Analytics 512: Take Home Final Exam

Michael Chon

Vehicle Data

```
require(mlbench)
```

```
## Loading required package: mlbench
```

```
data(Vehicle)
```

1

Use a decision tree to predict the vehicle class. Assess your prediction accuracy using a confusion matrix, once from the full data set and also with 10-fold cross validation.

```
library(tree)
library(boot)

set.seed(0923)

## the full data set


model.1 = tree(Class ~., data=Vehicle)          # tree model on the full data set
predict.1 = predict(model.1, type='class')      # perform on the full data set
tab.1 = table(Vehicle$Class, predict.1)         # confusion matrix

print(tab.1)
```

```
##       predict.1
##        bus opel saab van
##   bus  210    4    2   2
##   opel   1   94   95  22
##   saab   3   32  149  33
##   van    2    3    0 194
```

```
print(paste('The prediction accuracy rate is',
            round(sum(diag(tab.1))/sum(tab.1),5)))   # prediction accuracy
```

```
## [1] "The prediction accuracy rate is 0.76478"
```

```
## 10-fold cross validation

## I used the cross-validation code from diary0221


K = 10    # number of fold
N = 846   # number of observations

cvindex = rep(1:K, length=N)
cvindex = sample(cvindex, N, replace=F)
```

```
confusion = matrix(rep(0,4), ncol=4, nrow=4) # create a empty confusion matrix

for (j in 1:K)
{
  testset = cvindex==j
  testdf <- Vehicle[testset,]       # Define the test set
  traindf <- Vehicle[-testset,]     # Build the train set

  tree.fit <- tree(Class ~., data = traindf)   # fit a tree model on the train set
  pred <- predict(tree.fit, newdata = testdf, type = 'class')  # perform on the test set
  confusion <- confusion + table(testdf$Class, pred)   # confusion matrix
}

print(confusion)
```

```
##       pred
##        bus opel saab van
##   bus  210    4    2   2
##   opel   1   94   95  22
##   saab   3   32  149  33
##   van    2    3    0 194
```

```
print(paste('From 10-fold cross validation, the prediction accuracy rate is',
            round(sum(diag(confusion))/sum(confusion),5)))   # prediction accuracy
```

```
## [1] "From 10-fold cross validation, the prediction accuracy rate is 0.76478"
```

2

Now use PCA to find the principal components of the matrix of predictors, scaling these predictors so that all have unit variance before performing the PCA. It is possible to use the first k principal components to predict the vehicle class with a tree. Use 10-fold cross validation to select the best k. Assess the error using the misclassification rate.

```
set.seed(0923)

## 10-fold cross validation

## I used the PCA code from diary0418


Vehicle1 = Vehicle[,-19]    # remove the class from the original dataset
Vehicle.pca = prcomp(as.matrix(Vehicle1), scale = TRUE)   # scaling the predictors
X = Vehicle.pca$x  # pca matrix with 18 predictors


K = 10    # number of fold
N = 846   # number of observations

cvindex = rep(1:K, length=N)
cvindex = sample(cvindex, N, replace=F)

confusion = matrix(rep(0,4), ncol=4, nrow=4)
```

```
pca.fold <- function(K,m)     # create a function that computes the misclassification rate
                              # of tree model of k first principal components using confusion
                              # matrix
{
  X = data.frame(X[,1:m])
  X$Class = Vehicle$Class     # add Class to X
    for(j in 1:K)
    {
      testset = cvindex==j
      testdf <- X[testset,]
      traindf <- X[-testset,]

      tree.fit <- tree(Class ~., data=traindf)   # fit a tree model
      pred <- predict(tree.fit, newdata=testdf, type='class')   # perform on the test
      confusion <- confusion + table(testdf$Class, pred)   # confusion matrix
    }
  return(1 - round(sum(diag(confusion))/sum(confusion),5)) # misclassification rate
}

# tree model of 1 to 18 first principal components

pca.list <- c(pca.fold(10,1), pca.fold(10,2), pca.fold(10,3), pca.fold(10,4),
        pca.fold(10,5), pca.fold(10,6), pca.fold(10,7), pca.fold(10,8),
        pca.fold(10,9), pca.fold(10,10), pca.fold(10,11), pca.fold(10,12),
        pca.fold(10,13), pca.fold(10,14), pca.fold(10,15), pca.fold(10,16),
        pca.fold(10,17), pca.fold(10,18))

print('The misclassification rates are listed from k first principals components = 1 to 18:')
```

```
## [1] "The misclassification rates are listed from k first principals components = 1 to 18:"
```

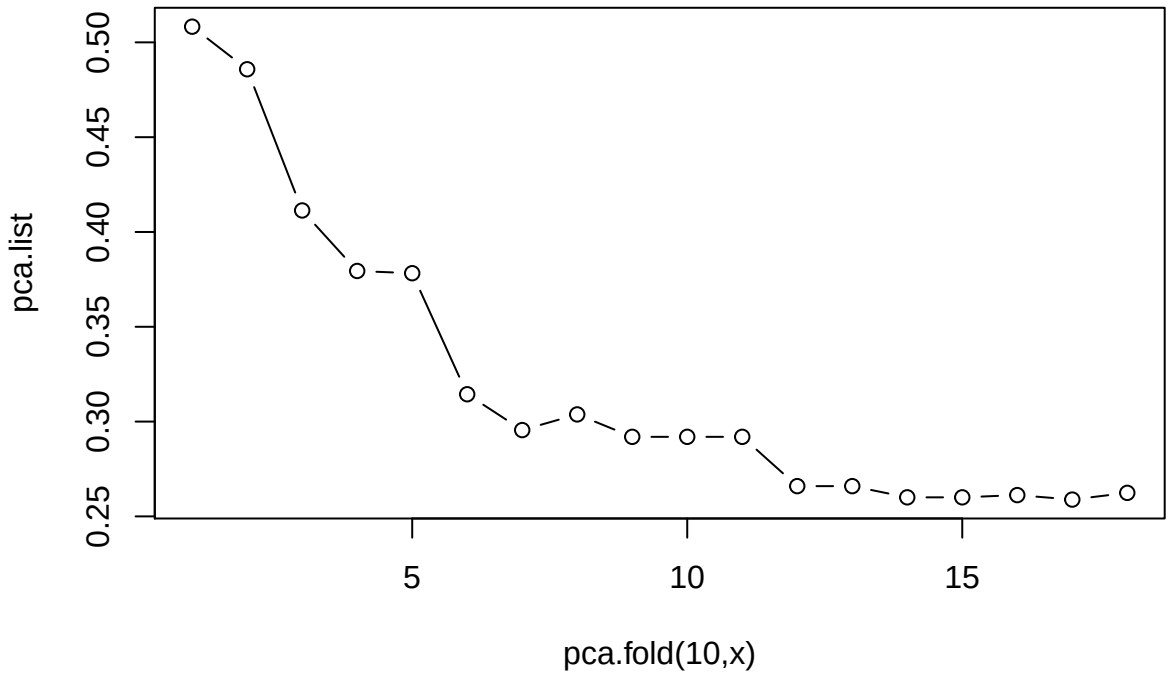
```
print(pca.list)
```

```
##  [1] 0.50827 0.48582 0.41135 0.37943 0.37825 0.31442 0.29551 0.30378
##  [9] 0.29196 0.29196 0.29196 0.26596 0.26596 0.26005 0.26005 0.26123
## [17] 0.25887 0.26241
```

```
print(paste('From 10-fold cross validation, the best k is', which.min(pca.list)))
```

```
## [1] "From 10-fold cross validation, the best k is 17"
```

```
plot(pca.list, xlab = 'pca.fold(10,x)', type = 'b')
```



3

Now use a random forest with the original dataset (not principal components). Predict the vehicle class with a random forest model where trees with up to m terminal nodes are used. The parameter m can be set with the maxnodes parameter. Use 10-fold cross validation to find the best m. Assess the error using the misclassification rate.

```
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
set.seed(0923)

## 10-fold cross validation


K = 10
N = 846

cvindex = rep(1:K, length=N)
cvindex = sample(cvindex, N, replace=F)

confusion = matrix(rep(0,4), ncol=4, nrow=4)


rand.fold <- function(K, m)    # create a function that computes the misclassification rate
                               # of a random forest model of m maxnodes using confusion
                               # matrix
{
    for (j in 1:K)
    {
```

```
      testset = cvindex==j
      testdf <- Vehicle[testset,]
      traindf <- Vehicle[-testset,]

      rf.fit <- randomForest(Class ~., data=traindf, mtry=sqrt(18),
                             maxnodes=m)    # fit a random forest model
      pred <- predict(rf.fit, newdata=testdf, type='class')   # perform on the test set
      confusion <- confusion + table(testdf$Class, pred)   # confusion matrix
    }
  return(1 - round(sum(diag(confusion))/sum(confusion),5))  # the misclassification rate
}

# random forest model of 2 to 10 maxnodes

rand.list <- c(NA, rand.fold(10,2), rand.fold(10,3), rand.fold(10,4),rand.fold(10,5),
        rand.fold(10,6), rand.fold(10,7), rand.fold(10,8), rand.fold(10,9),
        rand.fold(10,10))

print('The misclassification rates are listed from maxnodes = 2 to 10:')
```

```
## [1] "The misclassification rates are listed from maxnodes = 2 to 10:"
```

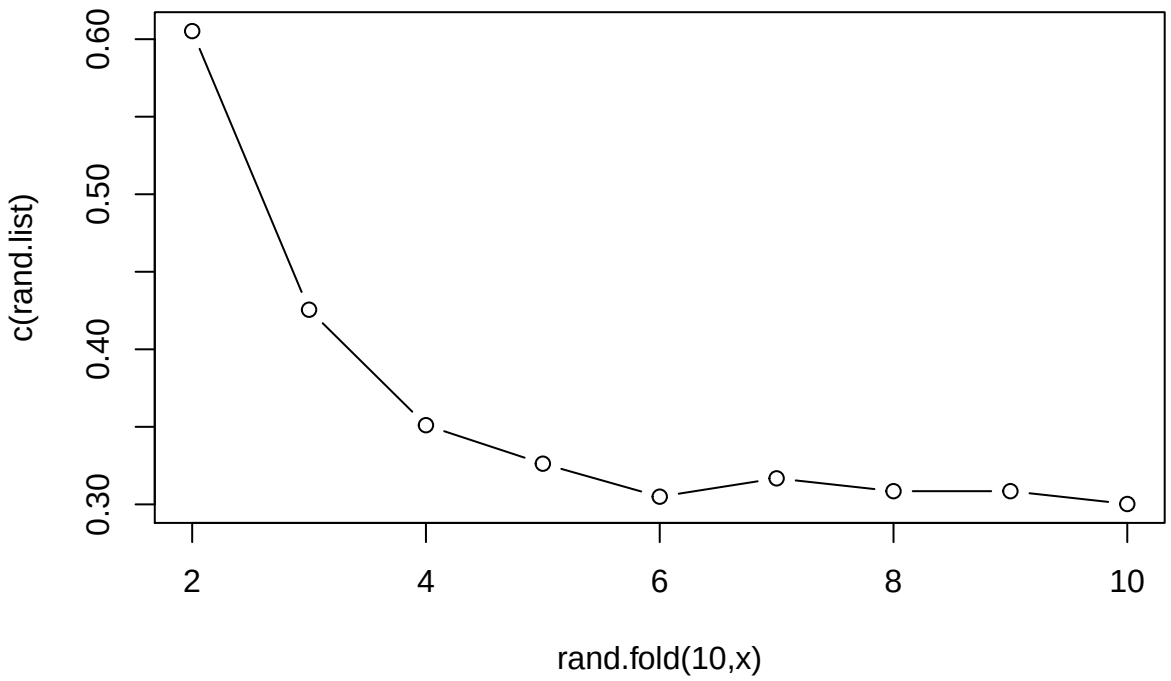
```
print(rand.list)
```

```
##  [1]      NA 0.60520 0.42553 0.35106 0.32624 0.30496 0.31678 0.30851
##  [9] 0.30851 0.30024
```

```
print(paste('From 10-fold cross validation, the best m is', which.min(rand.list)))
```

```
## [1] "From 10-fold cross validation, the best m is 10"
```

```
plot(c(rand.list) , xlim = c(2,10), type = 'b', xlab = 'rand.fold(10,x)')
```



4

It turns out that the two classes Opel and Saab are difficult to separate. Make a subset of vehicle data of Saabs and Opels only and predict the vehicle class using logistic regression and five predictors. You have to find a way to choose five predictors and explain your reasoning. Possible approaches for choosing predictors: run logistic regression for the whole set of predictors and discard the non-significant ones; or use regsubsets (choosing one of the possible methods); or compute correlations between all predictors and eliminate those that are highly correlated; or use regularized regression such as LASSO; or use a pruned tree. Evaluate the final model of your choice using 10-fold cross validation. Summarize the results of this exploration in a paragraph, using tables or graphs as you see fit.

```
set.seed(0923)

## a subset of vehicle data of Saabs and Opels only


subVehicle <- subset(Vehicle, Vehicle$Class == 'opel' | Vehicle$Class == 'saab')
glm.fit <- glm(Class ~., data = subVehicle, family = 'binomial')
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Class ~ ., family = "binomial", data = subVehicle)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2457  -0.9217   0.2448   0.9109   1.9909
##
## Coefficients:
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -19.511180  35.712680  -0.546   0.5848
## Comp           0.231124   0.035371   6.534 6.39e-11 ***
## Circ          -0.623145   0.154635  -4.030 5.58e-05 ***
## D.Circ         0.026696   0.029418   0.907   0.3642
## Rad.Ra        -0.008226   0.031425  -0.262   0.7935
## Pr.Axis.Ra     0.079992   0.096481   0.829   0.4071
## Max.L.Ra      -0.159308   0.150522  -1.058   0.2899
## Scat.Ra        0.087698   0.186631   0.470   0.6384
## Elong         -0.161654   0.236048  -0.685   0.4934
## Pr.Axis.Rect   0.588601   0.398954   1.475   0.1401
## Max.L.Rect     0.092930   0.054149   1.716   0.0861 .
## Sc.Var.Maxis  -0.028854   0.033348  -0.865   0.3869
## Sc.Var.maxis  -0.038950   0.027133  -1.436   0.1511
## Ra.Gyr         0.059873   0.012692   4.717 2.39e-06 ***
## Skew.Maxis     0.066892   0.070076   0.955   0.3398
## Skew.maxis     0.020320   0.026172   0.776   0.4375
## Kurt.maxis    -0.038468   0.016309  -2.359   0.0183 *
## Kurt.Maxis    -0.495084   0.117139  -4.226 2.37e-05 ***
## Holl.Ra        0.452459   0.099643   4.541 5.60e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 594.66  on 428  degrees of freedom
```

```
## Residual deviance: 469.68  on 410  degrees of freedom
## AIC: 507.68
##
## Number of Fisher Scoring iterations: 4
```

Based on the summary of the logistic regression fitted on the subVehicle above, the five most statistically significant predictors in the logistic regression are Holl.Ra, Kurt.Maxis, Ra.Gyr, Circ, and Comp because their p-values are extremely low. The two other predictors, Max.L.Rect and Kurt.maxis, are somewhat statistically significant but their p-values are not as low as the other five predictors mentioned before.

```
set.seed(0923)

## 10-fold cross validation
## containing only the five predictors stated above


subVehicle1 = subVehicle[c('Holl.Ra', 'Kurt.Maxis', 'Ra.Gyr', 'Circ', 'Comp')]
subVehicle1$Class = subVehicle$Class


K = 10      # the number of fold
N1 = 429    # the number of observations of the subset

cvindex1 = rep(1:K, length = N1)
cvindex1 = sample(cvindex1, N1, replace = F)

confusion1 = matrix(rep(0,2), ncol = 4, nrow = 2)


for (j in 1:10)
{
    testset1 = cvindex1 == j
    testdf1 <- subVehicle1[testset1,]
    traindf1 <- subVehicle1[-testset1,]

    glm.fit1 <- glm(Class ~., data = traindf1, family = 'binomial')
    prob <- predict(glm.fit1, newdata=testdf1, type = 'response')

    pred <- rep("down", length(prob))
    pred[prob >.5] = "up"

    tab <- table(pred,testdf1$Class)
    confusion1 <- confusion1 + tab
}
confusion1 <- confusion1[,2:3]

print(confusion1)
```

```
##
## pred   opel saab
##   down  144   71
##   up     68  146
```

```
print(round(sum(diag(confusion1))/sum(confusion1),5))
```

```
## [1] 0.67599
```

After dropping non-significant predictors in the dataset, logistic regression has been fitted using 10-fold cross validation. As seen above, according to the confusion matrix, the classification rate is .67599, which is not very high. 71 of 217 saabs are misclassified as opel and 68 of 212 opels are misclassified as saab. It seems that it is still difficult to separate Opel and Saab using logistic regression with the five most significant predictors in the dataset.

Smiley Data

5

Generate four different smiley data sets with n = 500 points each that have different values of standard deviations for the eyes and the mouth (these are the sd1 and sd2 arguments). Plot them with colors given by the target class labels.

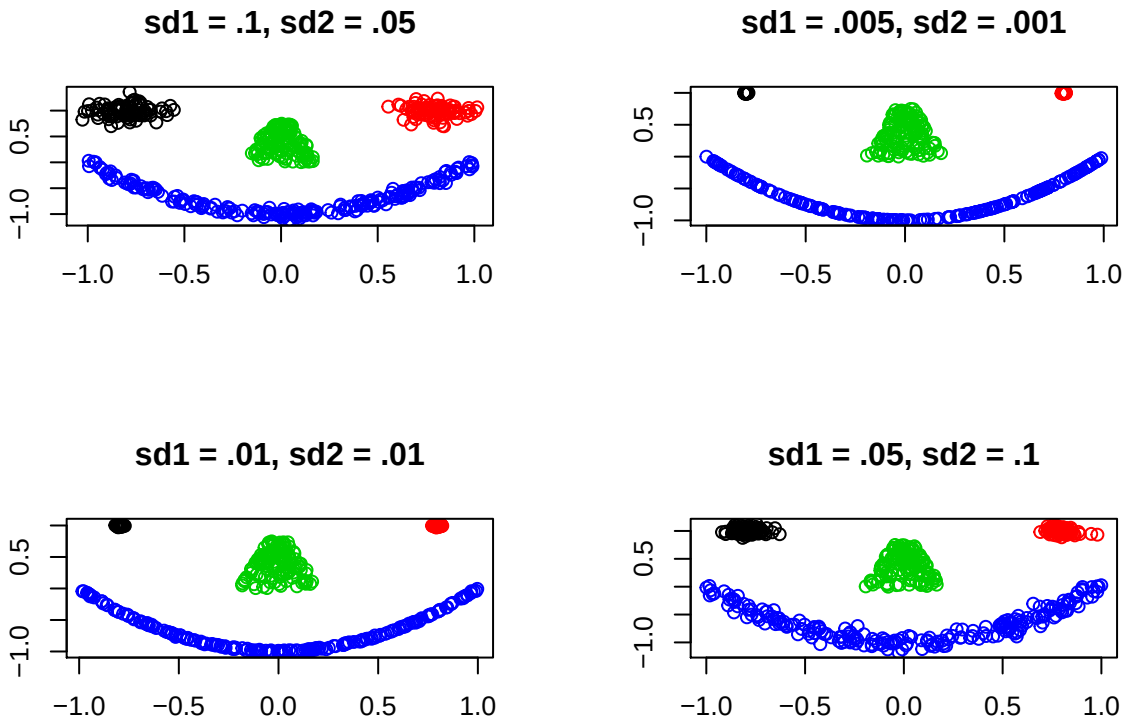
```
par(mfrow=c(2,2))

p1 <- mlbench.smiley(n = 500, sd1 = .1, sd2 = .05)      # default sd1, sd2
plot(p1, main = 'sd1 = .1, sd2 = .05')

p2 <- mlbench.smiley(n = 500, sd1 = .005, sd2 = .001)   # sd1 = .005, sd2 = .001
plot(p2, main = 'sd1 = .005, sd2 = .001')

p3 <- mlbench.smiley(n = 500, sd1 = .01, sd2 = .01)     # sd1 = .01, sd2 = .01
plot(p3, main = 'sd1 = .01, sd2 = .01')

p4 <- mlbench.smiley(n = 500, sd1 = .05, sd2 = .1)      # sd1 = .05, sd2 = .1
plot(p4, main = 'sd1 = .05, sd2 = .1')
```



6

Next, create a series of Smiley data sets in the following way: use sd1 = .1 and let sd2 range from 0.05 to 0.5. For each of these, use k-means clustering with k = 4 to cluster smiley data. Explore for which values of sd2 the clusters coincide (more or less) with the target class labels. Summarize the result of this exploration in a paragraph or two, using tables or graphs as you see fit.

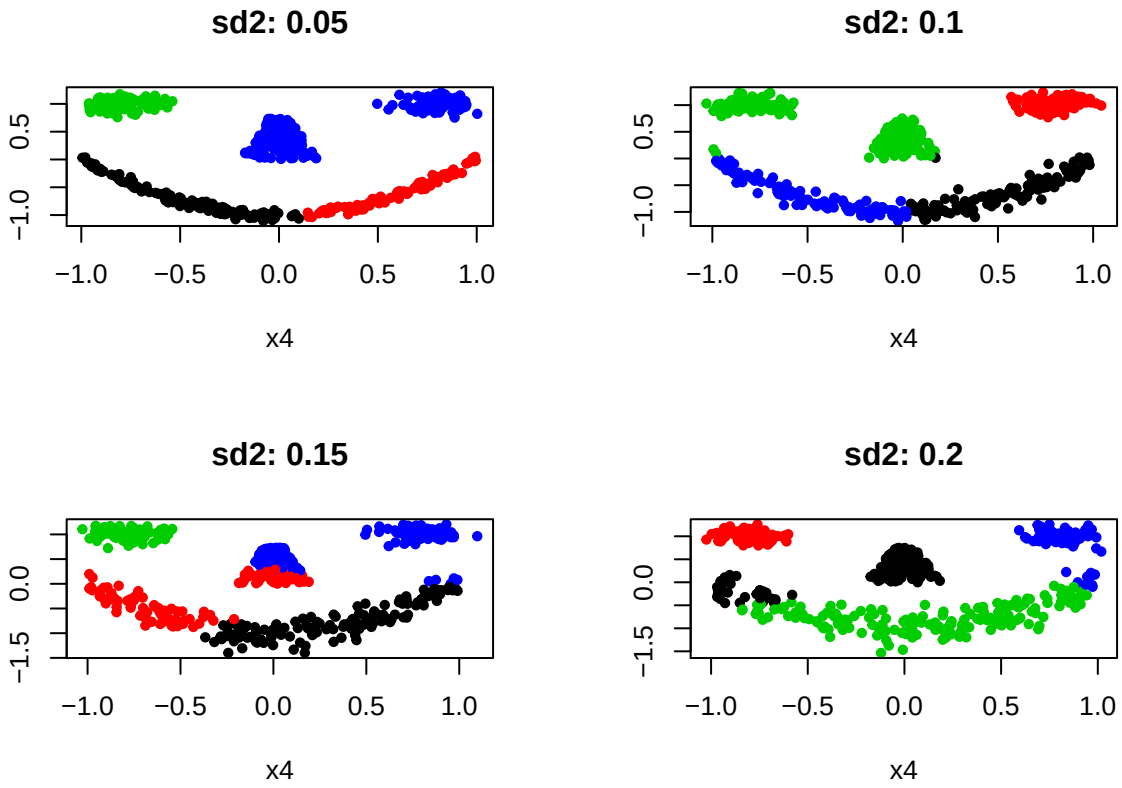
```
set.seed(6)

library(deldir)
```

```
## deldir 0.1-14
```

```
seq = seq(0.05, .2, .05)     # sd2 from 0.05 to .2 with an increase of .05 each
par(mfrow=c(2,2))

for(i in 1:4){
  p <- mlbench.smiley(n = 500, sd1 = .1, sd2 = seq[i])$x  # mlbench.smiley input value
  kmeans.p <- kmeans(p, 4)                                # k means clustering
  plot(p, col = kmeans.p$cluster, pch = 19, cex = .7,
       main = paste('sd2:', seq[i]))
  i = i+1
}
```

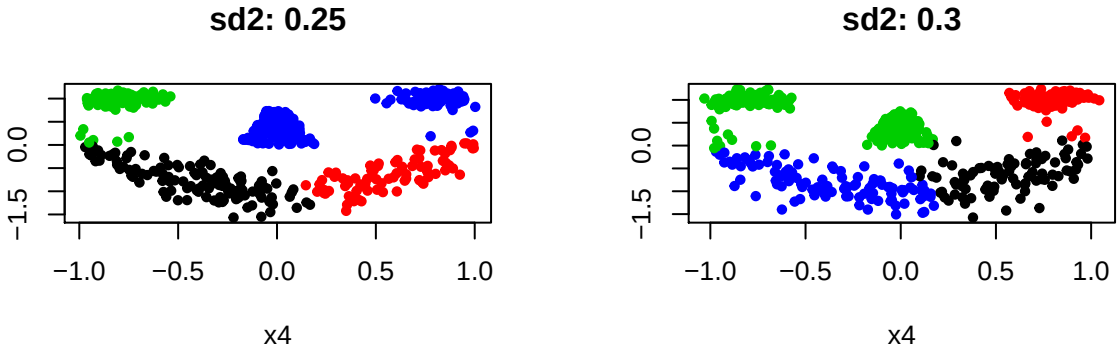


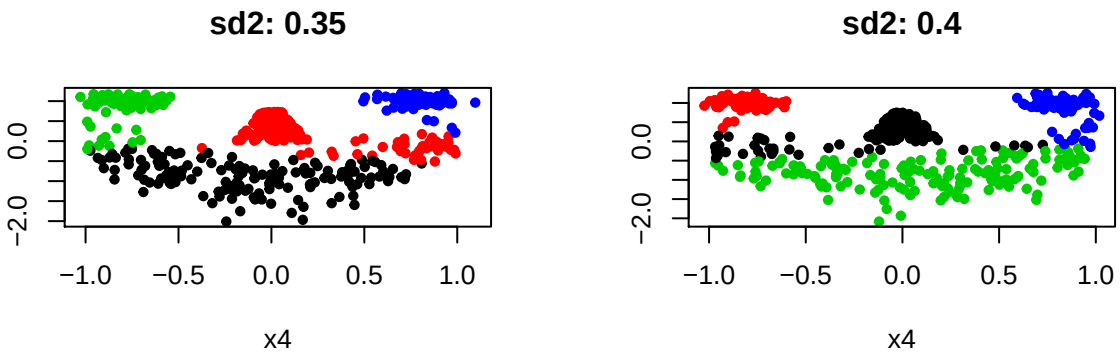

```
## same as above but different sd2 values
## here sd2 = .25 to .45 by .05

set.seed(6)

seq = seq(0.25, .45, .05)
par(mfrow=c(2,2))
```

```
for(i in 1:4){
  p <- mlbench.smiley(n = 500, sd1 = .1, sd2 = seq[i])$x
  kmeans.p <- kmeans(p, 4)
  plot(p, col = kmeans.p$cluster, pch = 19, cex = .7,
       main = paste('sd2:', seq[i]))
  i = i+1
}
```



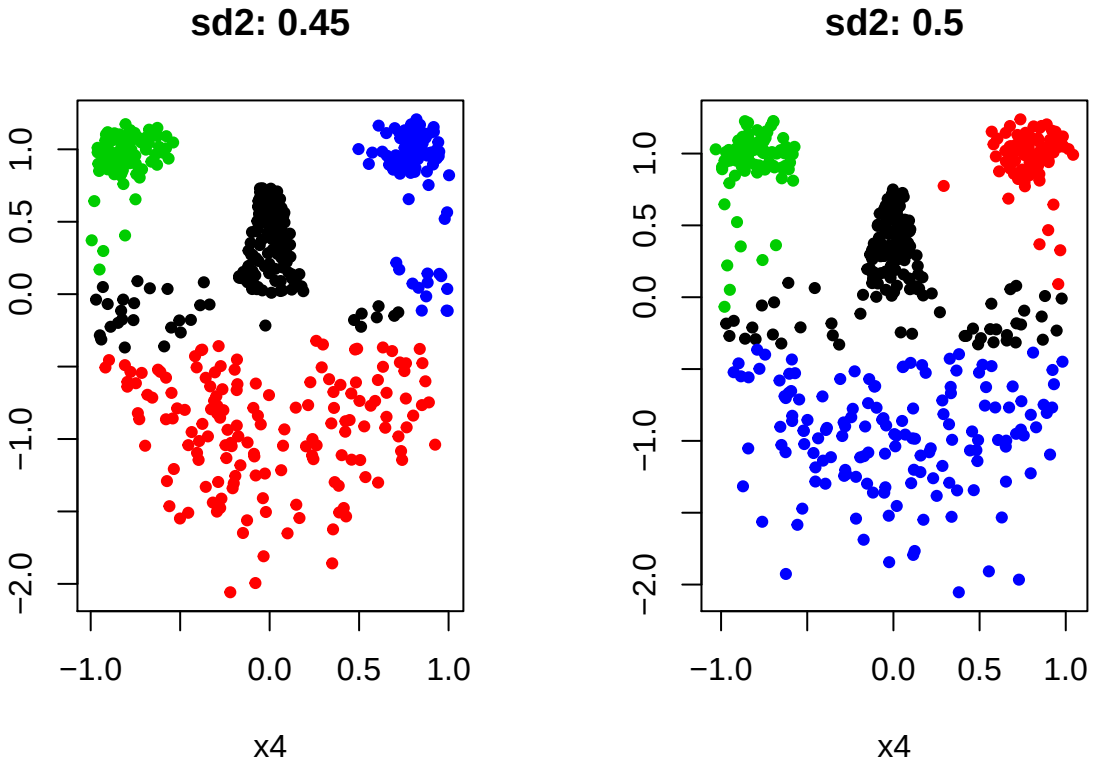



```
## same as above but different sd2 values
## here sd2 = .45 to .5 by .05

set.seed(6)

seq = seq(.45, .5, .05)
par(mfrow=c(1,2))

for(i in 1:2){
  p <- mlbench.smiley(n = 500, sd1 = .1, sd2 = seq[i])$x
  kmeans.p <- kmeans(p, 4, nstart = 1)
  plot(p, col = kmeans.p$cluster, pch = 19, cex = .7,
       main = paste('sd2:', seq[i]))
  i = i+1
}
```



The observation among the 10 plots above is that as sd2 increases from 0.05 to 0.5, the points of the mouths in the plots spread out. It is because sd2 means the standard deviation for the mouth and as the standard deviation of the mouth increases, the points get more dispersed throughout the plot. So, the plots of sd2 = 0.05 and .1 are very easy to tell the smiley faces. However, as sd2 increases to 0.5, it becomes difficult to tell whether the supposedly mouth is a mouth because the points are scattered.

For the values of sd2 from 0.05 to 0.3, the clusters of the plots do not coincide much with the target class labels because the mouths are splited in half with two different colors. However, for the values of sd2 from 0.35 to .5, the clusters of the plots coincide more with the target class labels, but not completely, because the tips of the mouths are not colored in different colors. For instance, in the plot of sd2 = 0.45, the tips of the mouths are colored in black and blue but the right color of the mouth is red.

7

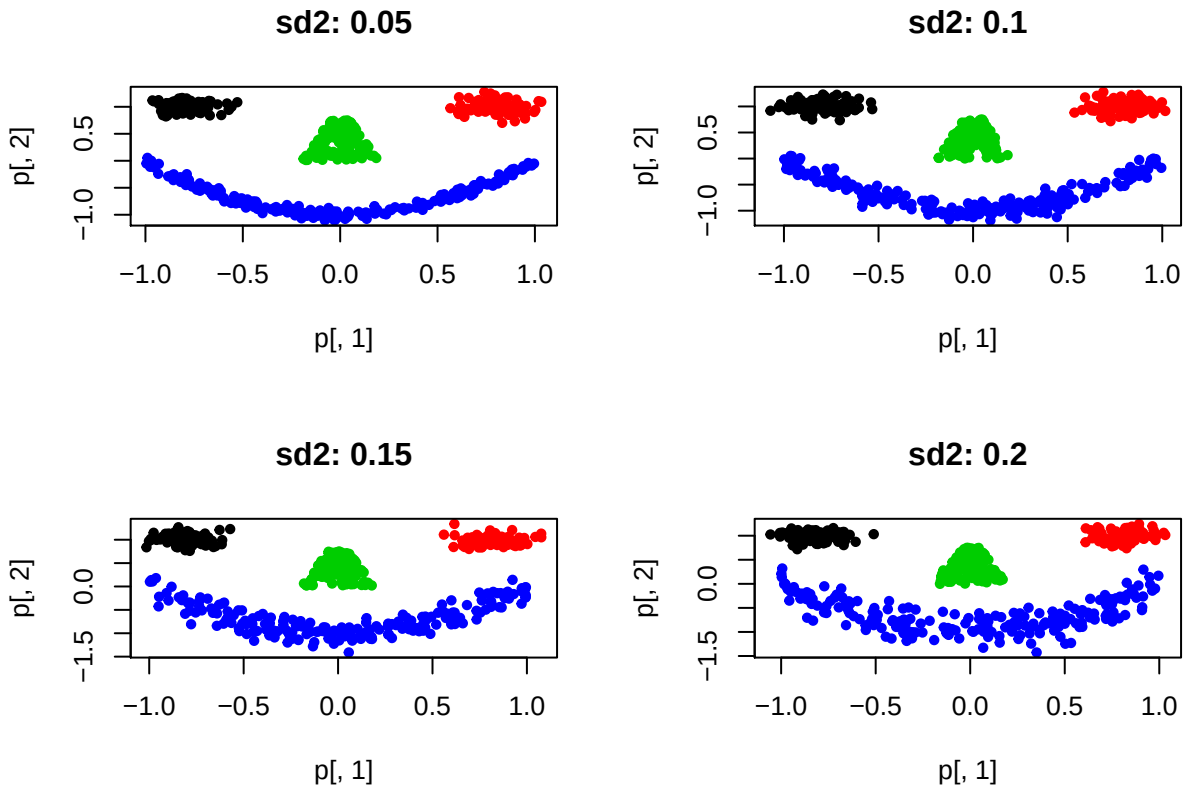
With the same series of Smiley data sets, use hierarchical clustering, followed by cutting the tree to k = 4 clusters for the smiley data. Use a linkage method of your choice. Explore for which values of sd2 the clusters coincide (more or less) with the target class labels. Summarize the result of this exploration in a paragraph or two, using tables or graphs as you see fit.

```
set.seed(7)

seq = seq(0.05, .2, .05) # sd2 from .05 to .2
par(mfrow=c(2,2))

for(i in 1:4){
  p <- mlbench.smiley(n = 500, sd1 = .1, sd2 = seq[i])$x    # mlbench.smiley input values
  pdist <- dist(p)                                          # turn p into distance
  hclust.p <- hclust(pdist, method = 'single')              # hierachical clustering with
                                                            # single linkage method
  hclust.cut <- cutree(hclust.p, 4)                         # cutting the tree to k = 4
```

```
  plot(p[,1], p[,2], col = hclust.cut, pch = 19, cex = .7,
       main = paste('sd2:', seq[i]))                          # plot
  i = i+1
}
```

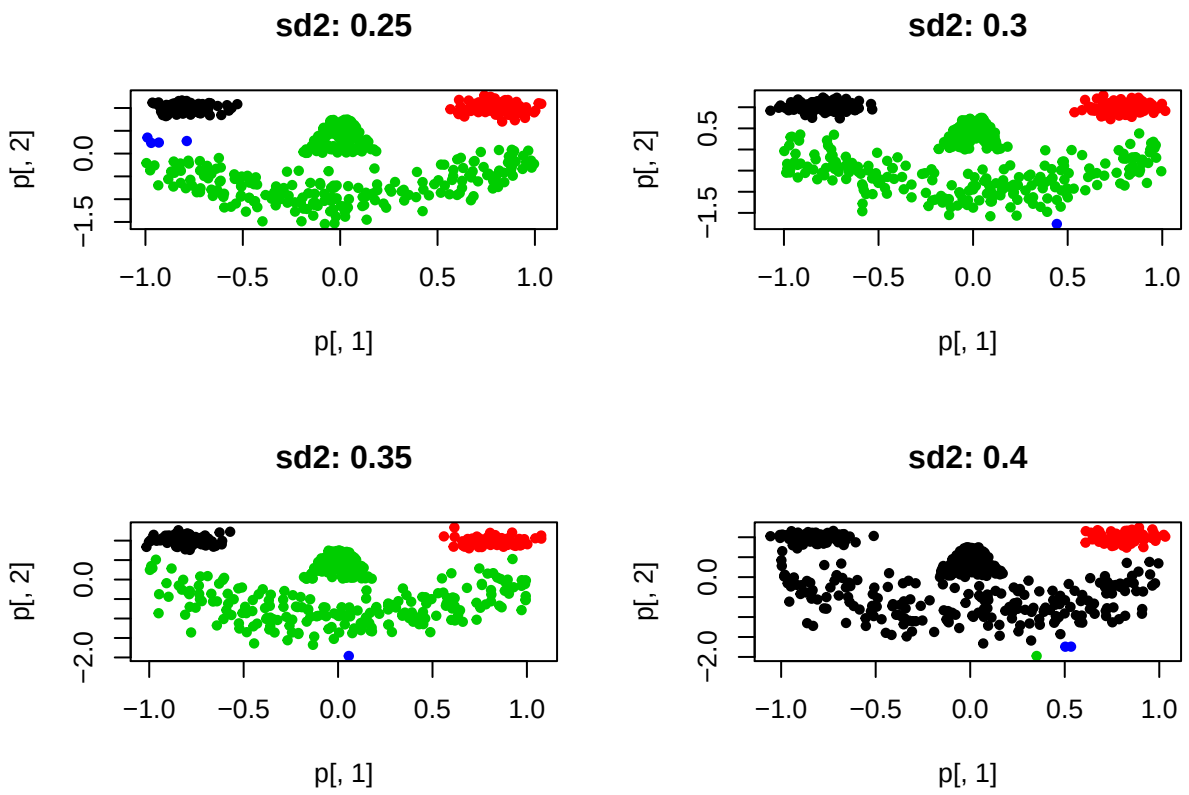



```
## same as above but different sd2 values
## here sd2 = .25 to .45 by .05

set.seed(7)

seq = seq(0.25, .45, .05)
par(mfrow=c(2,2))

for(i in 1:4){
  p <- mlbench.smiley(n = 500, sd1 = .1, sd2 = seq[i])$x
  pdist <- dist(p)
  hclust.p <- hclust(pdist, method = 'single')
  hclust.cut <- cutree(hclust.p, 4)
  plot(p[,1], p[,2], col = hclust.cut, pch = 19, cex = .7,
       main = paste('sd2:', seq[i]))
  i = i+1
}
```

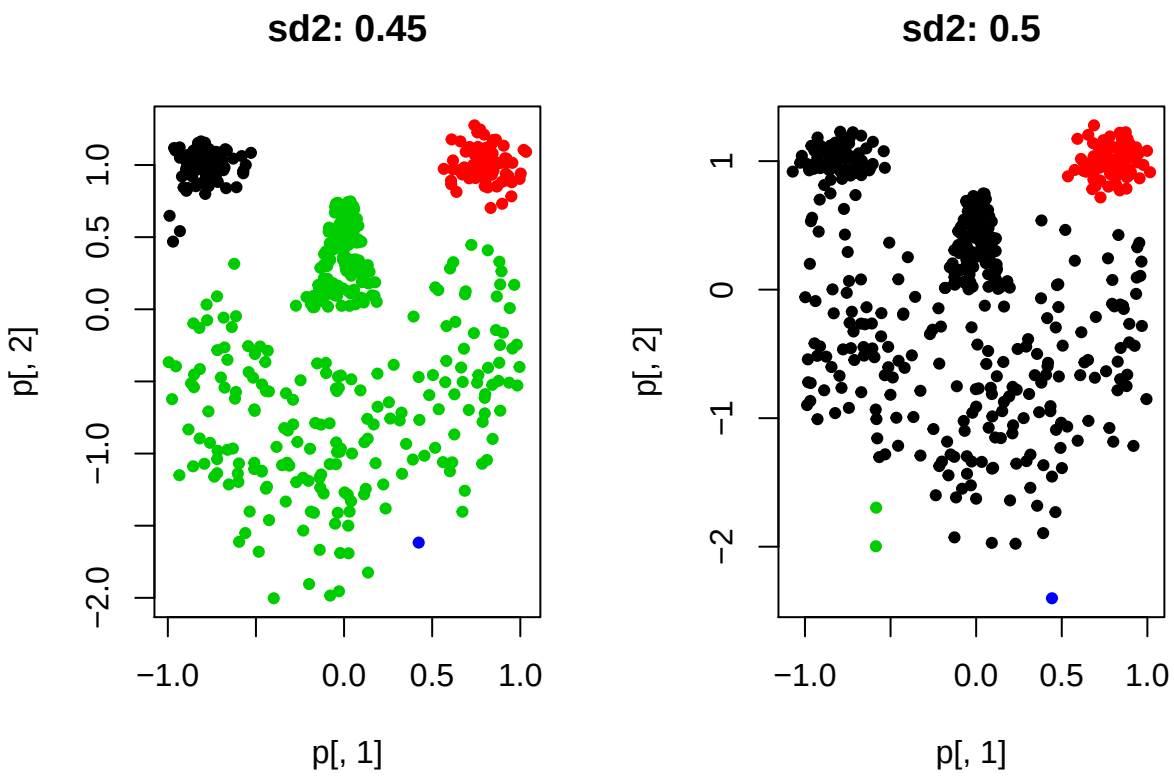



```
## same as above but different sd2 values
## here sd2 = .45 to .5 by .05

set.seed(7)

seq = seq(0.45, .5, .05)
par(mfrow=c(1,2))

for(i in 1:2){
  p <- mlbench.smiley(n = 500, sd1 = .1, sd2 = seq[i])$x
  pdist <- dist(p)
  hclust.p <- hclust(pdist, method = 'single')
  hclust.cut <- cutree(hclust.p, 4)
  plot(p[,1], p[,2], col = hclust.cut, pch = 19, cex = .7,
       main = paste('sd2:', seq[i]))
  i = i+1
}
```



Interestingly, the 10 plots produced above show quite an opposite result compared to the 10 plots from the question #6. First of all, the hierarchical clustering with 'single' method perfectly coincided with the target class labels. For the values of sd2 = 0.05 to 0.2, each eye, nose, and mouth are correctly classified as cluster with the same colors used to classify the clusters in the plot of a regular mlbench.smiley() function.

However, as sd2 increases from 0.25 to 0.5 and the points of the mouth become more dispersed, the hierarchical clustering with 'single' method does not work very well because a color that distinguishes the mouth also classifies the nose. For example, in the plots of sd2 = 0.25, 0.3, and 0.35, the green color classifies the nose and the mouth but it should not. For the plots of sd2 = 0.5, the black color classifies the eye, the nose and the mouth, the red color clusters one of the two eyes and the rest of the colors represent only 3 points. So, the hierarchical clustering with 'single' method works well when the points are not dispersed.