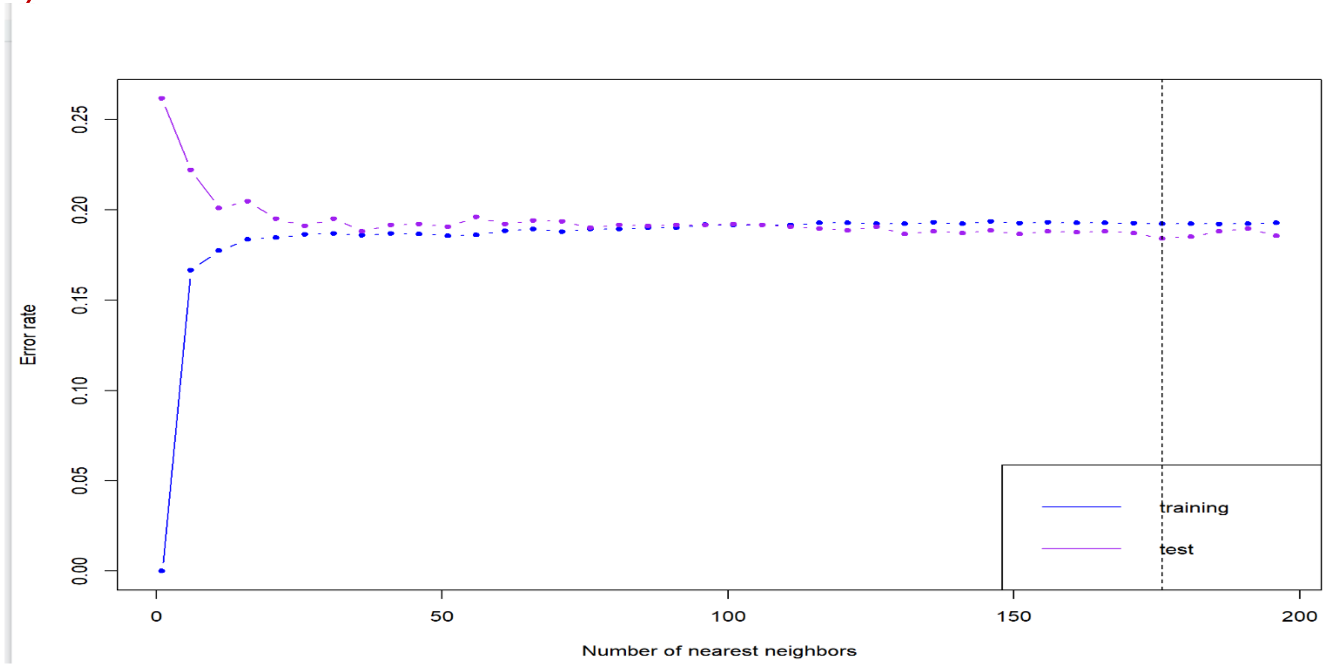
Mini Project #1
Ashish Mani Acharya

Section 1

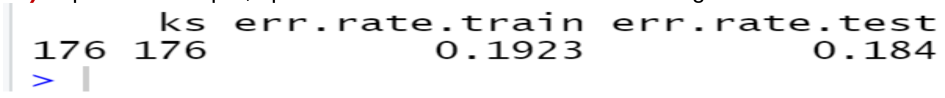
1a) I have done KNN with K=1,6,….,200 as suggested

1b)

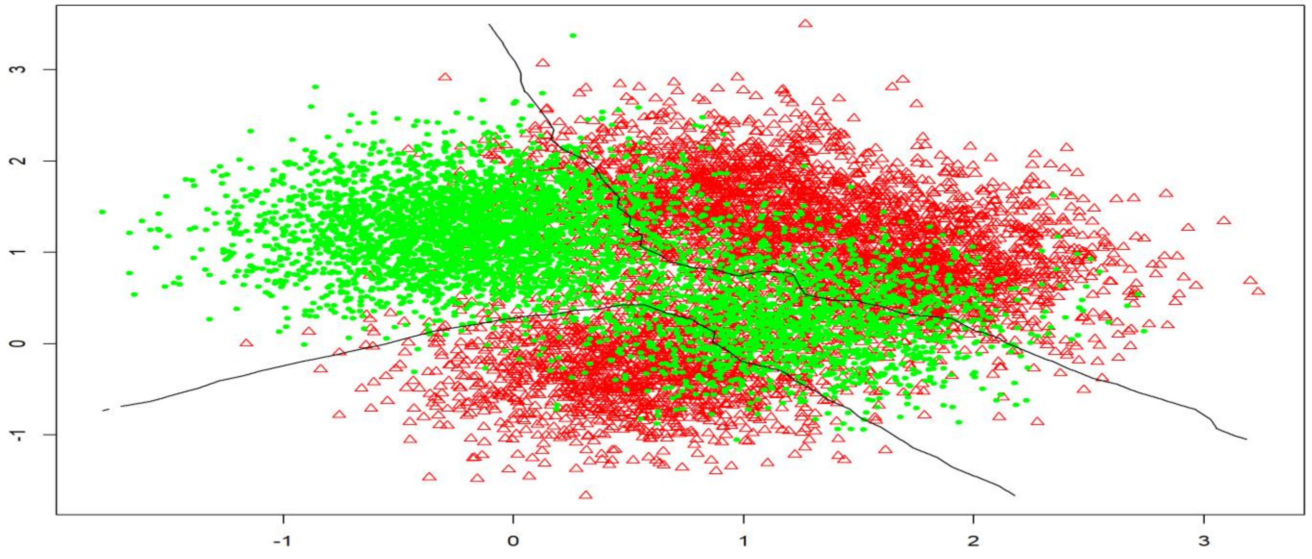


We can see that training error rate increases significantly but stabilizes as K's value increases. Similarly testing error rate goes U shaped as we discussed in the class. It is not as dramatic or as visible of a U shape but we can see test error rate is making a slight U turn.

1c) As per the R output, optimal value is 176 and minimum training error rate is 0.1923 while minimum test error is 0.184.



```
     ks err.rate.train err.rate.test
176 176         0.1923         0.184
>
```



1d)
The decision boundaries is supposed to recognize those regions of the input class space that matches to each class. As we can see the decision boundary clearly separates green class with red class. Decision boundary is supposed to delaminate 2 sides on its either side. Looking at the picture we can say that it manages to do what decision boundary is supposed to do as majority of green and red are divided by decision boundary. Hence it seems sensible.

Mini Project #1
Ashish Mani Acharya

2(a) Show that MSE{ ˆf(x0)} = (Bias{ ˆf(x0)}) 2 + var{ ˆf(x0)}

$MSE = E(\widehat{f(x_0)} - f(x_0))^2$*(definition of MSE)*

$MSE = E((\widehat{f(x_0)} - E(\widehat{f(x_0)})) + (E(\widehat{f(x_0)}) - f(x_0)))^2$

$MSE = E((\widehat{f(x_0)} - E(\widehat{f(x_0)}))^2 + E\left(\left(E(\widehat{f(x_0)}) - f(x_0)\right)\right)^2 - 2E(\left(\left(\widehat{f(x_0)} - E(\widehat{f(x_0)})\right)\left(E(\widehat{f(x_0)}) - f(x_0)\right)\right)$
$[\because\ of\ definition\ of\ variance\ and\ definition\ of\ bias\ ]$

$MSE = var(\widehat{f(x_0)}) + bias(\widehat{f(x_0)})^2 + 2\left(E(\widehat{f(x_0)}) - f(x_0)\right)E\left(\widehat{f(x_0)} - E(\widehat{f(x_0)})\right)$

$MSE = var(\widehat{f(x_0)}) + bias(\widehat{f(x_0)})^2 + 2\left(E(\widehat{f(x_0)}) - f(x_0)\right)\left(E(\widehat{f(x_0)}) - E(\widehat{f(x_0)})\right)$

$MSE = var(\widehat{f(x_0)}) + bias(\widehat{f(x_0)})^2$

2(b) Show that E(Yˆ 0 − Y0) 2 = (Bias{ ˆf(x0)})^ 2 + var{ ˆf(x0)} + σ ^2 .

$E(\hat{Y}_0 - Y_0)^2 = E(\widehat{f(x_0)} - (f(x_0) + \epsilon))^2$

$E(\hat{Y}_0 - Y_0)^2 = E((\widehat{f(x_0)} - (f(x_0))^2 + E(\epsilon^2) - E\left(2\epsilon\left((\widehat{f(x_0)} - (f(x_0))\right)\right)$

$E(\hat{Y}_0 - Y_0)^2 = MSE + (var(\epsilon^2) + E(\epsilon)^2) - 2(\widehat{f(x_0)} - \widehat{f(x_0)})E(\epsilon)$
$[\because\ (var(\epsilon^2) = \sigma^2\ \&\ E(\epsilon) = 0]$

$E(\hat{Y}_0 - Y_0)^2 = bias(\widehat{f(x_0)})^2 + var(\widehat{f(x_0)}) + \sigma^2$

Section 2

```
test <- read.csv("/Users/alexk/OneDrive/Desktop/stat 6340/1-test_data.csv", header = TRUE)
train <- read.csv("/Users/alexk/OneDrive/Desktop/stat 6340/1-training_data.csv", header = TRUE)
head(test)
head(train)
str(test)
str(train)
train.x <- cbind(train$x.1,train$x.2)
train.y<-(train$y)
test.x<-cbind(test$x.1,test$x.2)
test.y<-cbind(test$y)

#applying KNN#
library(class)
set.seed(1)
mod.train <- knn(train.x, train.x, train.y, k = 1)
set.seed(1)
mod.test <- knn(train.x, test.x, train.y, k = 1)
ks <- c(seq(1, 200, by = 5))

nks <- length(ks)
err.rate.train <- numeric(length = nks)
err.rate.test <- numeric(length = nks)
names(err.rate.train) <- names(err.rate.test) <- ks

min(err.rate.test)

for (i in seq(along = ks)) {
  set.seed(1)
  mod.train <- knn(train.x, train.x, train.y, k = ks[i])
  set.seed(1)
  mod.test <- knn(train.x, test.x, train.y, k = ks[i])
  err.rate.train[i] <- 1 - sum(mod.train == train.y)/length(train.y)
  err.rate.test[i] <- 1 - sum(mod.test == test.y)/length(test.y)
}
plot(ks, err.rate.train, xlab = "Number of nearest neighbors", ylab = "Error rate",
     type = "b", ylim = range(c(err.rate.train, err.rate.test)), col = "blue", pch = 20)
lines(ks, err.rate.test, type="b", col="purple", pch = 20)
legend("bottomright", lty = 1, col = c("blue", "purple"), legend = c("training", "test"))
abline(v=176,lty=20)
result <- data.frame(ks, err.rate.train, err.rate.test)

n.grid <- 50
x1.grid <- seq(f = min(train.x[, 1]), t = max(train.x[, 1]), l = n.grid)
x2.grid <- seq(f = min(train.x[, 2]), t = max(train.x[, 2]), l = n.grid)
grid <- expand.grid(x1.grid, x2.grid)
min <- result[err.rate.test == min(result$err.rate.test), ]#finding the optimal value #
min

k.opt <- 176
set.seed(1)
mod.opt <- knn(train.x, grid, train.y, k = k.opt, prob = T)
prob <- attr(mod.opt, "prob")
prob <- ifelse(mod.opt == "yes", prob, 1 - prob)
prob <- matrix(prob, n.grid, n.grid)

plot(train.x, pch = ifelse(train.y == "yes", 2, 20), col = ifelse(train.y == "yes","red", "green"))

contour(x1.grid, x2.grid, prob, levels = 0.5, labels = "", xlab = "", ylab = "",
        main = "", add = T)
```