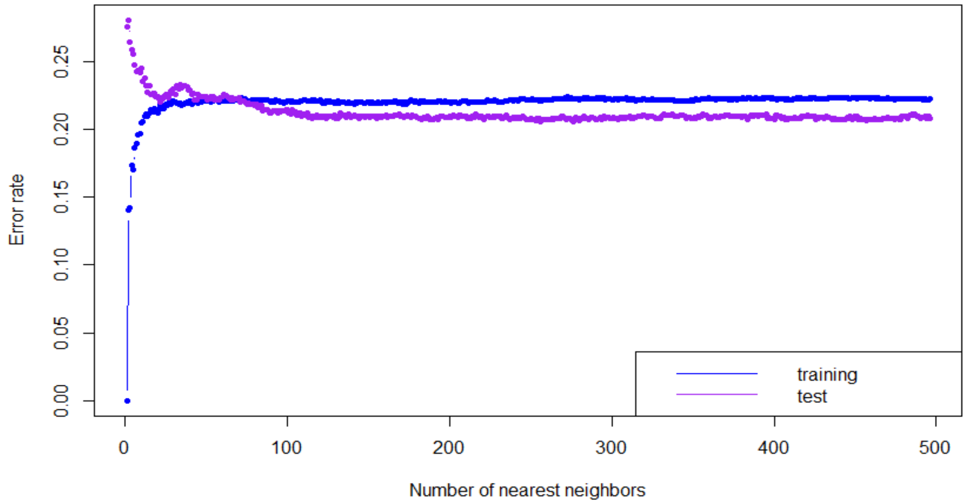
Mini Project 1 (Solutions)

Section 1

1) In this problem we use the KNN classification method on the training data set and predict the classes of the test data set.
   a) `knn()` function was used for the K values = `seq(1, 496, by = 1)`
   b) Training and test error rates were calculated for different K values and they were plotted against K values as follows.

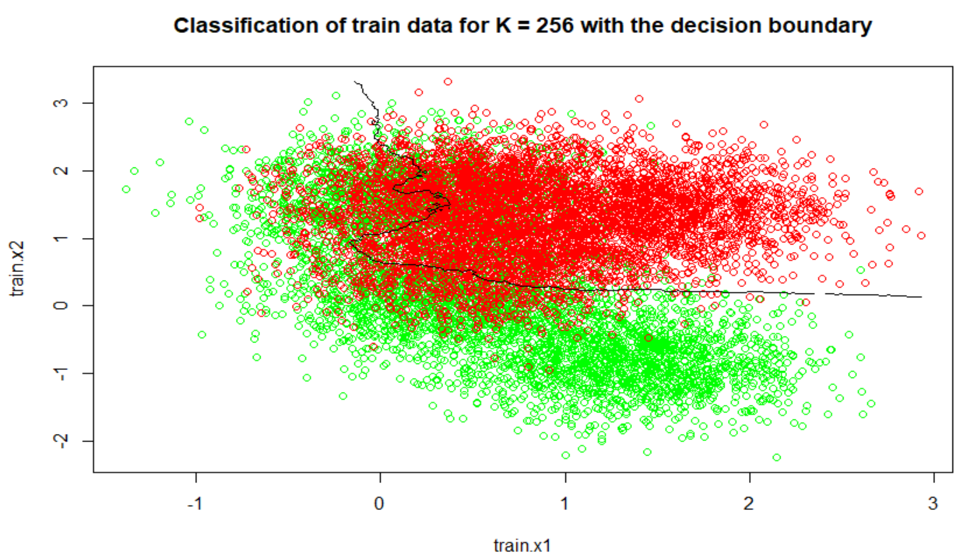


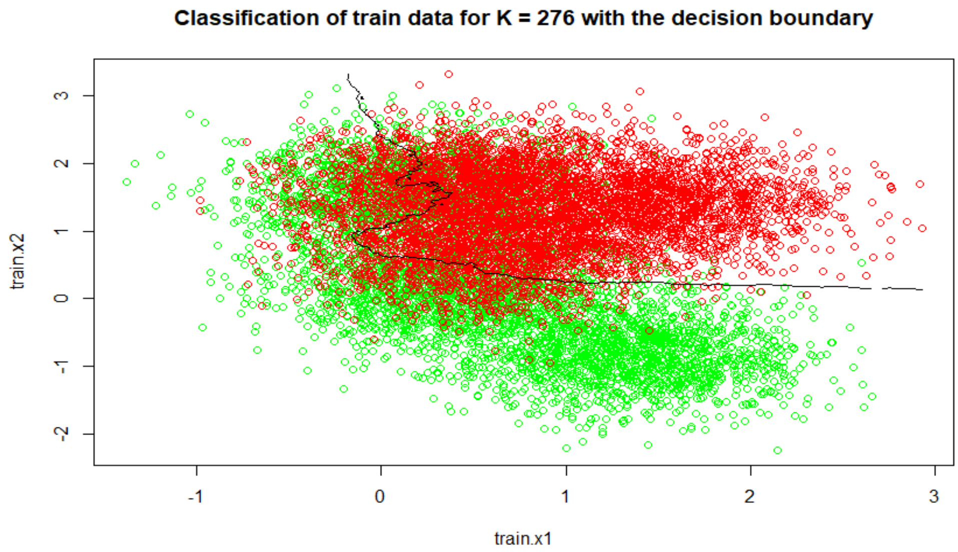
The plot expected to have a concave up curve for test error rates and concave down curve for training error rates, separated from the $var(\epsilon)$ horizontal line and they do not cross over each other. This seems to be true in the region $0 \leq K < 50$ but when K is increased, the test error rates curve cross over the other.

   c) Optimal value of K is found by finding the minimum value of test error rates and there were two K values that minimizes the test error rates, K = 256 and K = 276 . Associated training and test error rates for optimum K values are shown on the following table.

ks	err.rate.train	err.rate.test
256	0.2221	0.206
257	0.2226	0.206

   d) Since we had two K values that makes the test error rate minimize, corresponding decision boundaries were plotted separately as follows.

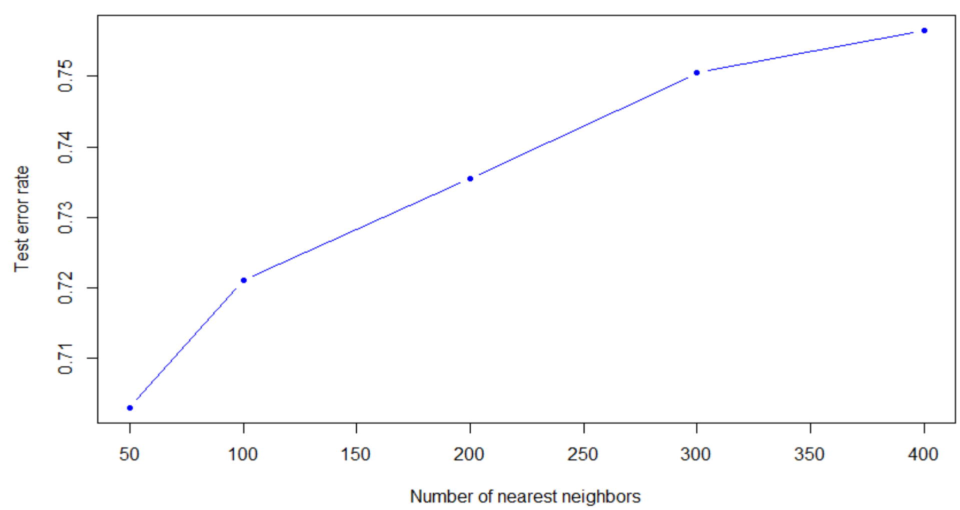




2) 1/5th of the data set e CIFAR-10 were used to classify using KNN method with K = 50, 100, 200, 300, 400.

a) Following table shows the error rates for different K values.

ks	50	100	200	300	400
err.rate.test	0.7030	0.7210	0.7355	0.7505	0.7565



b) Minimum error of test error rate was 0.7030 and the associate K value was 50. Below is the confusion matrix.

```
          y.test
mod.test    0   1   2   3   4   5   6   7   8   9
       0  117  11  31   9   4   5   4   6  19   6
       1    0   7   0   1   0   0   0   0   0   2
       2   16  25  66  31  26  42  32  34   8  16
       3    1   2   2  17   0   6   1   4   0   1
       4   32  55  94  81 120  78  86  93  16  49
       5    0   3   1   8   0  30   0   5   4   2
       6    4  25  10  33   6  21  51  13   5  12
       7    1   2   1   2   0   2   1  20   2   5
       8   44  62  10  20  12  11  10  25 137 101
       9    0   6   1   2   0   1   2   4   1  26
```

- The KNN classified 2000 data into 10 classes.
- Accuracy of the classifier = $591/2000 = 0.2955$
- Misclassification rate = $1 - 0.2955 = 0.7045$
- Sensitivity of each class

Class	Sensitivity
0	117/215 = 0.5442
1	7/198 = 0.0354
2	66/216 = 0.3056
3	17/204 = 0.0833
4	120/168 = 0.7143
5	30/196 = 0.1531
6	51/187 = 0.2727
7	20/204 = 0.0980
8	137/192 = 0.7135
9	26/220 = 0.1182

c) Since the optimum K value is unknown, we need to check several K values. As shown in part a) even to get an approximation for the optimum K is a bit tricky. Also for a large data set like CIFAR-10, the computation time for KNN was high. By looking at the sensitivity of each classes, only few classes were classified with high probability. Thus we can't guarantee that KNN is a good classification with this data set.

Section 2

```
# problem 1
library(class) # for knn

training.data <- read.csv(file.choose(), header = T) # Get the training data
test.data <- read.csv(file.choose(), header = T) # Get the test data

# seperate the data as train and test for knn
set.seed(1)
train.X <- cbind(training.data$x.1, training.data$x.2)
train.Y <- training.data$y
test.X <- cbind(test.data$x.1, test.data$x.2)
test.Y <- test.data$y

# part a)
ks <- seq(1, 496, by = 1) # set the K values for KNN
nks <- length(ks)
# create vectors to store train and test error rates
err.rate.train <- numeric(length = nks)
err.rate.test <- numeric(length = nks)
names(err.rate.train) <- names(err.rate.test) <- ks

# set KNN for train and test data
for (i in seq(along = ks)) {
  set.seed(1)
  mod.train <- knn(train.X, train.X, train.Y, k = ks[i]) # KNN for train
  set.seed(1)
  mod.test <- knn(train.X, test.X, train.Y, k = ks[i]) # KNN for test
  err.rate.train[i] <- mean(mod.train != train.Y) # calculate train error rate
  err.rate.test[i] <- mean(mod.test != test.Y) # calculate test error rate

}
```

```
# plot train and test error rates against K values
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"))

# part c) calculate minimum of test error rates
result <- data.frame(ks, err.rate.train, err.rate.test)
optimal.point <- result[err.rate.test == min(result$err.rate.test), ]

# > optimal.point
#      ks err.rate.train err.rate.test
# 256 256         0.2221         0.206
# 276 276         0.2226         0.206

# part d)
n.grid <- 200
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) # grid to plot the decision boundary

# 1st plot with K = 256
k.opt <- optimal.point[1,1] # 256
set.seed(1)
mod.opt <- knn(train.X, grid, train.Y, k = k.opt, prob = T) # KNN with K = 256
prob <- attr(mod.opt, "prob") # prob is voting fraction for winning class
prob <- ifelse(mod.opt == "yes", prob, 1 - prob) # now it is voting fraction for
Direction == "Up"
prob <- matrix(prob, n.grid, n.grid)

plot(train.X, col = ifelse(train.Y == "yes", "green", "red"))
contour(x1.grid, x2.grid, prob, levels = 0.5, labels = "", xlab = "", ylab = "",
main = "", add = T)

# 2nd plot with K = 256

k.opt <- optimal.point[2,1] # 276
set.seed(1)
mod.opt <- knn(train.X, grid, train.Y, k = k.opt, prob = T) # KNN with K = 276
prob <- attr(mod.opt, "prob") # prob is voting fraction for winning class
prob <- ifelse(mod.opt == "yes", prob, 1 - prob) # now it is voting fraction for
Direction == "Up"
prob <- matrix(prob, n.grid, n.grid)

plot(train.X, col = ifelse(train.Y == "yes", "green", "red"))
contour(x1.grid, x2.grid, prob, levels = 0.5, labels = "", xlab = "", ylab = "",
main = "", add = T)
```

#problem 2

```
library(class) # for KNN
library(keras) # for cifar10 data
cifar <- dataset_cifar10()
str(cifar)
x.train <- cifar$train$x
y.train <- cifar$train$y
x.test <- cifar$test$x
y.test <- cifar$test$y

# reshape the images as vectors (column-wise)
# (aka flatten or convert into wide format)
```

```
# (for row-wise reshaping, see ?array_reshape)
dim(x.train) <- c(nrow(x.train), 32*32*3) # 50000 x 3072
dim(x.test) <- c(nrow(x.test), 32*32*3) # 50000 x 3072
# rescale the x to lie between 0 and 1
x.train <- x.train/255
x.test <- x.test/255
# categorize the response
y.train <- as.factor(y.train)
y.test <- as.factor(y.test)
# randomly sample 1/5 of the data to reduce computing time
set.seed(1)
id.train <- sample(1:50000, 10000)
id.test <- sample(1:10000, 2000)
x.train <- x.train[id.train,]
y.train <- y.train[id.train]
x.test <- x.test[id.test,]
y.test <- y.test[id.test]

# part a)
ks <- c(50, seq(100, 400, by = 100)) # set K values for KNN
nks <- length(ks)
err.rate.test <- numeric(length = nks)
names(err.rate.test) <- ks

# set KNN for test data
for (i in seq(along = ks)) {
  set.seed(1)
  mod.test <- knn(x.train, x.test, y.train, k = ks[i])
  err.rate.test[i] <- mean(mod.test != y.test)
}

> err.rate.test
    50    100    200    300    400
0.7030 0.7210 0.7355 0.7505 0.756

plot(ks, err.rate.test, xlab = "Number of nearest neighbors", ylab = "Test error
rate", type = "b", ylim = range(c(err.rate.test)), col = "blue", pch = 20)


# part b)
result <- data.frame(ks, err.rate.test)
optimal.point <- result[err.rate.test == min(result$err.rate.test), ]
> optimal.point
   ks err.rate.test
50 50         0.703


> mod.test <- knn(x.train, x.test, y.train, k = 50) KNN for K=50
> table(mod.test, y.test) # confusion matrix
        y.test
mod.test   0   1   2   3   4   5   6   7   8   9
       0 117  11  31   9   4   5   4   6  19   6
       1   0   7   0   1   0   0   0   0   0   2
       2  16  25  66  31  26  42  32  34   8  16
       3   1   2   2  17   0   6   1   4   0   1
       4  32  55  94  81 120  78  86  93  16  49
       5   0   3   1   8   0  30   0   5   4   2
       6   4  25  10  33   6  21  51  13   5  12
       7   1   2   1   2   0   2   1  20   2   5
       8  44  62  10  20  12  11  10  25 137 101
       9   0   6   1   2   0   1   2   4   1  26
```