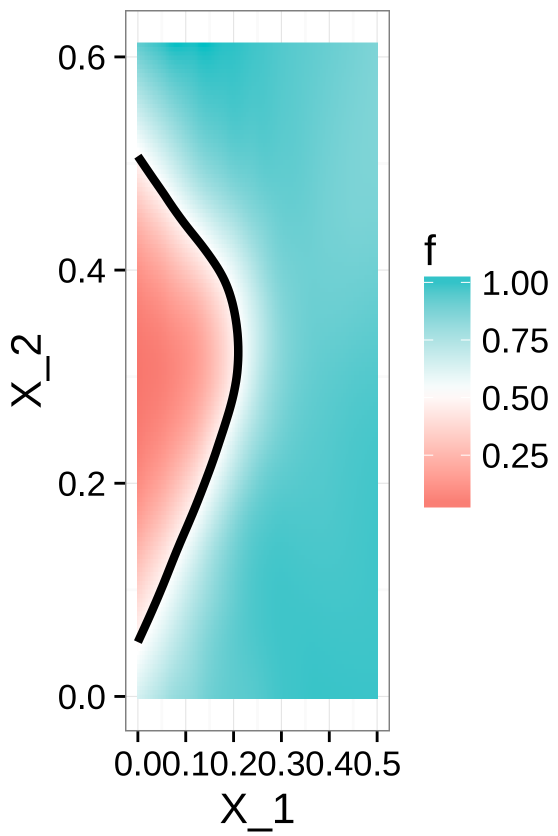
So here are the data and contour plots showing the two normal densities:

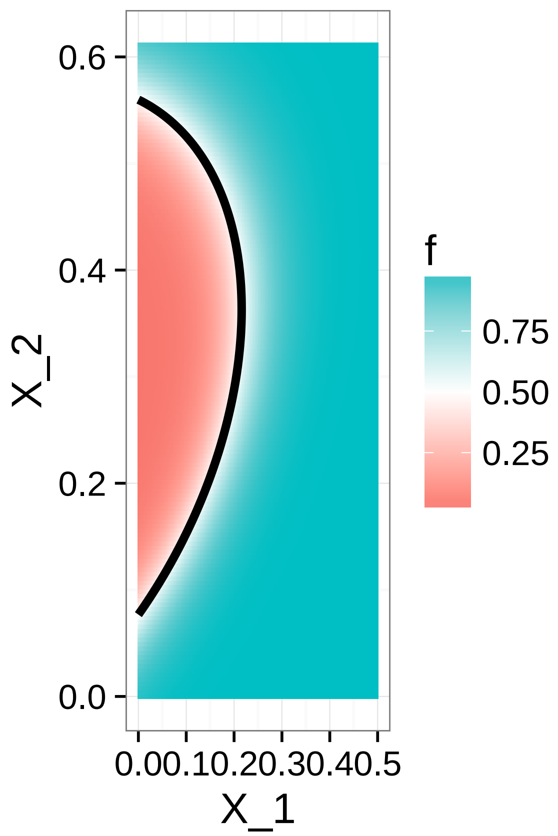
```r
train_set %>% mutate(y = factor(y)) %>%
  ggplot(aes(X_1, X_2, fill = y, color=y)) +
  geom_point(pch=21,cex=5, color="black") +
  stat_ellipse(lwd=2, type="norm")
```



This defines the following estimate of $f(x_1, x_2)$

```
library(mvtnorm)

get_p <- function(params, data){
  dmvnorm( cbind(data$X_1, data$X_2),
               mean = c(params$avg_1, params$avg_2),
               sigma = matrix( c(params$sd_1^2,
                                 params$sd_1*params$sd_2*params$r,
                                 params$sd_1*params$sd_2*params$r,
                                 params$sd_2^2),2,2))
}
pi <- 0.5
p0 <- get_p(params[1,], true_f)
p1 <- get_p(params[2,], true_f)

f_hat_qda <- pi*p1/(pi*p1 + (1-pi)*p0)

p <-true_f %>% mutate(f=f_hat_qda) %>%
 ggplot(aes(X_1, X_2, fill=f))  +
  scale_fill_gradientn(colors=c("#F8766D","white","#00BFC4")) +
  geom_raster()  + #guides(fill=FALSE) +
  stat_contour(aes(x=X_1,y=X_2,z=f),
               data=mutate(true_f, f=f_hat_qda),
               breaks=c(0.5),color="black",lwd=1.5)

grid.arrange(true_f_plot, p, nrow=1)
```





Here we have 2 predictors and had to compute 4 means, 4 SDs and 2 correlations. How many parameters would we have if instead of 2 predictors we had 10?

The main problems comes from estimating correlations for 10 of predictors. With 10, we have 45 correlations for each class. In general the formula is $p(p-1)/2$ which gets big fast.

A relatively solution to this is to assume that the correlation structure is the same for all classes. Which reduces the number of parameters we need to estimate. When we do this, we can show mathematical that the solution is "linear", in the linear algebra sense and we call it Linear Discriminant Analysis (LDA).

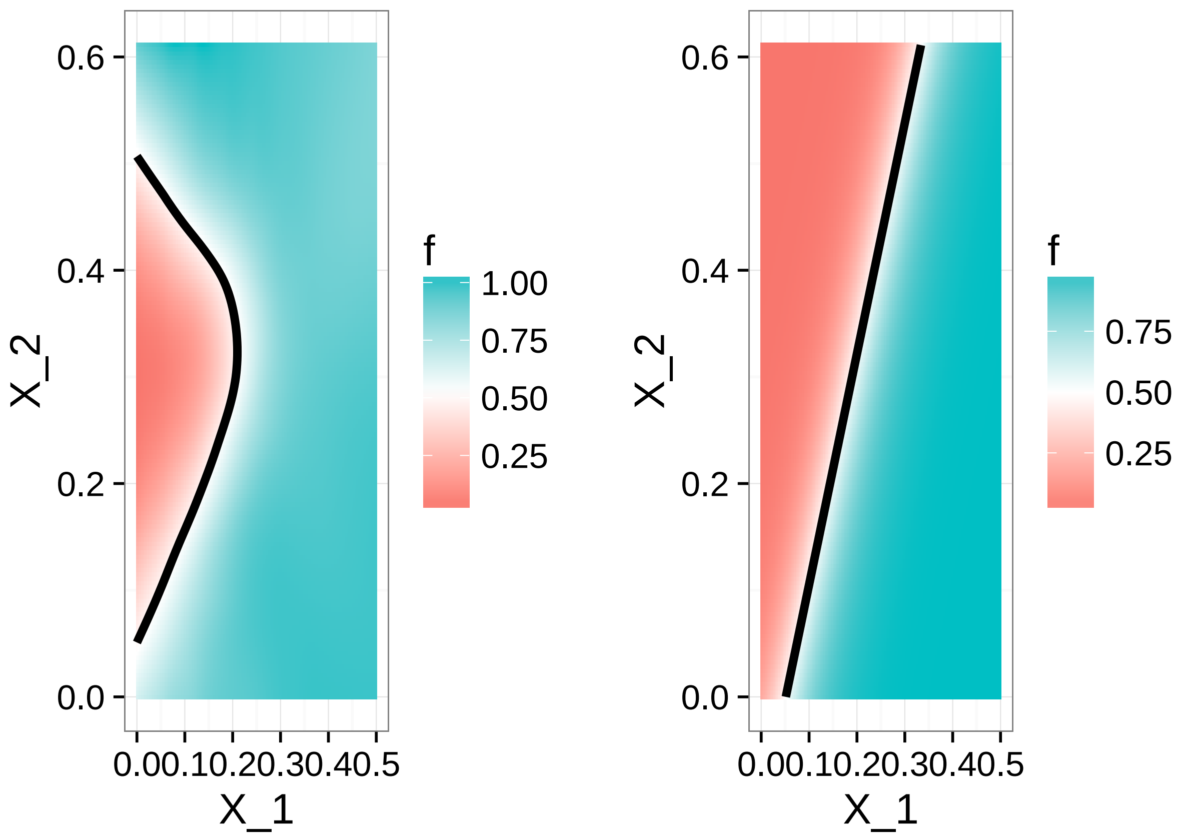
```
params <- train_set %>% group_by(y) %>%
  summarize(avg_1 = mean(X_1), avg_2 = mean(X_2), sd_1= sd(X_1), sd_2 = sd(X_2), r = cor(X_1,X_2))
params <-params %>% mutate(sd_1 = mean(sd_1), sd_2=mean(sd_1), r=mean(r))
params
```

```
## Source: local data frame [2 x 6]
##
##       y avg_1 avg_2  sd_1  sd_2     r
##   (dbl) (dbl) (dbl) (dbl) (dbl) (dbl)
## 1     0  0.13  0.29  0.07  0.07  0.45
## 2     1  0.24  0.29  0.07  0.07  0.45
```

This defines the following estimate of $f(x_1, x_2)$ and the boundary becomes linear:

```
library(mvtnorm)
p0 <- get_p(params[1,], data=true_f)
p1 <- get_p(params[2,], data=true_f)
p <- 0.5
```

```
f_hat_lda <- pi*p1/(pi*p1 + (1-pi)*p0)

p <- true_f %>% mutate(f=f_hat_lda) %>%
 ggplot(aes(X_1, X_2, fill=f))  +
  scale_fill_gradientn(colors=c("#F8766D","white","#00BFC4")) +
  geom_raster()  + #guides(fill=FALSE) +
  stat_contour(aes(x=X_1,y=X_2,z=f),
               data=mutate(true_f, f=f_hat_lda),
               breaks=c(0.5),color="black",lwd=1.5)

grid.arrange(true_f_plot, p, nrow=1)
```



Connection distance

The normal density is

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{\sigma^2}\right\}$$

Note if we remove the constant $1/(\sqrt{2\pi}\sigma)$ and then take the log we get:

$$-\frac{(x-\mu)^2}{\sigma^2}$$

which is the negative of a distance squared scaled by the standard deviation. For higher dimensions the same is true expect the scaling is more complex and involved correlations.

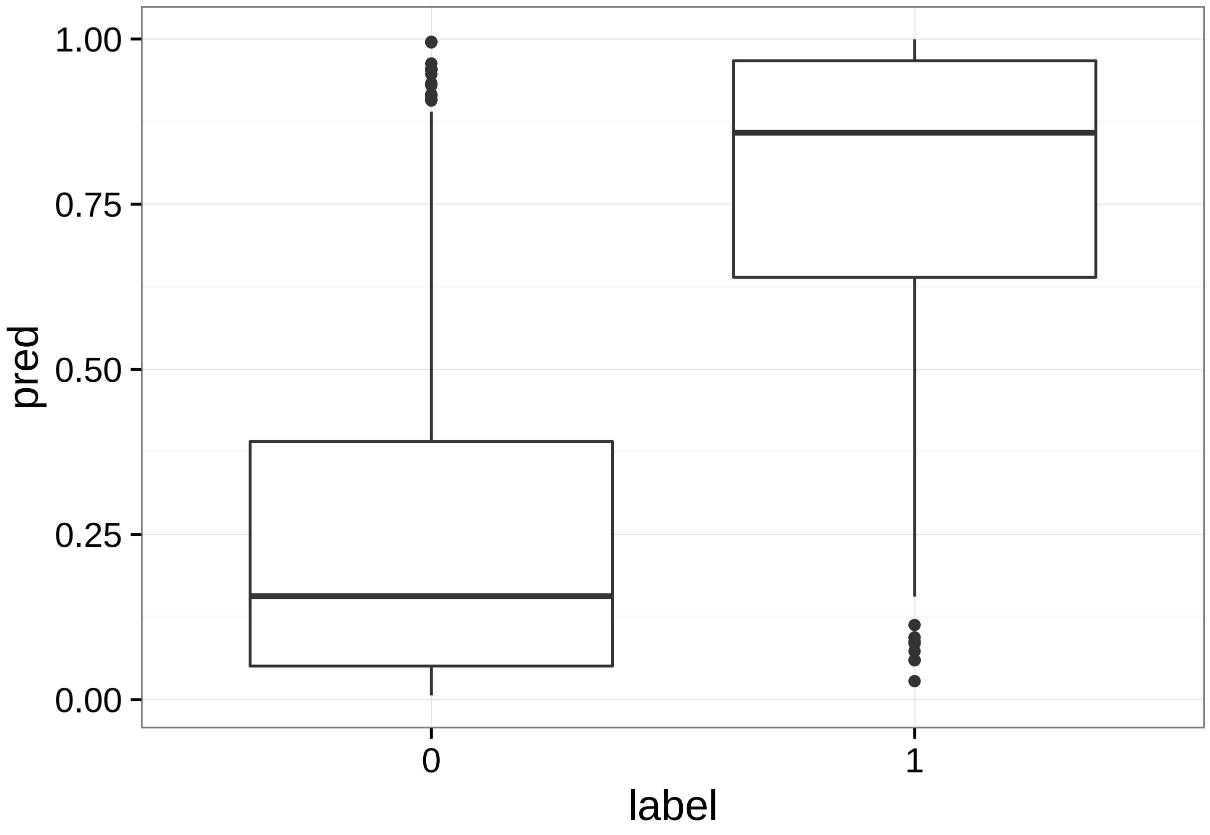
ROC

With the example we have been examining we can make two types of errors: calling a 2 a 7 or calling a 7 a 2. More generally, we binary data we call these false positives (calling a 0 a 1) and false negatives (calling a 1 a 0). Here we have arbitrarily made 7s 1s and 2s 0s.

This concept is important in many areas and in particular in health where one type of mistake can be much more costly than another. Note that we have been predicting 1s based on the rule $\hat{f}(x_1, x_2) > 0.5$ but we pick another cutoff, depending on how the cost of errors. For example, if we are predicting if a plane will malfunction, then we want a very low false negative rate and are willing to sacrifice our true positive rate.

We can see that the estimated probabilities are in a continuum:

```r
params <- train_set %>% group_by(y) %>%
  summarize(avg_1 = mean(X_1), avg_2 = mean(X_2),
            sd_1= sd(X_1), sd_2 = sd(X_2),
            r = cor(X_1,X_2))
p0 <- get_p(params[1,], dat= test_set)
p1 <- get_p(params[2,], dat= test_set)
pi <- 0.5
pred_qda <- pi*p1/(pi*p1 + (1-pi)*p0)
test_set %>% mutate(pred=pred_qda, label=as.factor(y)) %>%
  ggplot(aes(label,pred)) + geom_boxplot()
```

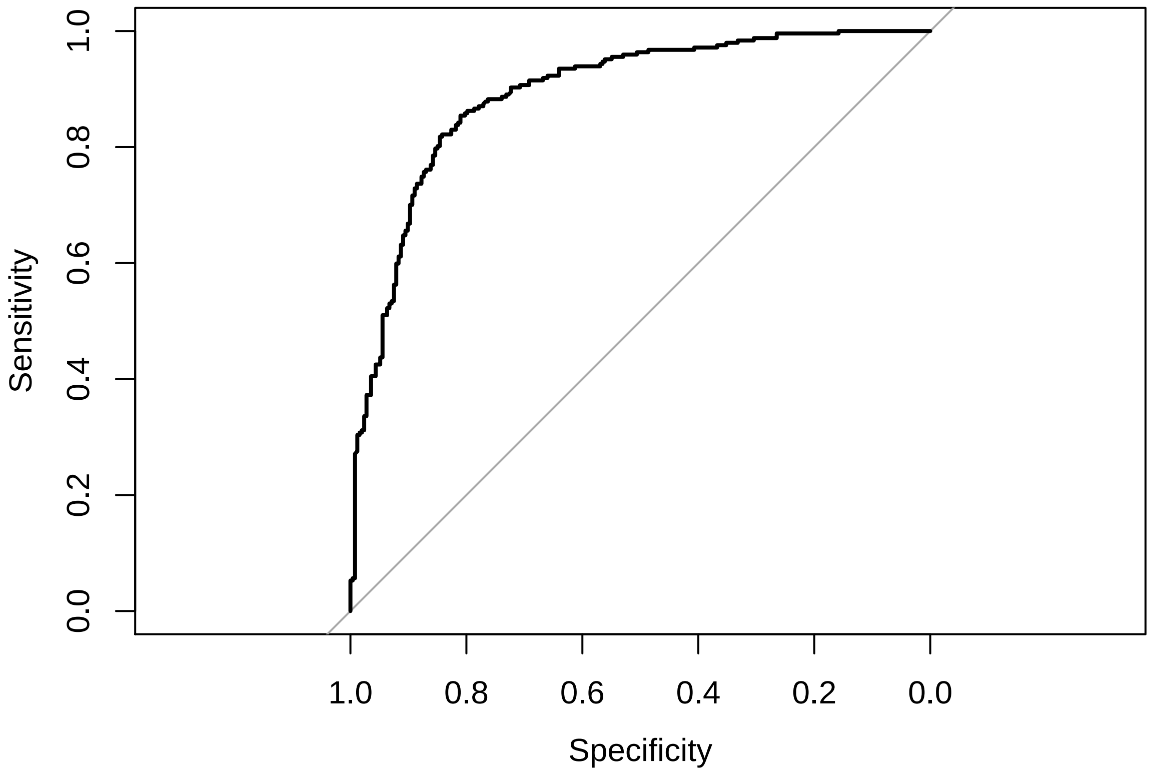


The Receiver Operator Characteristic Curve plots true positive rate versus false positive rates for several choices of cutoff. We can create this curve with the following code:

```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
roc_qda <- roc(test_set$y, pred_qda)
plot(roc_qda)
```



```
##
## Call:
## roc.default(response = test_set$y, predictor = pred_qda)
##
## Data: pred_qda in 253 controls (test_set$y 0) < 247 cases (test_set$y 1).
## Area under the curve: 0.89
```

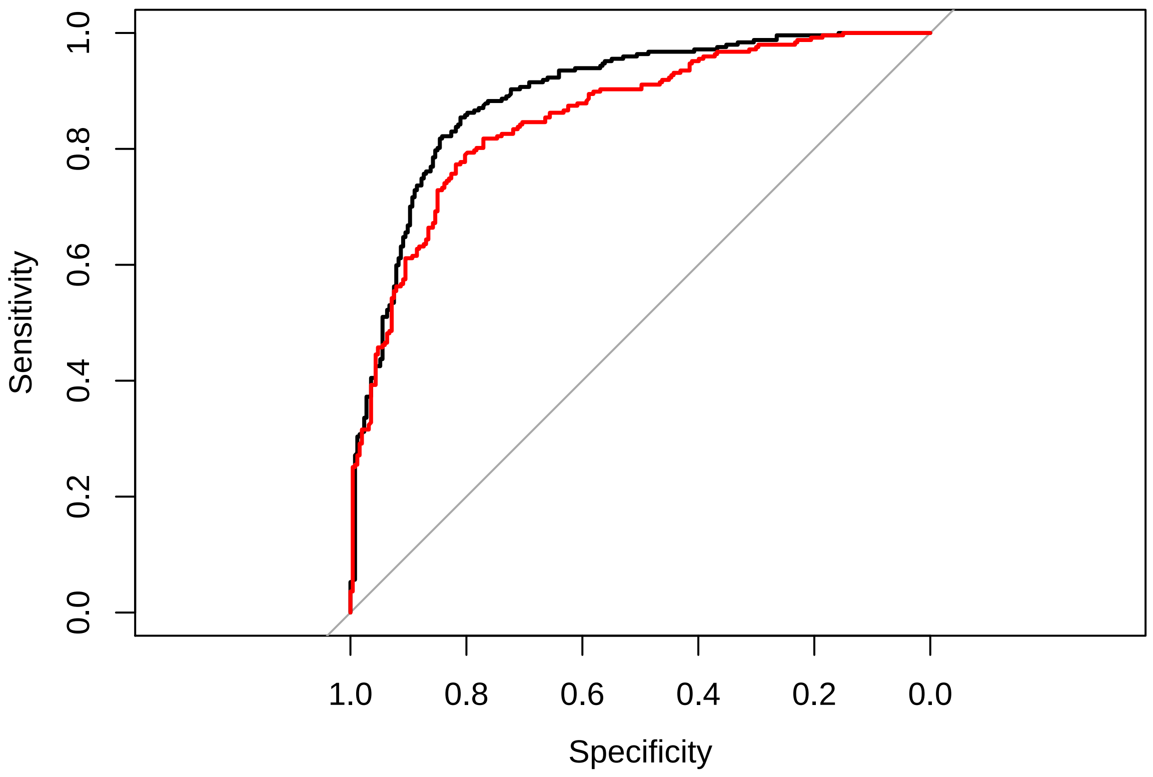
Here are the results for LDA

```r
params <-params %>% mutate(sd_1 = mean(sd_1),
                           sd_2 = mean(sd_1),
                           r=mean(r))
```

```
p0 <- get_p(params[1,], dat = test_set)
p1 <- get_p(params[2,], dat = test_set)
pi <- 0.5
pred_lda <- pi*p1/(pi*p1 + (1-pi)*p0)

roc_lda <- roc(test_set$y, pred_lda)
plot(roc_qda)
```

```
##
## Call:
## roc.default(response = test_set$y, predictor = pred_qda)
##
## Data: pred_qda in 253 controls (test_set$y 0) < 247 cases (test_set$y 1).
## Area under the curve: 0.89
```

```
plot(roc_lda, add=TRUE, col=2)
```



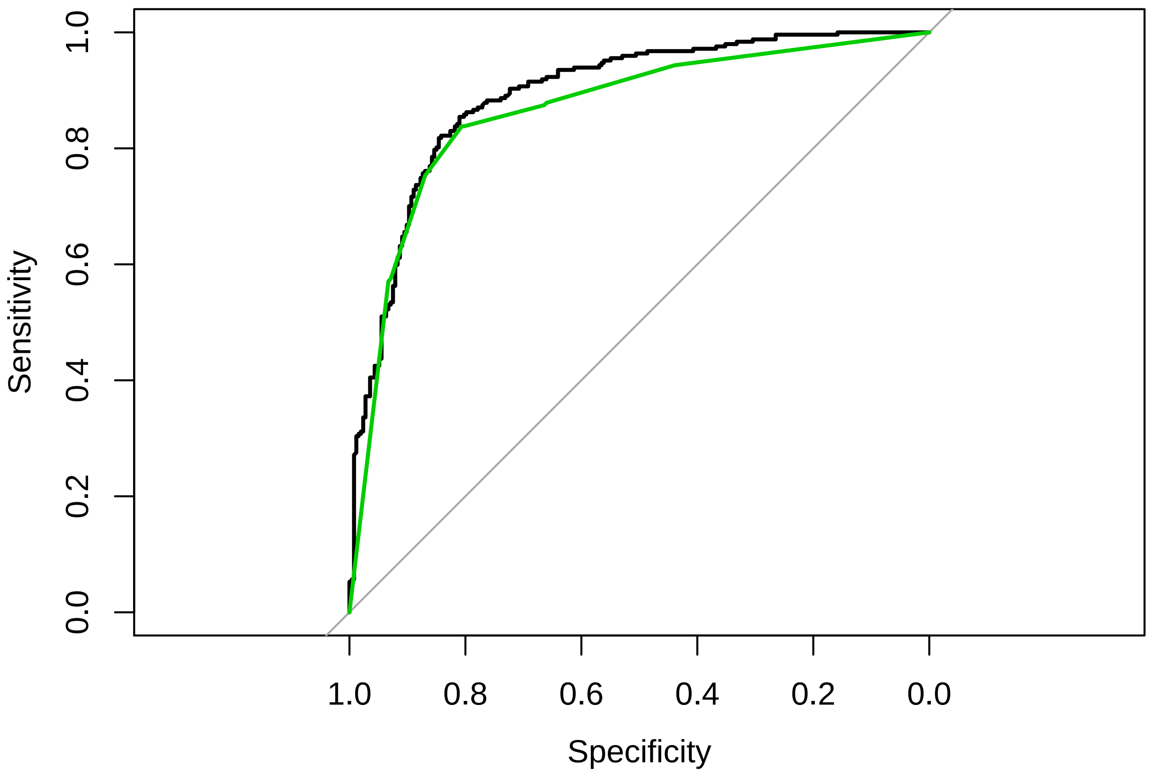
```
##
## Call:
## roc.default(response = test_set$y, predictor = pred_lda)
##
## Data: pred_lda in 253 controls (test_set$y 0) < 247 cases (test_set$y 1).
## Area under the curve: 0.86
```

We can also compare to kin

```r
fit <- knn3(y~., data = train_set, k=5)
pred_knn_5 <- predict(fit, newdata = test_set)[,2]
roc_knn_5 <- roc(test_set$y, pred_knn_5)
plot(roc_qda)
```

```
##
## Call:
## roc.default(response = test_set$y, predictor = pred_qda)
##
## Data: pred_qda in 253 controls (test_set$y 0) < 247 cases (test_set$y 1).
## Area under the curve: 0.89
```

```r
plot(roc_knn_5, add=TRUE, col=3)
```



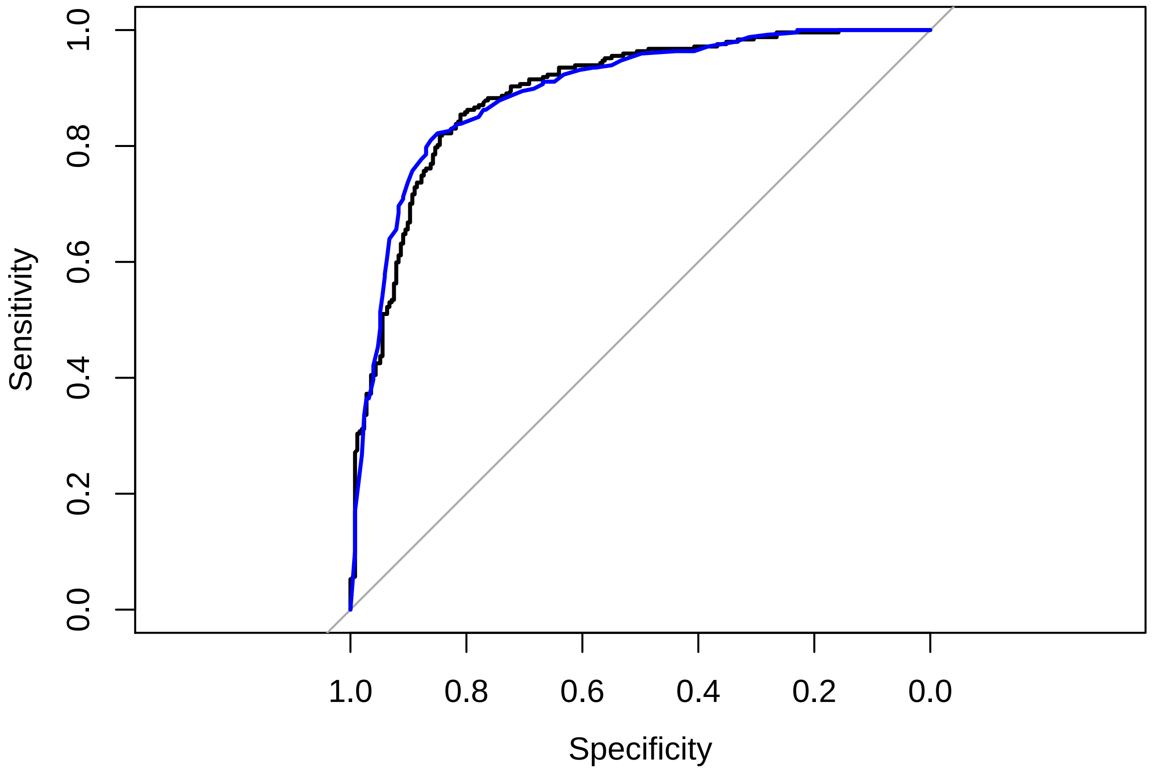
```
##
## Call:
## roc.default(response = test_set$y, predictor = pred_knn_5)
##
## Data: pred_knn_5 in 253 controls (test_set$y 0) < 247 cases (test_set$y 1).
## Area under the curve: 0.86
```

```r
fit <- knn3(y~., data = train_set, k=51)
pred_knn_51 <- predict(fit, newdata = test_set)[,2]
roc_knn_51 <- roc(test_set$y, pred_knn_51)
plot(roc_qda)
```

```
##
```

```
## Call:
## roc.default(response = test_set$y, predictor = pred_qda)
##
## Data: pred_qda in 253 controls (test_set$y 0) < 247 cases (test_set$y 1).
## Area under the curve: 0.89
```

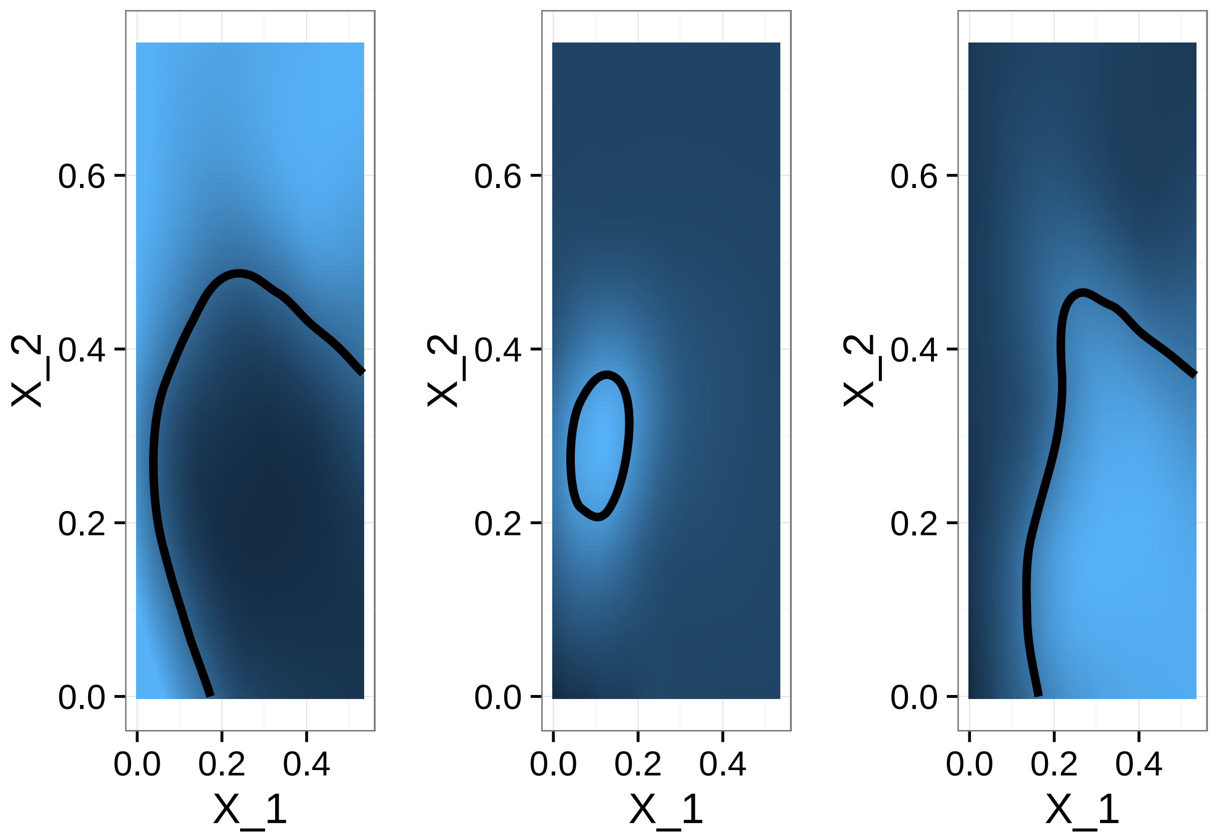
```
plot(roc_knn_51, add=TRUE, col=4)
```



```
##
## Call:
## roc.default(response = test_set$y, predictor = pred_knn_51)
##
## Data: pred_knn_51 in 253 controls (test_set$y 0) < 247 cases (test_set$y 1).
## Area under the curve: 0.89
```

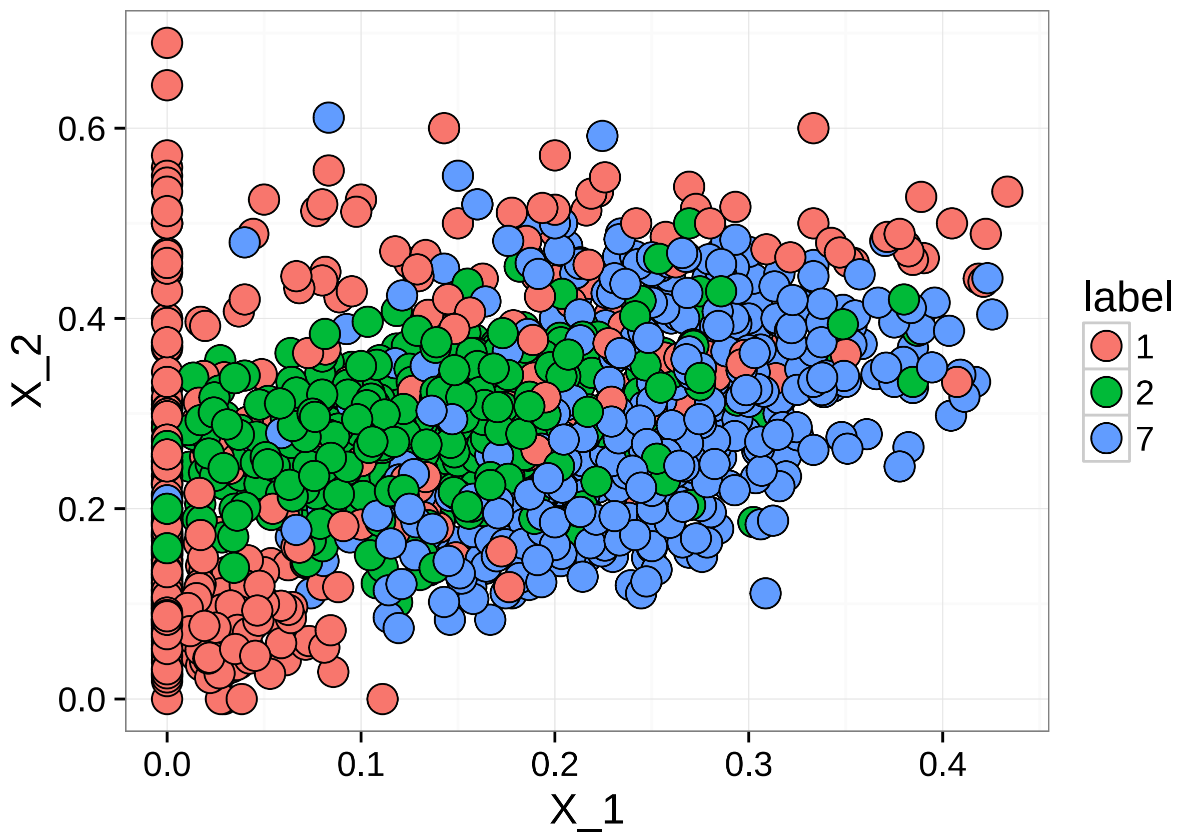
Three classes

```
## Warning in rm(dat27, X, X1, X2, fit, GS, X1s, X2s, yhat, f1, f2, f7):
## object 'dat27' not found
```



```r
##create the training set
set.seed(1)
dat <- sample_n(dat, 3000) %>% select(label, X_1, X_2) %>% mutate(label=as.factor(label))
library(caret)
inTrain <- createDataPartition(y = dat$label, p=0.5)
train_set <- slice(dat, inTrain$Resample1)
test_set <- slice(dat, -inTrain$Resample1)
```

```r
train_set %>% ggplot(aes(X_1,X_2,fill=label)) + geom_point(cex=5, pch=21)
```



```r
params <- train_set %>% group_by(label) %>%
  summarize(avg_1 = mean(X_1), avg_2 = mean(X_2), sd_1= sd(X_1), sd_2 = sd(X_2), r = cor(X_1,X_2))
params <-params %>% mutate(sd_1 = mean(sd_1), sd_2=mean(sd_1), r=mean(r))
params
```

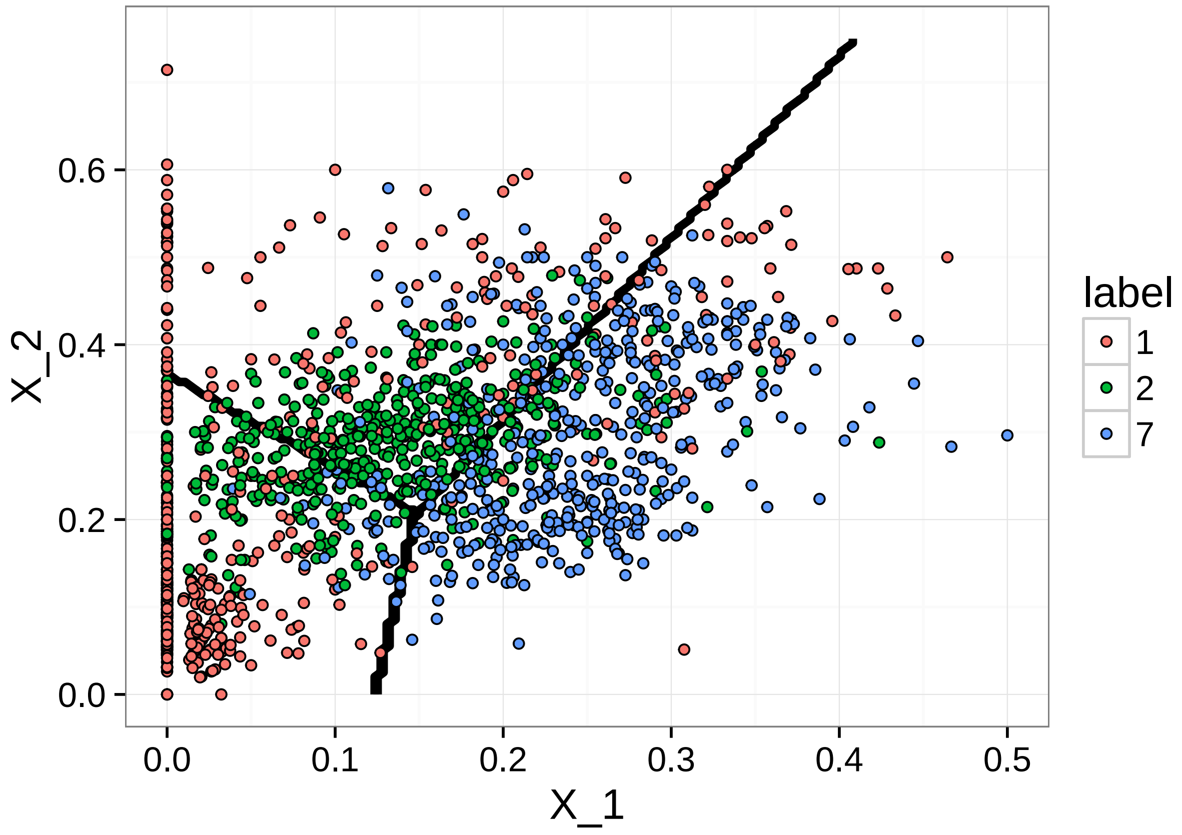
LDA

```
## Source: local data frame [3 x 6]
##
##    label avg_1 avg_2  sd_1  sd_2     r
##   (fctr) (dbl) (dbl) (dbl) (dbl) (dbl)
## 1      1 0.069  0.22 0.082 0.082  0.47
## 2      2 0.134  0.29 0.082 0.082  0.47
## 3      7 0.236  0.29 0.082 0.082  0.47
```

This defines the following estimate of $f(x_1, x_2)$ and the boundary becomes linear:

```r
library(mvtnorm)
p0 <- get_p(params[1,], true_f)
p1 <- get_p(params[2,], true_f)
p2 <- get_p(params[3,], true_f)

pred <- apply(cbind(p0, p1, p2),1,which.max)
```

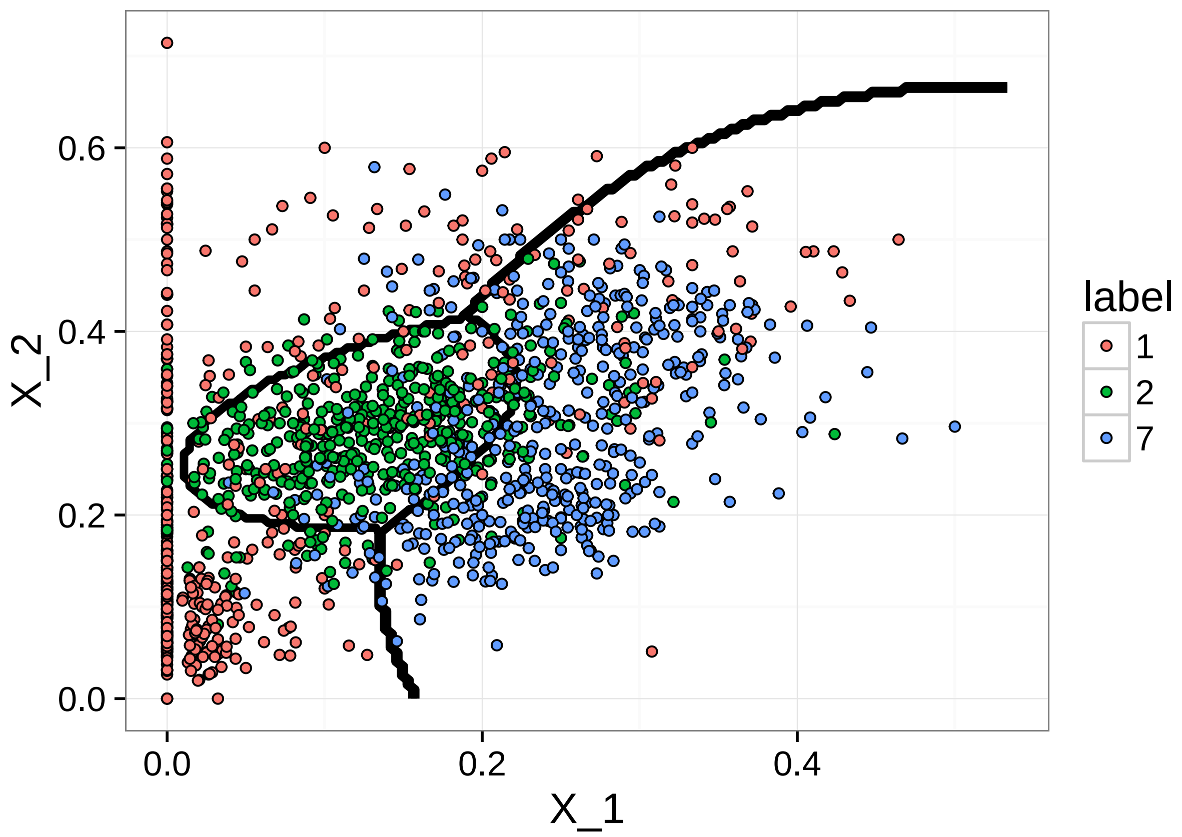
```
tmp <- true_f %>% mutate(pred=pred)
tmp %>% ggplot() +
  stat_contour(aes(x=X_1,y=X_2,z=pred),
               breaks=c(1,2,3),color="black",lwd=1.5) +
  geom_point(aes(X_1,X_2,fill=label), dat=test_set,pch=21)
```



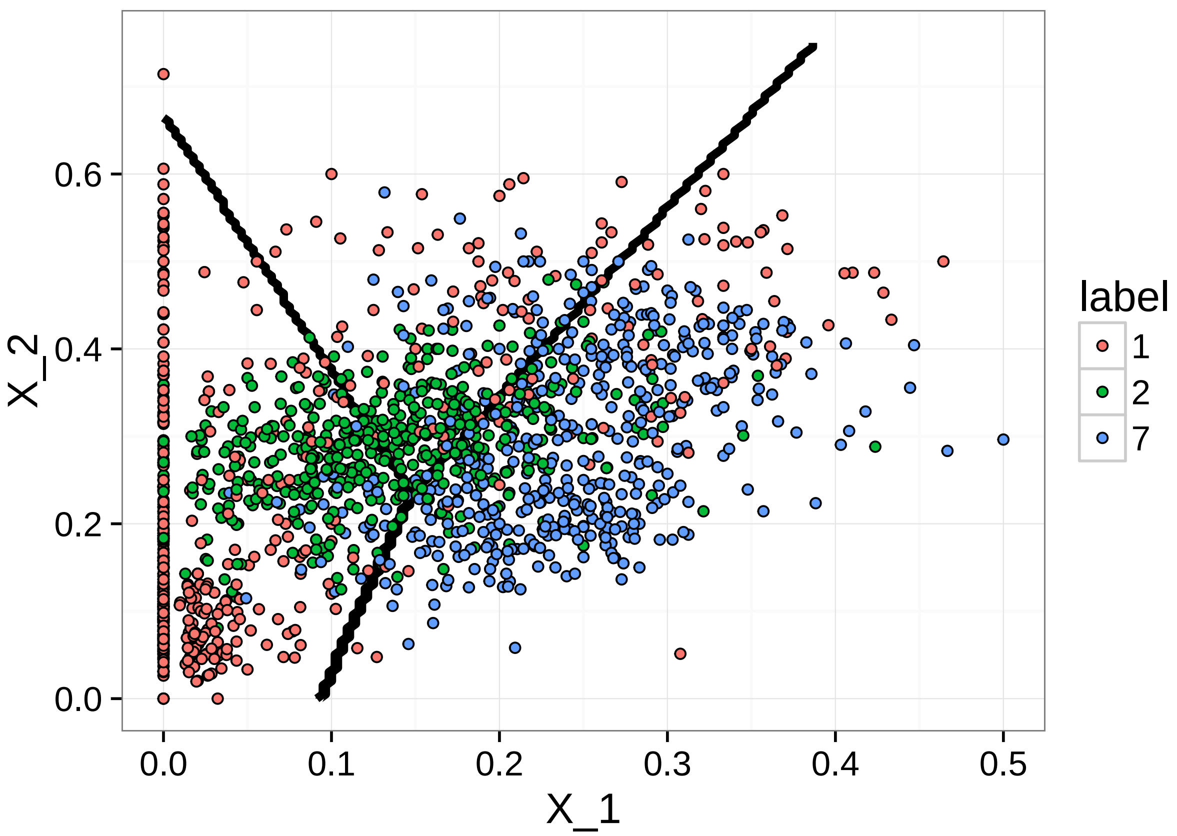
```
params <- train_set %>% group_by(label) %>%
  summarize(avg_1 = mean(X_1), avg_2 = mean(X_2), sd_1= sd(X_1), sd_2 = sd(X_2), r = cor(X_1,X_2))
```

QDA This defines the following estimate of $f(x_1, x_2)$ and the boundary becomes linear:

```
library(mvtnorm)
p0 <- get_p(params[1,], true_f)
p1 <- get_p(params[2,], true_f)
p2 <- get_p(params[3,], true_f)

pred <- apply(cbind(p0, p1, p2),1,which.max)
tmp <- true_f %>% mutate(pred=pred)
tmp %>% ggplot() +
  stat_contour(aes(x=X_1,y=X_2,z=pred),
               breaks=c(1,2,3),color="black",lwd=1.5) +
  geom_point(aes(X_1,X_2,fill=label), dat=test_set,pch=21)
```

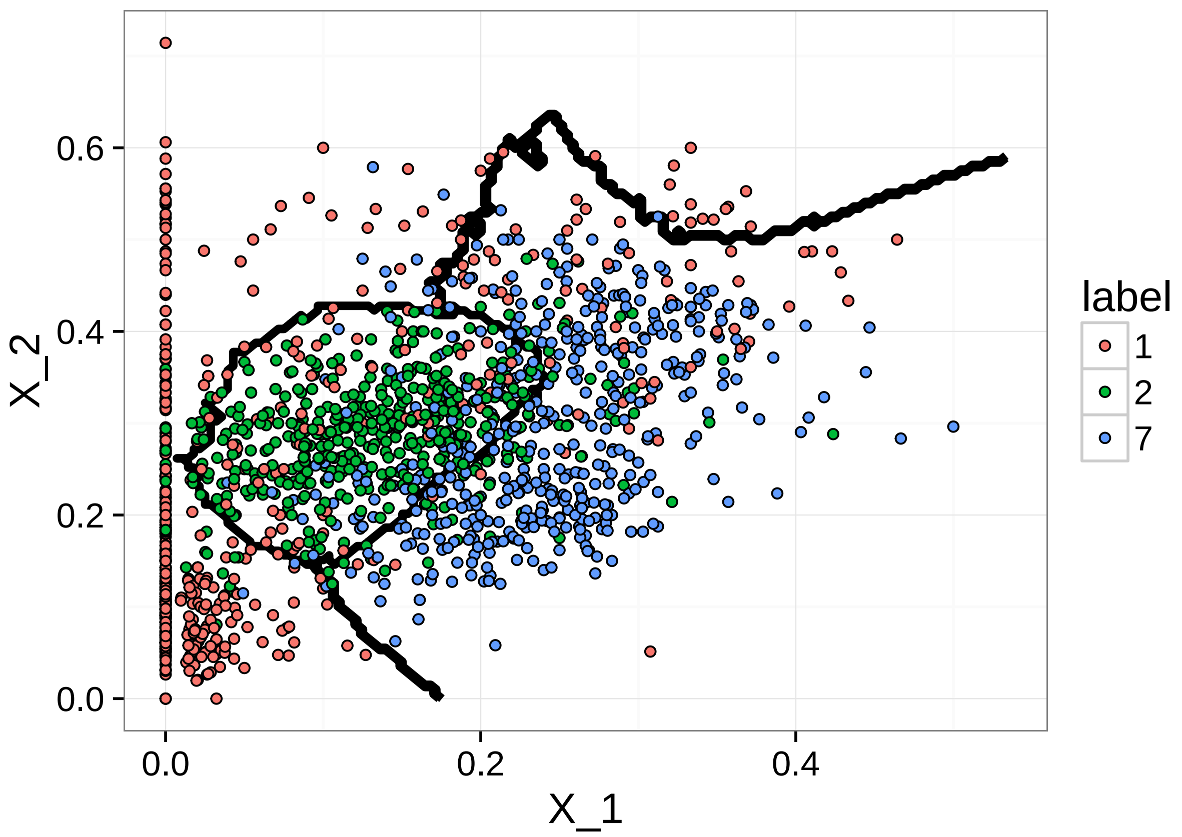


```r
library(caret)

fit1 <- glm(y~X_1+X_2, data=mutate(train_set,
                                   y=label=="1"),family="binomial")
fit2 <- glm(y~X_1+X_2, data=mutate(train_set,
                                   y=label=="2"),family="binomial")
fit7 <- glm(y~X_1+X_2, data=mutate(train_set,
                                   y=label=="7"),family="binomial")

f_hat1 <- predict(fit1, newdata = true_f, type = "response")
f_hat2 <- predict(fit2, newdata = true_f, type ="response")
f_hat7 <- predict(fit7, newdata = true_f, type = "response")

pred <- apply(cbind(f_hat1, f_hat2, f_hat7),1,which.max)
tmp <- true_f %>% mutate(pred=pred)
tmp %>% ggplot() +
  stat_contour(aes(x=X_1,y=X_2,z=pred),
             breaks=c(1,2,3),color="black",lwd=1.5) +
  geom_point(aes(X_1,X_2,fill=label), dat=test_set,pch=21)
```



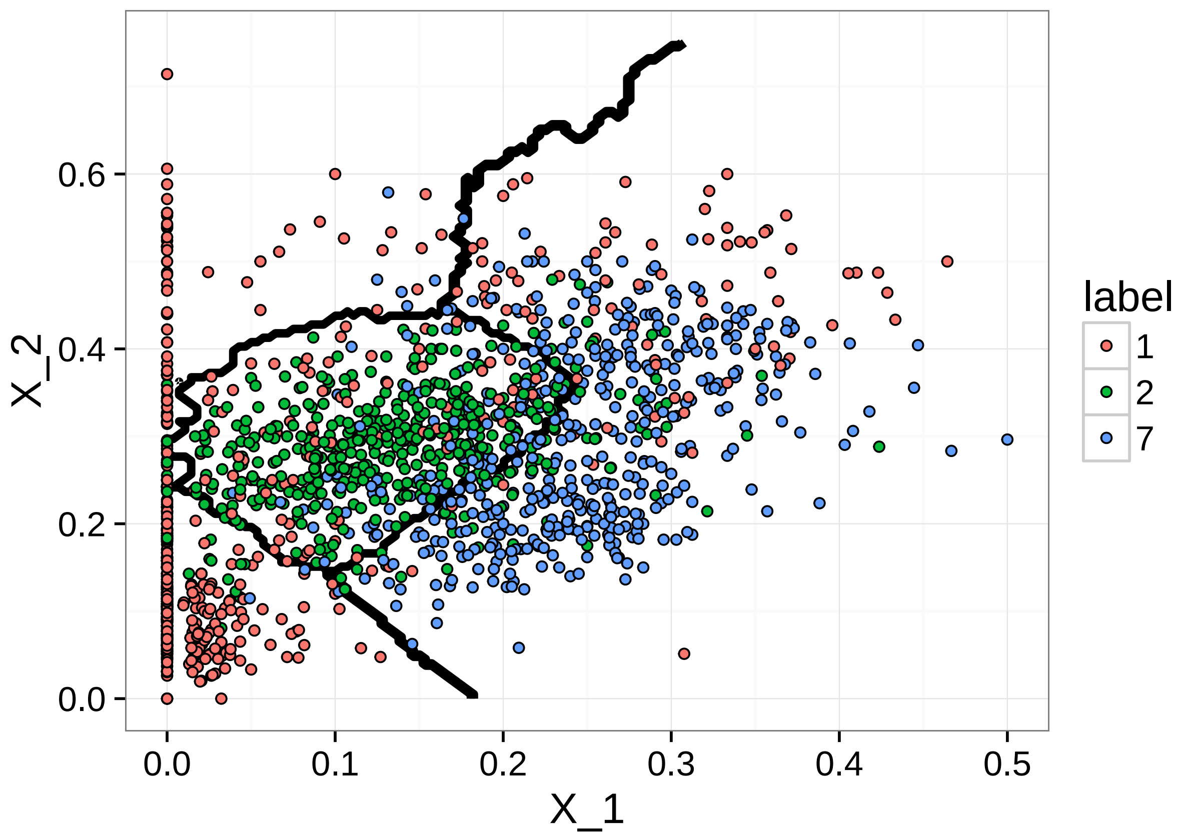
GLM

```
library(caret)

fit <- knn3(label~., data=train_set, k=51)
f_hat <- predict(fit, newdata = true_f)
f_hat_max <- apply(f_hat,1,max)
pred <- apply(f_hat,1,which.max)
tmp <- true_f %>% mutate(pred=pred)
tmp %>% ggplot() +
  stat_contour(aes(x=X_1,y=X_2,z=pred),
               breaks=c(1,2,3),color="black",lwd=1.5) +
  geom_point(aes(X_1,X_2,fill=label), dat=test_set,pch=21)
```



kNN

```r
library(caret)

fit <- knn3(label~., data=train_set, k=101)
f_hat <- predict(fit, newdata = true_f)
f_hat_max <- apply(f_hat,1,max)
pred <- apply(f_hat,1,which.max)
tmp <- true_f %>% mutate(pred=pred)
tmp %>% ggplot() +
  stat_contour(aes(x=X_1,y=X_2,z=pred),
               breaks=c(1,2,3),color="black",lwd=1.5) +
  geom_point(aes(X_1,X_2,fill=label), dat=test_set,pch=21)
```



```
##QDA
params <- train_set %>% group_by(label) %>%
  summarize(avg_1 = mean(X_1), avg_2 = mean(X_2), sd_1= sd(X_1), sd_2 = sd(X_2), r = cor(X_1,X_2))
p0 <- get_p(params[1,], test_set)
p1 <- get_p(params[2,], test_set)
p2 <- get_p(params[3,], test_set)

pred_qda <- apply(cbind(p0, p1, p2),1,which.max)

##LDA
params <-params %>% mutate(sd_1 = mean(sd_1), sd_2=mean(sd_1), r=mean(r))
p0 <- get_p(params[1,], test_set)
p1 <- get_p(params[2,], test_set)
p2 <- get_p(params[3,], test_set)
pred_lda <- apply(cbind(p0, p1, p2),1,which.max)

##GLM
fit1 <- glm(y~X_1+X_2, data=mutate(train_set,
                                   y=label=="1"),family="binomial")
fit2 <- glm(y~X_1+X_2, data=mutate(train_set,
                                   y=label=="2"),family="binomial")
fit7 <- glm(y~X_1+X_2, data=mutate(train_set,
                                   y=label=="7"),family="binomial")
```

```
f_hat1 <- predict(fit1, newdata = test_set, type = "response")
f_hat2 <- predict(fit2, newdata = test_set, type ="response")
f_hat7 <- predict(fit7, newdata = test_set, type = "response")

pred_glm <- apply(cbind(f_hat1, f_hat2, f_hat7),1,which.max)

library(caret)

##KNN 51
fit <- knn3(label~., data=train_set, k=51)
f_hat <- predict(fit, newdata = test_set)
pred_knn_51 <- apply(f_hat,1,which.max)

##KNN 101
fit <- knn3(label~., data=train_set, k=101)
f_hat <- predict(fit, newdata = test_set)
pred_knn_101 <- apply(f_hat,1,which.max)
```

Comparison Let's compare:

```
tab <- table(factor(pred_lda, labels=c("1","2","7")), test_set$label)
confusionMatrix(tab)
```

```
## Confusion Matrix and Statistics
##
##
##         1   2   7
##     1 344 128  25
##     2 131 261  80
##     7  57  86 388
##
## Overall Statistics
##
##                Accuracy : 0.662
##                  95% CI : (0.637, 0.686)
##     No Information Rate : 0.355
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.493
##  Mcnemar's Test P-Value : 0.00524
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 7
## Sensitivity             0.647    0.549    0.787
## Specificity             0.842    0.794    0.858
## Pos Pred Value          0.692    0.553    0.731
## Neg Pred Value          0.813    0.792    0.892
## Prevalence              0.355    0.317    0.329
## Detection Rate          0.229    0.174    0.259
## Detection Prevalence    0.331    0.315    0.354
## Balanced Accuracy       0.744    0.672    0.823
```

```r
confusionMatrix(tab)$overall[1]
```

```
## Accuracy
##     0.66
```

```r
tab <- table(factor(pred_qda, labels=c("1","2","7")), test_set$label)
confusionMatrix(tab)$overall[1]
```

```
## Accuracy
##     0.76
```

```r
tab <- table(factor(pred_glm, labels=c("1","2","7")), test_set$label)
confusionMatrix(tab)$overall[1]
```

```
## Accuracy
##     0.63
```

```r
tab <- table(factor(pred_knn_51, labels=c("1","2","7")), test_set$label)
confusionMatrix(tab)$overall[1]
```

```
## Accuracy
##     0.77
```

```r
tab <- table(factor(pred_knn_101, labels=c("1","2","7")), test_set$label)
confusionMatrix(tab)$overall[1]
```

```
## Accuracy
##     0.76
```