Assignment3

Zeyi Pan

4/27/2020

Q1

The estinated model is $\tilde{Y} = \frac{1}{1+exp(-3.8938+0.1986X)}$.
$OR = e^{0.1986\times(15-5)} = 7.2856$
$CI_\beta = \tilde{\beta} \pm 2 \times SE = 0.1986 \pm 2 \times 0.0915 = [0.0157, 0.3815]$
$CI_{OR} = exp(CI_\beta) = [1.0158, 1.4645]$
The odds ratio is 7.2856. The confidence interval is [1.0158, 1.4645].

```
beta = 0.19859
SE = 0.09147
OR = exp(beta*(15-5))
OR
```

```
## [1] 7.285601
```

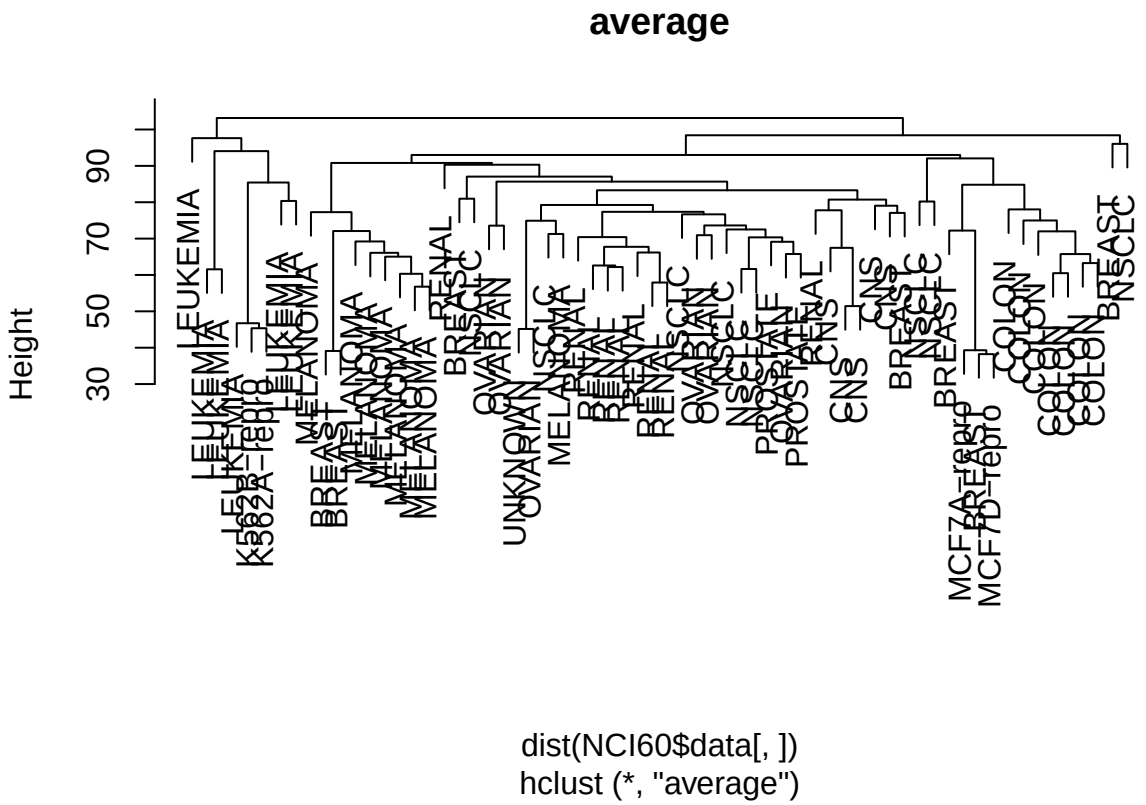
```
CI_beta.up = beta + 2*SE
CI_beta.low = beta - 2*SE
CI_OR.up = exp(CI_beta.up)
CI_OR.low = exp(CI_beta.low)
c(CI_OR.low, CI_OR.up)
```

```
## [1] 1.015773 1.464524
```

Q2

(a)

```
library(ISLR)
hfit = hclust(dist(NCI60$data[,]), method="average")
plot(hfit,labels=NCI60$labs, main='average')
```



(b)

Maximum cophenestic distances between samples within a type are:
(1) MELANOMA: 88.9280
(2) RENAL: 95.9021
(3) COLON: 78.0307

Minimum cophenetic distances between a sample in a type and a sample not in this type:
(1) MELANOMA and RENAL: 79.5328
(2) MELANOMA and COLON: 93.4497
(3) RENAL and COLON: 93.4497

When the maximum cophenetic distance between samples within a type is smaller than the minimum cophenetic distance mentioned above, this type will cluster well and doesn't mix up with other types.

```
m = which(NCI60$labs == "MELANOMA")
m
```

```
## [1] 23 56 59 60 61 62 63 64
```

```
r = which(NCI60$labs == "RENAL")
r
```

```
## [1]  4 11 12 13 14 15 16 17 20
```

```r
c = which(NCI60$labs == "COLON")
c
```

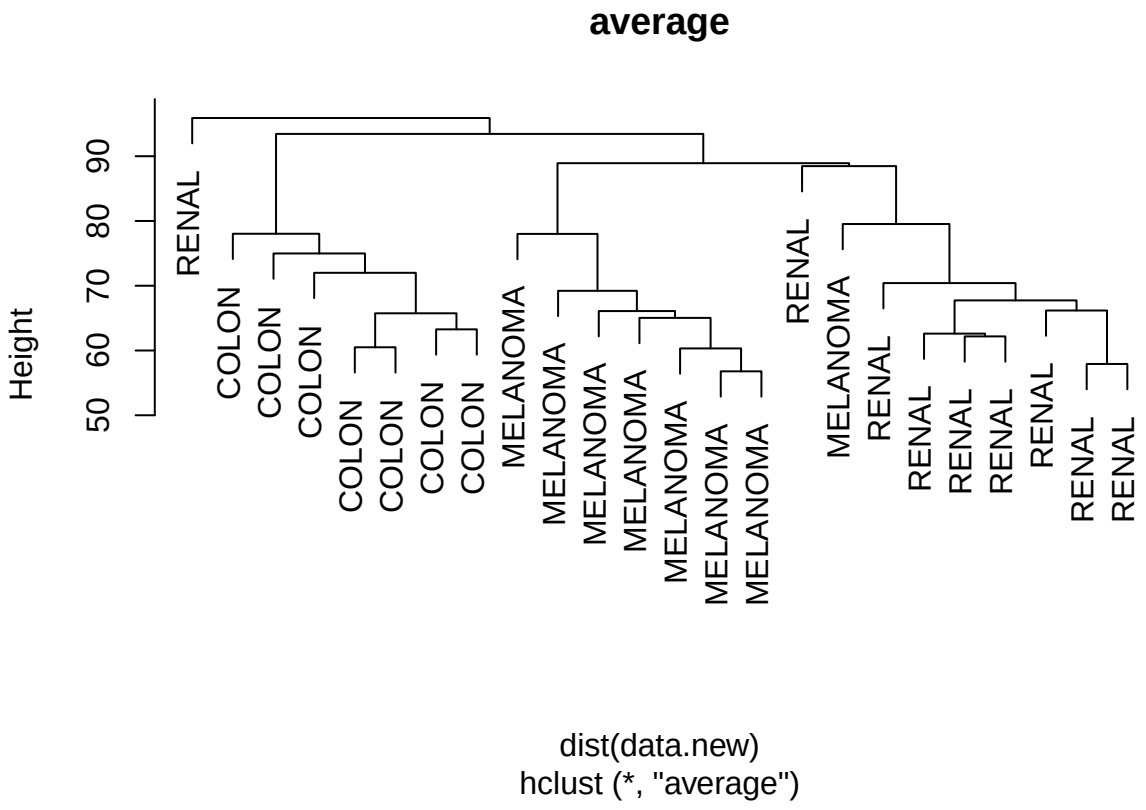
```
## [1] 42 43 44 45 46 47 48
```

```r
position = c(m, r, c)
data.new = NCI60$data[position,]
labs.new = NCI60$labs[position]
hfit.new = hclust(dist(data.new), method="average")
cophenetic(hfit.new)
```

```
##          V23      V56      V59      V60      V61      V62      V63      V64
## V56 88.92802
## V59 88.92802 78.00480
## V60 88.92802 78.00480 66.08965
## V61 88.92802 78.00480 66.08965 60.32136
## V62 88.92802 78.00480 66.08965 60.32136 56.78015
## V63 88.92802 78.00480 69.21691 69.21691 69.21691 69.21691
## V64 88.92802 78.00480 66.08965 65.03494 65.03494 65.03494 69.21691
## V4  88.47362 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802
## V11 79.53280 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802
## V12 79.53280 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802
## V13 79.53280 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802
## V14 79.53280 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802
## V15 79.53280 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802
## V16 79.53280 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802
## V17 79.53280 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802 88.92802
## V20 95.90211 95.90211 95.90211 95.90211 95.90211 95.90211 95.90211 95.90211
## V42 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V43 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V44 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V45 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V46 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V47 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V48 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
##           V4      V11      V12      V13      V14      V15      V16      V17
## V56
## V59
## V60
## V61
## V62
## V63
## V64
## V4
## V11 88.47362
## V12 88.47362 66.17789
## V13 88.47362 66.17789 57.91726
## V14 88.47362 67.73732 67.73732 67.73732
## V15 88.47362 67.73732 67.73732 67.73732 62.60555
## V16 88.47362 67.73732 67.73732 67.73732 62.60555 62.17806
## V17 88.47362 70.41020 70.41020 70.41020 70.41020 70.41020 70.41020
## V20 95.90211 95.90211 95.90211 95.90211 95.90211 95.90211 95.90211 95.90211
## V42 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
```

```
## V43 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V44 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V45 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V46 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V47 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
## V48 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967 93.44967
##          V20      V42      V43      V44      V45      V46      V47
## V56
## V59
## V60
## V61
## V62
## V63
## V64
## V4
## V11
## V12
## V13
## V14
## V15
## V16
## V17
## V20
## V42 95.90211
## V43 95.90211 71.99333
## V44 95.90211 60.49651 71.99333
## V45 95.90211 65.74220 71.99333 65.74220
## V46 95.90211 65.74220 71.99333 65.74220 63.26414
## V47 95.90211 78.03069 78.03069 78.03069 78.03069 78.03069
## V48 95.90211 74.98528 74.98528 74.98528 74.98528 74.98528 78.03069
```

```
plot(hfit.new,labels=labs.new, main='average')
```




```
hfit.new$height
```

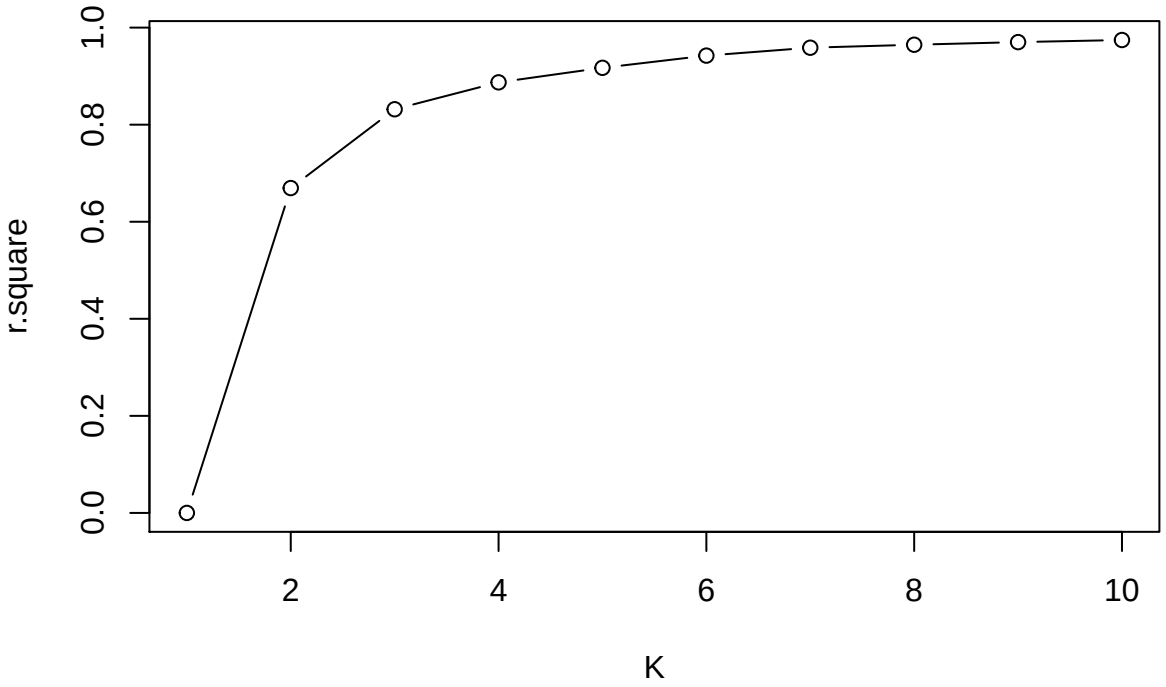
```
##  [1] 56.78015 57.91726 60.32136 60.49651 62.17806 62.60555 63.26414 65.03494
##  [9] 65.74220 66.08965 66.17789 67.73732 69.21691 70.41020 71.99333 74.98528
## [17] 78.00480 78.03069 79.53280 88.47362 88.92802 93.44967 95.90211
```

Q3

(a)

K^* is 3.

```
library(MASS)
mammals.log = log(mammals)
r.square = rep(0,10)
for (i in 1:10) {
  fit = kmeans(mammals.log,centers=i, nstart=100)
  r.square[i] = 1 - fit$tot.withinss/fit$totss
}
plot(r.square, xlab="K" ,type='b')
```



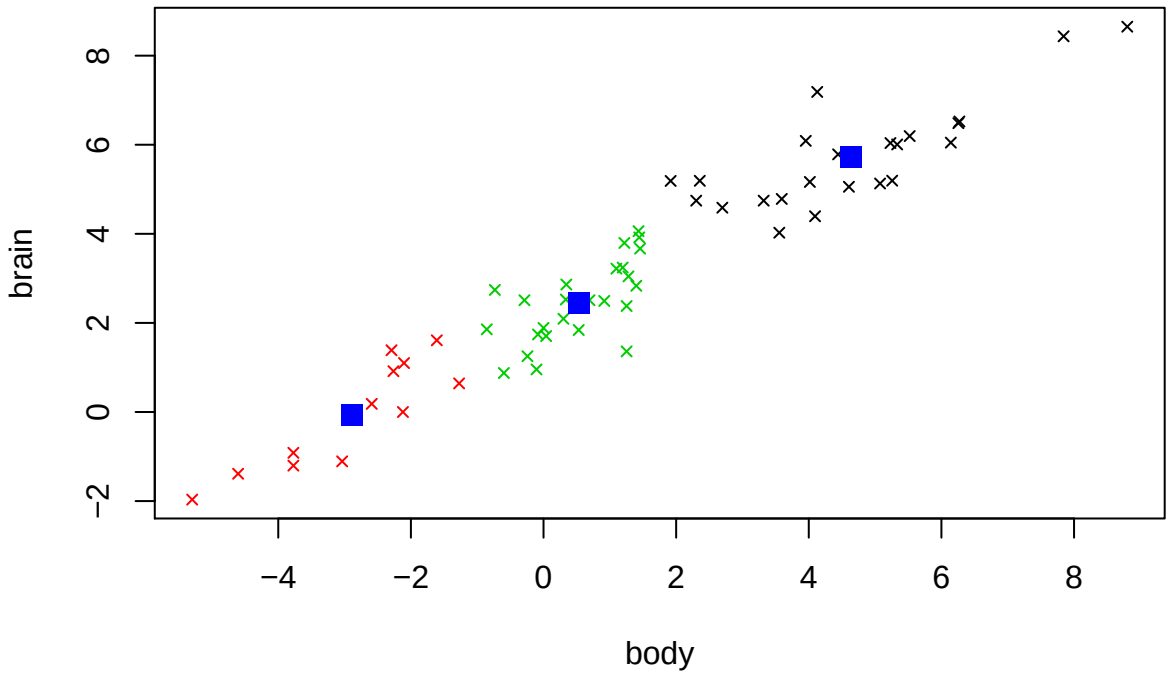

```
k.star = which(r.square>=0.8)[1]
k.star
```

```
## [1] 3
```

(b)

It doesn't make sense to cluster like this. For some clusters, the body and brain weights are quite different within a cluster such as human and african elephant are in the same cluster. For some clusters, the weights of animals belong to different clusters are very close. For example, atlantic ground squirrels and ground squirrels are quite close to each other but are assigned into different clusters.

```
fit.star = kmeans(mammals.log,centers=k.star, nstart=100)
mammals.new = cbind(mammals.log, fit.star$cluster)
plot(mammals.new[,1:2], col=mammals.new[,3], pch=4, cex=0.7)
points(fit.star$centers, col="blue", pch=15, cex=1.5)
```




```
row.names(mammals)[which(fit.star$cluster==1)]
```

```
##  [1] "Cow"              "Grey wolf"        "Goat"             "Roe deer"
##  [5] "Asian elephant"   "Donkey"           "Horse"            "Patas monkey"
##  [9] "Giraffe"          "Gorilla"          "Grey seal"        "Human"
## [13] "African elephant" "Rhesus monkey"    "Kangaroo"         "Okapi"
## [17] "Sheep"            "Jaguar"           "Chimpanzee"       "Baboon"
## [21] "Giant armadillo"  "Pig"              "Brazilian tapir"
```

```
row.names(mammals)[which(fit.star$cluster==2)]
```

```
##  [1] "Ground squirrel"          "Lesser short-tailed shrew"
##  [3] "Star-nosed mole"          "Big brown bat"
##  [5] "Galago"                   "Golden hamster"
##  [7] "Mouse"                    "Little brown bat"
##  [9] "Rat"                      "E. American mole"
## [11] "Mole rat"                 "Musk shrew"
## [13] "Tree shrew"
```

```
row.names(mammals)[which(fit.star$cluster==3)]
```

```
##  [1] "Arctic fox"               "Owl monkey"
##  [3] "Mountain beaver"          "Guinea pig"
##  [5] "Verbet"                   "Chinchilla"
##  [7] "Arctic ground squirrel"   "African giant pouched rat"
##  [9] "Nine-banded armadillo"    "Tree hyrax"
## [11] "N.A. opossum"             "European hedgehog"
## [13] "Cat"                      "Genet"
## [15] "Rock hyrax-a"             "Water opossum"
## [17] "Yellow-bellied marmot"    "Slow loris"
```

```
## [19] "Rabbit"                    "Desert hedgehog"
## [21] "Rock hyrax-b"              "Raccoon"
## [23] "Echidna"                   "Tenrec"
## [25] "Phalanger"                 "Red fox"
```

```
plot(mammals.new[,1:2], col=mammals.new[,3], cex=1)
text(mammals.new[,1:2], labels=row.names(mammals), cex= 0.7)
```

Q4

(a)

```
library(ISLR)
library(class)
library(stats)
x = rbind(Khan$xtrain,Khan$xtest)
y = c(Khan$ytrain,Khan$ytest)
dim(x)
```

```
## [1]   83 2308
```

```
length(y)
```

```
## [1] 83
```

```
prc = prcomp(x[,1:2308], center = TRUE, scale. = FALSE)
```

(b)

The second classifier is better. (1) Smaller CE. (2) Curse of dimensionality. In KNN, the more dimensions we add, the larger the volume in the hyperspace needs to be to capture fixed number of neighbors. As the volume grows larger and larger, the "neighbors" become less and less "similar" to the query point as they are now all relatively distant from the query point considering all different dimensions that are included when computing the pairwise distances. (3) Time comlexity. The brute force KNN has $O(n \times m)$. When the number of samples n is much larger than the number of features m, its time complexity will be $O(n)$. (3) The 10 principal components take 64.76% proportion of variance which means 10 principal components are representative to some extend.

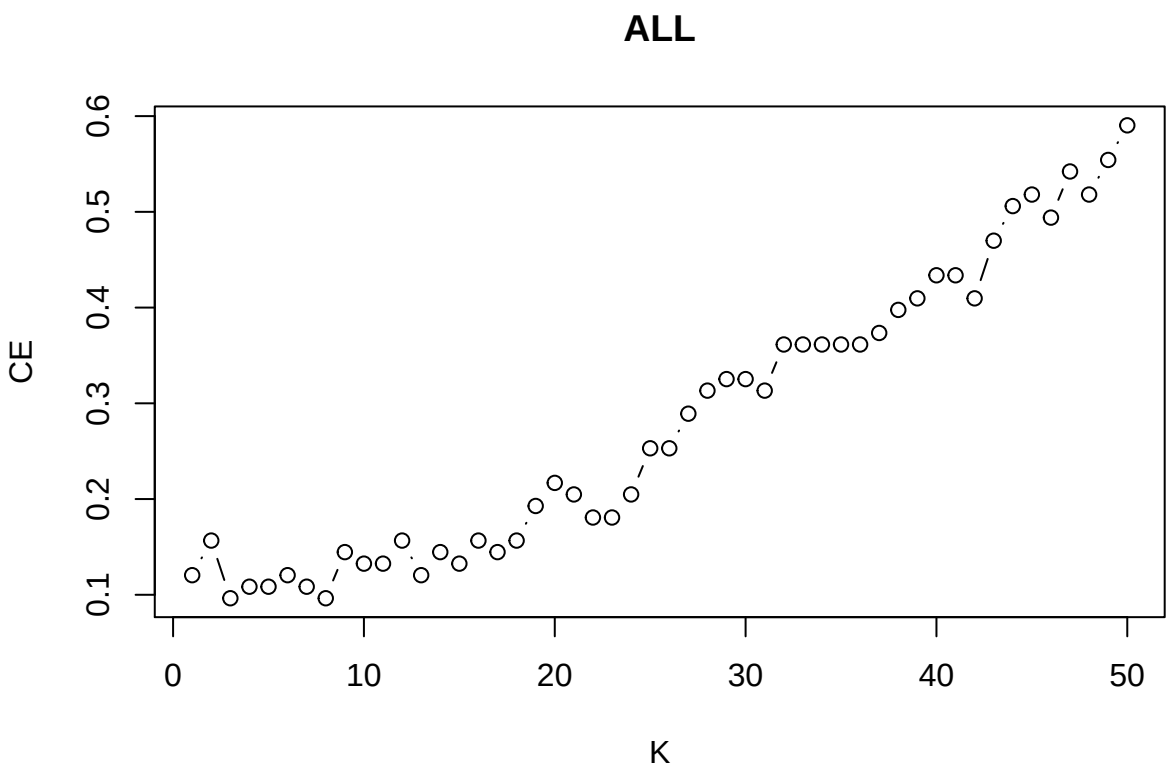
```
# a
fa = function(confusion.table){
  CE = 1-sum(diag(confusion.table))/sum(confusion.table)
  return(CE)
}

knn.function = function(k.list, train, gr) {
  pr.table = matrix(NA,length(k.list),1)
  colnames(pr.table) = "CE"
  for (i in 1:length(k.list)) {
    knn.fit = knn.cv(train,gr,k=k.list[i],use.all=T)
    c_table = table(knn.fit,gr)
    pr.table[i,] = fa(c_table)
  }
  return(pr.table)
}

k.list = seq(1,50,1)
result1 = knn.function(k.list, x, y)
min(result1)
```

```
## [1] 0.09638554
```

```
# plot CE vs K
plot(k.list,result1[,1],main="ALL" ,type="b",xlab="K",ylab="CE")
```



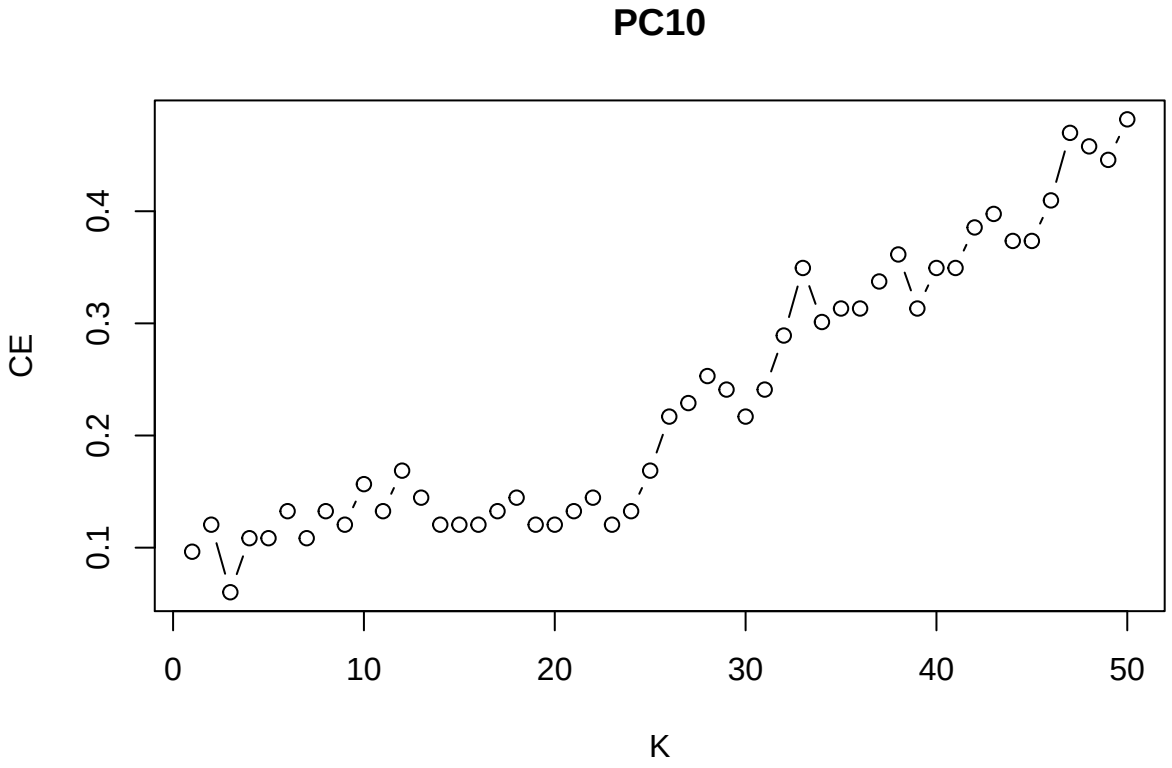

```
# b
prc10 = prcomp(x[,1:2308], center = TRUE, scale. = FALSE, rank=10)
summary(prc10)
```

```
## Importance of first k=10 (out of 83) components:
##                             PC1     PC2      PC3     PC4     PC5    PC6    PC7
## Standard deviation     12.8299 10.5420 10.10813 8.44593 7.58067 7.4190 6.6992
## Proportion of Variance  0.1507  0.1018  0.09356 0.06532 0.05262 0.0504 0.0411
## Cumulative Proportion   0.1507  0.2525  0.34606 0.41138 0.46400 0.5144 0.5555
##                             PC8     PC9    PC10
## Standard deviation     6.24124 5.99965 5.06687
## Proportion of Variance 0.03567 0.03296 0.02351
## Cumulative Proportion  0.59117 0.62413 0.64764
```

```
x.new = prc10$x
result2 = knn.function(k.list, x.new, y)
min(result2)
```

```
## [1] 0.06024096
```

```
# plot CE vs K
plot(k.list,result2[,1],main="PC10", type="b",xlab="K",ylab="CE")
```



Q5

K means clusters data based on distances which are sensitive to measuring unit. The features with high magnitude will give high weight to the distance and the algorithm will be biased. Specifically, K-means requires us to find a solution to maximaze $R^2 = 1 - \frac{SS_{within}}{SS_{total}}$. To calculate SS_{within} and SS_{total}, we need to calculate $d(x, g(A))^2$. For example, if we use Euclidean distance and we have two features. We calculate distance between point (1000, 0.1) to (990, 0.2). It is obvious that the first feature dominates the distance change as it has higher magnitude.

LDA is not affected. LDA finds a set of discriminant axes by computing eigenvectors of $W^{-1}B$, where W and B are within- and between-class scatter matrices. Specifically, it is to find v and λ in $Bv = \lambda Wv$. Suppose we do scaling with a diagnal matrix A. $Bv = \lambda Wv$ is quivalent to t$ABAA^{-1}v = \lambda AWAA^{-1}v$. And the equation will become $B_{new}A^{-1}v = \lambda W_{new}A^{-1}v$ and $A^{-1}v$ is the new eigenvector. However, the eigenvalue λ stay unchanged. This means the classification will not change.