BDA Coursework 2

MSc Data Science

Mark Rotchell, 13181875

1. Bayesian Networks and Naïve Bayes Classifiers

1.a) Conditional Probability Tables

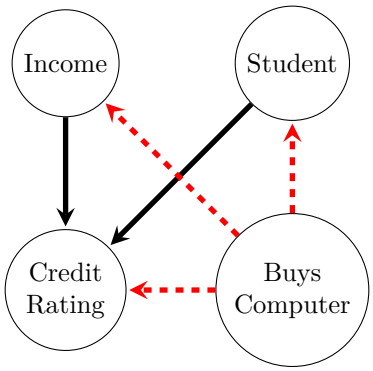
Buy Computer = No	Buy Computer = Yes
16/30 = 0.533333...	14/30 = 0.466666...

	Student = False	Student = True
Buy Computer = No	5/16 = 0.3125	11/16 = 0.6875
Buy Computer = Yes	7/14 = 0.5	7/14 = 0.5

	Income = High	Income = Low
Buy Computer = No	7/16 = 0.4375	9/16 = 0.5625
Buy Computer = Yes	5/14 = 0.357143...	9/14 = 0.642857...

			Credit Rating = Excellent	Credit Rating = Fair
Income = High	Student = False	Buy Computer = No	2/3 = 0.6666...	1/3 = 0.3333...
Income = High	Student = False	Buy Computer = Yes	2/3 = 0.6666...	1/3 = 0.3333...
Income = High	Student = True	Buy Computer = No	2/4 = 0.5	2/4 = 0.5
Income = High	Student = True	Buy Computer = Yes	1/2 = 0.5	1/2 = 0.5
Income = Low	Student = False	Buy Computer = No	1/2 = 0.5	1/2 = 0.5
Income = Low	Student = False	Buy Computer = Yes	2/4 = 0.5	2/4 = 0.5
Income = Low	Student = True	Buy Computer = No	2/7 = 0.285714...	5/7 = 0.714286...
Income = Low	Student = True	Buy Computer = Yes	2/5 = 0.4	3/5 = 0.6

1.a) Bayesian Network Classifier prediction



From above graph we can deduce that:

$$P(Buys\ Computer|Income, Student, Credit\ Rating) \propto P(Credit\ Rating|Income, Student, Buys\ Computer) \cdot P(Income|Buys\ Computer) \cdot P(Student|Buys\ Computer) \cdot P(Buys\ Computer)$$

For testing Instance_31:

$$\begin{aligned}
&P(Buys\ Computer = \text{Yes}|Income = \text{Low}, Student = \text{False}, Credit\ Rating = \text{Excellent}) \\
\propto &P(Credit\ Rating = \text{Excellent}|Income = \text{Low}, Student = \text{False}, Buys\ Computer = \text{Yes}) \\
&\cdot P(Income = \text{Low}|Buys\ Computer = \text{Yes}) \\
&\cdot P(Student = \text{False}|Buys\ Computer = \text{Yes}) \\
&\cdot P(Buys\ Computer = \text{Yes}) \\
= &\frac{1}{2} \cdot \frac{9}{14} \cdot \frac{1}{2} \cdot \frac{7}{15} = \frac{3}{40} = 0.075
\end{aligned}$$

$$\begin{aligned}
&P(Buys\ Computer = \text{No}|Income = \text{Low}, Student = \text{False}, Credit\ Rating = \text{Excellent}) \\
\propto &P(Credit\ Rating = \text{Excellent}|Income = \text{Low}, Student = \text{False}, Buys\ Computer = \text{No}) \\
&\cdot P(Income = \text{Low}|Buys\ Computer = \text{No}) \\
&\cdot P(Student = \text{False}|Buys\ Computer = \text{No}) \\
&\cdot P(Buys\ Computer = \text{No}) \\
= &\frac{1}{2} \cdot \frac{9}{16} \cdot \frac{5}{16} \cdot \frac{8}{15} = \frac{3}{64} = 0.046875
\end{aligned}$$

The predicted class label is **Yes** for testing Instance_31 because

$$P(Buys\ Computer = \text{Yes}|Instance\_31) > P(Buys\ Computer = \text{No}|Instance\_31)$$
$$\frac{3}{40} > \frac{3}{64} \qquad \Leftrightarrow \qquad 0.075 > 0.046875$$

For testing Instance_32:

Following the same logic as above...

$$P(Buys\ Computer = \text{Yes}|Income = \text{High}, Student = \text{False}, Credit\ Rating = \text{Fair}) \propto \frac{1}{3} \cdot \frac{5}{14} \cdot \frac{1}{2} \cdot \frac{7}{15} = \frac{1}{36} = 0.02777\ldots$$
$$P(Buys\ Computer = \text{No}|Income = \text{High}, Student = \text{False}, Credit\ Rating = \text{Fair}) \propto \frac{1}{3} \cdot \frac{7}{16} \cdot \frac{5}{16} \cdot \frac{8}{15} = \frac{7}{288} = 0.02430555\ldots$$

The predicted class label is **Yes** for testing Instance_32 because

$$P(Buys\ Computer = \text{Yes}|Instance\_32) > P(Buys\ Computer = \text{No}|Instance\_32)$$
$$\frac{1}{36} > \frac{7}{288} \qquad \Leftrightarrow \qquad 0.02777\ldots > 0.02430555\ldots$$

1.c) Conditional Probability Tables assuming independent predictors

Buy Computer = No	Buy Computer = Yes
16/30 = 0.533333...	14/30 = 0.466666...

	Student = False	Student = True
Buy Computer = No	5/16 = 0.3125	11/16 = 0.6875
Buy Computer = Yes	7/14 = 0.5	7/14 = 0.5

	Income = High	Income = Low
Buy Computer = No	7/16 = 0.4375	9/16 = 0.5625
Buy Computer = Yes	5/14 = 0.357143...	9/14 = 0.642857...

	Credit Rating = Excellent	Credit Rating = Fair
Buy Computer = No	7/16 = 0.4375	9/16 = 0.5625
Buy Computer = Yes	7/14 = 0.5	7/14 = 0.5

1.d) Naïve Bayes classifier predictions

Under the naïve Bayes model

$$\begin{aligned}P(Buys\ Computer|Income, Student, Credit\ Rating) \propto & P(Credit\ Rating|Buys\ Computer) \\ & \cdot P(Income|Buys\ Computer) \cdot P(Student|Buys\ Computer) \\ & \cdot P(Buys\ Computer)\end{aligned}$$

For testing Instance_31:

$$P(Buys\ Computer = \text{Yes}|Income = \text{Low}, Student = \text{False}, Credit\ Rating = \text{Excellent}) \propto \frac{1}{2} \cdot \frac{9}{14} \cdot \frac{1}{2} \cdot \frac{7}{15} = \frac{3}{40} = 0.075$$

$$\begin{aligned}P(Buys\ Computer = \text{No}|Income = \text{Low}, Student = \text{False}, Credit\ Rating = \text{Excellent}) \propto & \frac{7}{16} \cdot \frac{9}{16} \cdot \frac{5}{16} \cdot \frac{8}{15} \\ = & \frac{21}{512} = 0.041015625\end{aligned}$$

The predicted class label is **Yes** for testing Instance_31 because

$$P(Buys\ Computer = \text{Yes}|Instance\_31) > P(Buys\ Computer = \text{No}|Instance\_31)$$

$$\frac{3}{40} > \frac{21}{512} \qquad \Leftrightarrow \qquad 0.075 > 0.041015625$$

For testing Instance_32:

$$P(Buys\ Computer = \text{Yes}|Income = \text{High}, Student = \text{False}, Credit\ Rating = \text{Fair}) \propto \frac{1}{2} \cdot \frac{5}{14} \cdot \frac{1}{2} \cdot \frac{7}{15} = \frac{1}{24} = 0.041666\ldots$$

$$\begin{aligned}P(Buys\ Computer = \text{No}|Income = \text{High}, Student = \text{False}, Credit\ Rating = \text{Fair}) \propto & \frac{9}{16} \cdot \frac{7}{16} \cdot \frac{5}{16} \cdot \frac{8}{15} \\ = & \frac{21}{512} = 0.041015625\ldots\end{aligned}$$

The predicted class label is **Yes** for testing Instance_32 because

$$P(Buys\ Computer = \text{Yes}|Instance\_32) > P(Buys\ Computer = \text{No}|Instance\_32)$$

$$\frac{1}{24} > \frac{21}{512} \qquad \Leftrightarrow \qquad 0.041666\ldots > 0.041015625\ldots$$

2. Decision Trees and Random Forests

2.a) Train a tree on occupancy data

Load Data

```
train = read.csv('RoomOccupancy_Training.txt')
test = read.csv('RoomOccupancy_Testing.txt')
train$Occupancy = as.factor(train$Occupancy)
test$Occupancy = as.factor(test$Occupancy)
```

Build a tree model

```
library(tree)
tree_model <- tree(Occupancy~.,train)
summary(tree_model)
```

```
##
## Classification tree:
## tree(formula = Occupancy ~ ., data = train)
## Variables actually used in tree construction:
## [1] "Light"       "Temperature" "CO2"         "Humidity"
## Number of terminal nodes:  6
## Residual mean deviance:  0.07156 = 142.7 / 1994
## Misclassification error rate: 0.0125 = 25 / 2000
```

Evaluation Predictive performance

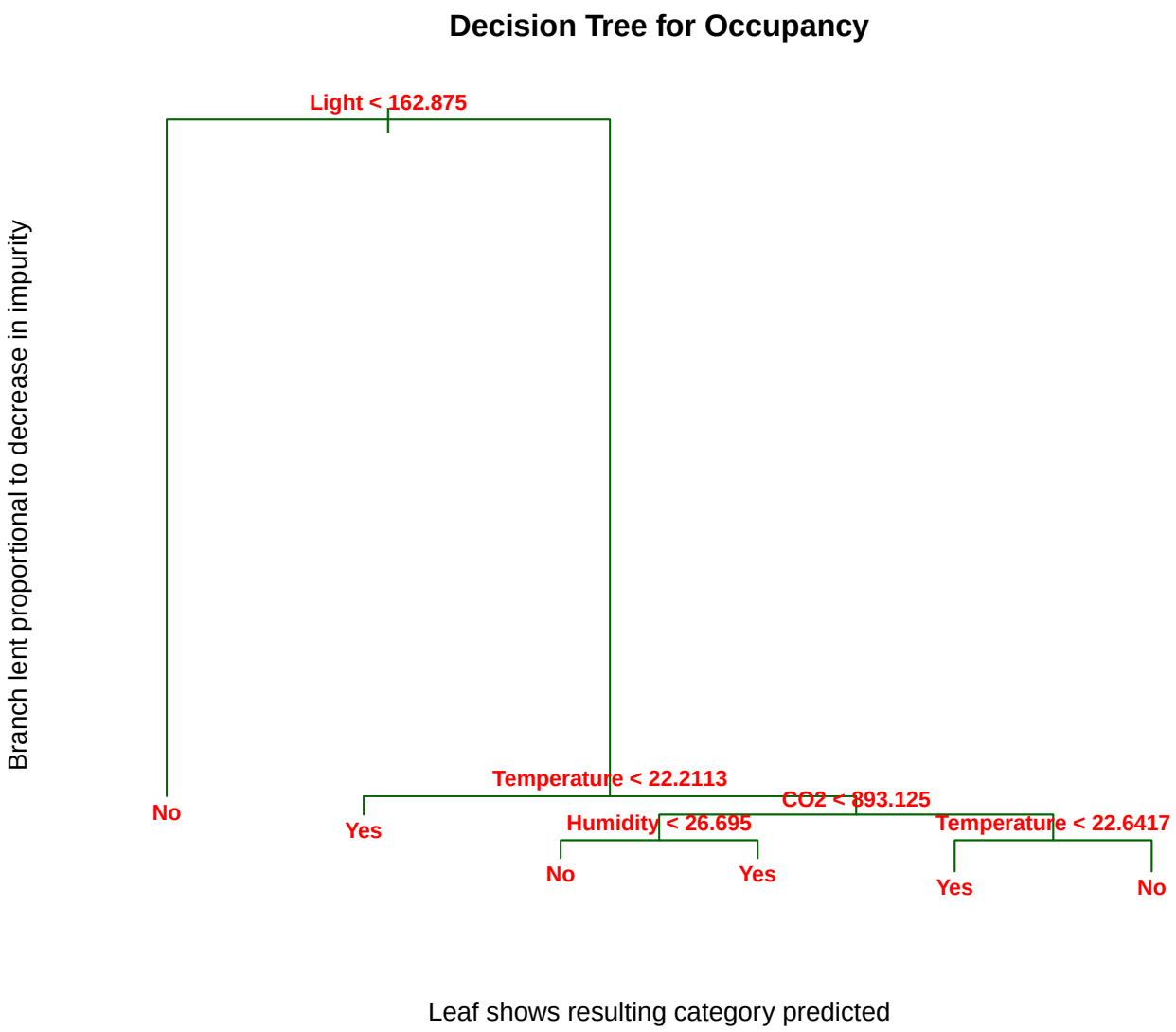
```
tree_predictions = predict(tree_model,newdata=test,type='class')
mean(tree_predictions != test$Occupancy)
```

```
## [1] 0.2033333
```

The error rate obtained on the testing data is 20.3%

2.b) Output and Analyse the tree.

```
plot(tree_model, col='dark green')
text(tree_model, pretty=0, cex=0.8, font=2,col='red')
title(main='Decision Tree for Occupancy',
      ylab='Branch lent proportional to decrease in impurity',
      xlab='Leaf shows resulting category predicted')
```



2.c) Random Forest classifier

Choosing `mtry` to be $2 \approx \sqrt{5}$

```
library(randomForest)
```

```
set.seed(550)
#Build randomForest model, selecting 2 predictors for each tree
forest = randomForest(y=train$Occupancy, x=train[,-6],ytest=test$Occupancy, xtest=test[,-6],
                      mtry=2, ntree=200, importance=TRUE)

#Test Error Rate
forest$test$err.rate[200,1][[1]]
```

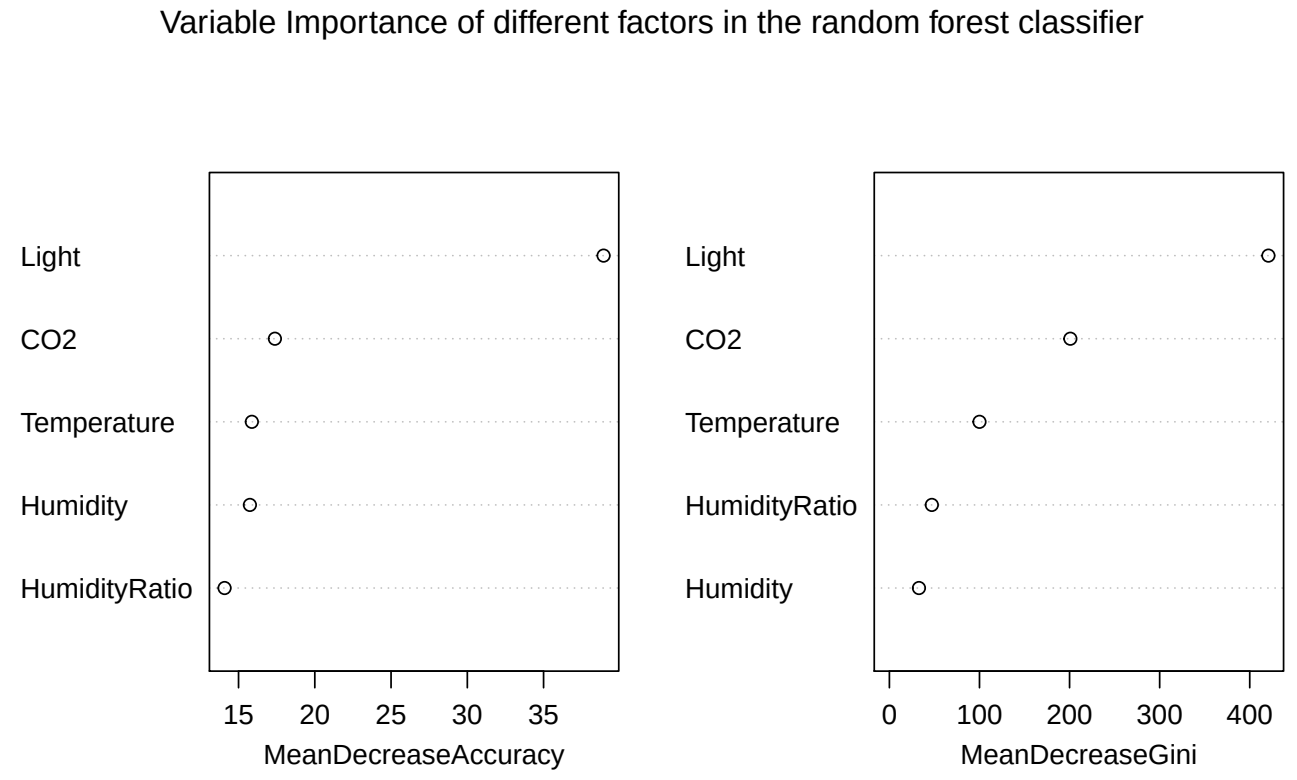
```
## [1] 0.2033333
```

2.d) Output and analyse the feature importance

```
kable(importance(forest))
```

	No	Yes	MeanDecreaseAccuracy	MeanDecreaseGini
Temperature	9.720758	9.725051	15.87244	100.00088
Humidity	9.503671	12.598939	15.74416	32.82776
Light	26.905715	36.988319	38.93518	420.72564
CO2	9.045737	16.113845	17.38359	200.85418
HumidityRatio	8.703706	10.428048	14.08431	47.10201

```
varImpPlot(forest,main='Variable Importance of different factors in the random forest classifier')
```



3. Support Vector Machines

3.a) Load wine data, find optimal parameters for and train a linear-kernel SVM

Load and prepare data

```
train = read.csv('WineQuality_training.txt')
test = read.csv('WineQuality_testing.txt')
train$quality = as.factor(train$quality)
test$quality = as.factor(test$quality)
```

Find Optimal cost for linear kernel

```
library(e1071)
set.seed(100)
tuner_linear = tune(svm,quality~.,data=train,kernel='linear',ranges=list(cost=c(0.01, 0.1, 1, 5, 10)))
summary(tuner_linear)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     1
##
## - best performance: 0.2393333
##
## - Detailed performance results:
##    cost     error dispersion
## 1  0.01 0.2473333 0.02243234
## 2  0.10 0.2423333 0.02061104
## 3  1.00 0.2393333 0.02130322
## 4  5.00 0.2423333 0.02043055
## 5 10.00 0.2416667 0.01995365
```

best cost value is 1

3.b) Train svm classifier using linear kernel and report predictive performance

Train a linear-kernel SVM classifier

```
linear_svm_model = svm(quality~., data=train, kernel='linear', cost=1, probability=TRUE)
summary(linear_svm_model)
```

```
##
## Call:
## svm(formula = quality ~ ., data = train, kernel = "linear", cost = 1,
##     probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##
## Number of Support Vectors:  1710
##
##  ( 855 855 )
##
##
## Number of Classes:  2
##
## Levels:
##  Bad Good
```

Make predictions on the testing data set and report performance

```
linear_predictions = predict(linear_svm_model, newdata=test)

kable(table(linear_predictions, test$quality))
```

	Bad	Good
Bad	104	89
Good	38	169

```
mean(linear_predictions != test$quality)
```

```
## [1] 0.3175
```

The error rate on the testing data is 31.75%

3.c) Find optimal hyper-parameters for RBF kernel

```
set.seed(100)
tuner_radial = tune(svm,quality~.,data=train,kernel='radial',
                    ranges=list(cost=c(0.01, 0.1, 1, 5, 10),
                                gamma = c(0.01, 0.03, 0.1, 0.5, 1)))
summary(tuner_radial)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     5   0.5
##
## - best performance: 0.1586667
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1   0.01  0.01 0.2826667 0.02412928
## 2   0.10  0.01 0.2516667 0.01715938
## 3   1.00  0.01 0.2336667 0.01745895
## 4   5.00  0.01 0.2086667 0.02838014
## 5  10.00  0.01 0.2046667 0.02863995
## 6   0.01  0.03 0.2643333 0.01414650
## 7   0.10  0.03 0.2400000 0.01968894
## 8   1.00  0.03 0.2056667 0.02629604
## 9   5.00  0.03 0.1950000 0.02178741
## 10 10.00  0.03 0.1930000 0.02279539
## 11  0.01  0.10 0.2660000 0.01546002
## 12  0.10  0.10 0.2090000 0.02171931
## 13  1.00  0.10 0.1893333 0.01623896
## 14  5.00  0.10 0.1806667 0.01639030
## 15 10.00  0.10 0.1740000 0.01916787
## 16  0.01  0.50 0.5166667 0.01798490
## 17  0.10  0.50 0.2300000 0.01632993
## 18  1.00  0.50 0.1653333 0.01596292
## 19  5.00  0.50 0.1586667 0.02167094
## 20 10.00  0.50 0.1650000 0.02121320
## 21  0.01  1.00 0.5180000 0.01642040
## 22  0.10  1.00 0.2196667 0.03802046
## 23  1.00  1.00 0.1653333 0.02315807
## 24  5.00  1.00 0.1603333 0.02622081
## 25 10.00  1.00 0.1590000 0.02615481
```

3.d) Train svm classifier using RBF kernel and report predictive performance

Train a radial-kernel SVM classifier

```
radial_svm_model = svm(quality~., data=train, kernel='radial',
                       cost=5, gamma=0.5, probability=TRUE)
summary(radial_svm_model)
```

```
##
## Call:
## svm(formula = quality ~ ., data = train, kernel = "radial", cost = 5,
##     gamma = 0.5, probability = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  5
##
## Number of Support Vectors:  1870
##
##  ( 867 1003 )
##
##
## Number of Classes:  2
##
## Levels:
##  Bad Good
```

Make predictions on the testing data set and report performance

```
radial_predictions = predict(radial_svm_model, newdata=test)

kable(table(radial_predictions, test$quality))
```

	Bad	Good
Bad	113	108
Good	29	150

```
mean(radial_predictions != test$quality)
```

```
## [1] 0.3425
```

3.e) ROC Curve analysis

```
library(ROCR)
```

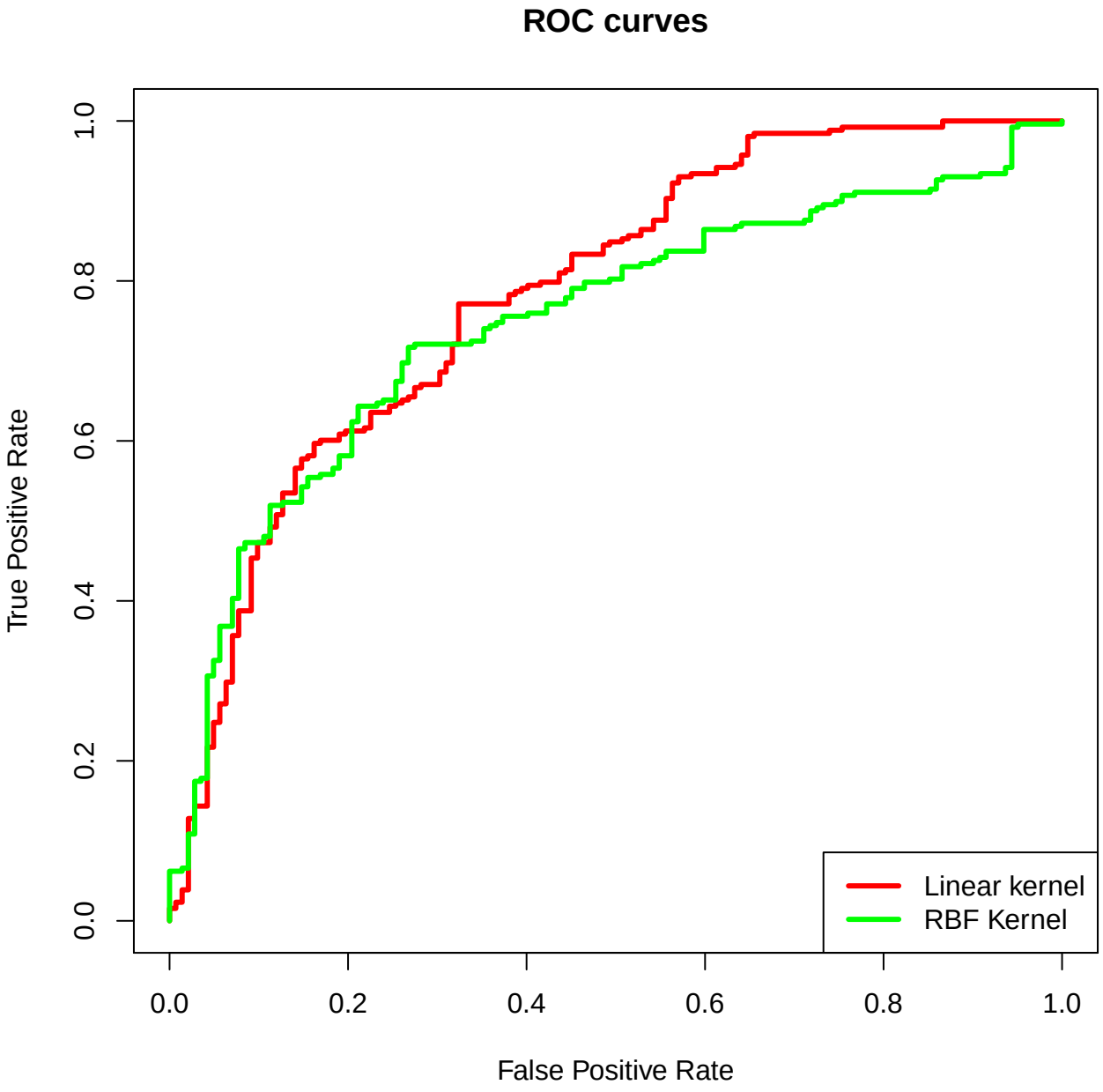
```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
# set up some helper functions to reduce code duplication
get_pred <- function(model){
  probs = predict(model, test, probability = TRUE, type='response')
  prob_Good = attributes(probs)[['probabilities']][,'Good']
  pred = prediction(prob_Good, test$quality)
  return(pred)
}

get_ROC_Curve <- function(model){
  perf = performance(get_pred(model), measure="tpr", x.measure="fpr")
  return(perf)
}

get_AUROC_Value <- function(model){
  perf = performance(get_pred(model), measure="auc")@y.values[[1]]
  return(perf)
}
```

```
#Plot ROC Curves
plot(0,xlab = "False Positive Rate", ylab = "True Positive Rate",
     main = "ROC curves",  ylim = 0:1, xlim = 0:1, type = "l")

plot(get_ROC_Curve(linear_svm_model), lwd=3, col="red", add=TRUE)
plot(get_ROC_Curve(radial_svm_model), lwd=3, col="green", add=TRUE)

legend("bottomright", c('Linear kernel','RBF Kernel'), col=c('red','green'), lty=c(1,1), lwd=c(3,3))
```




```
#Area under the linear svm model
get_AUROC_Value(linear_svm_model)
```

```
## [1] 0.7837646
```

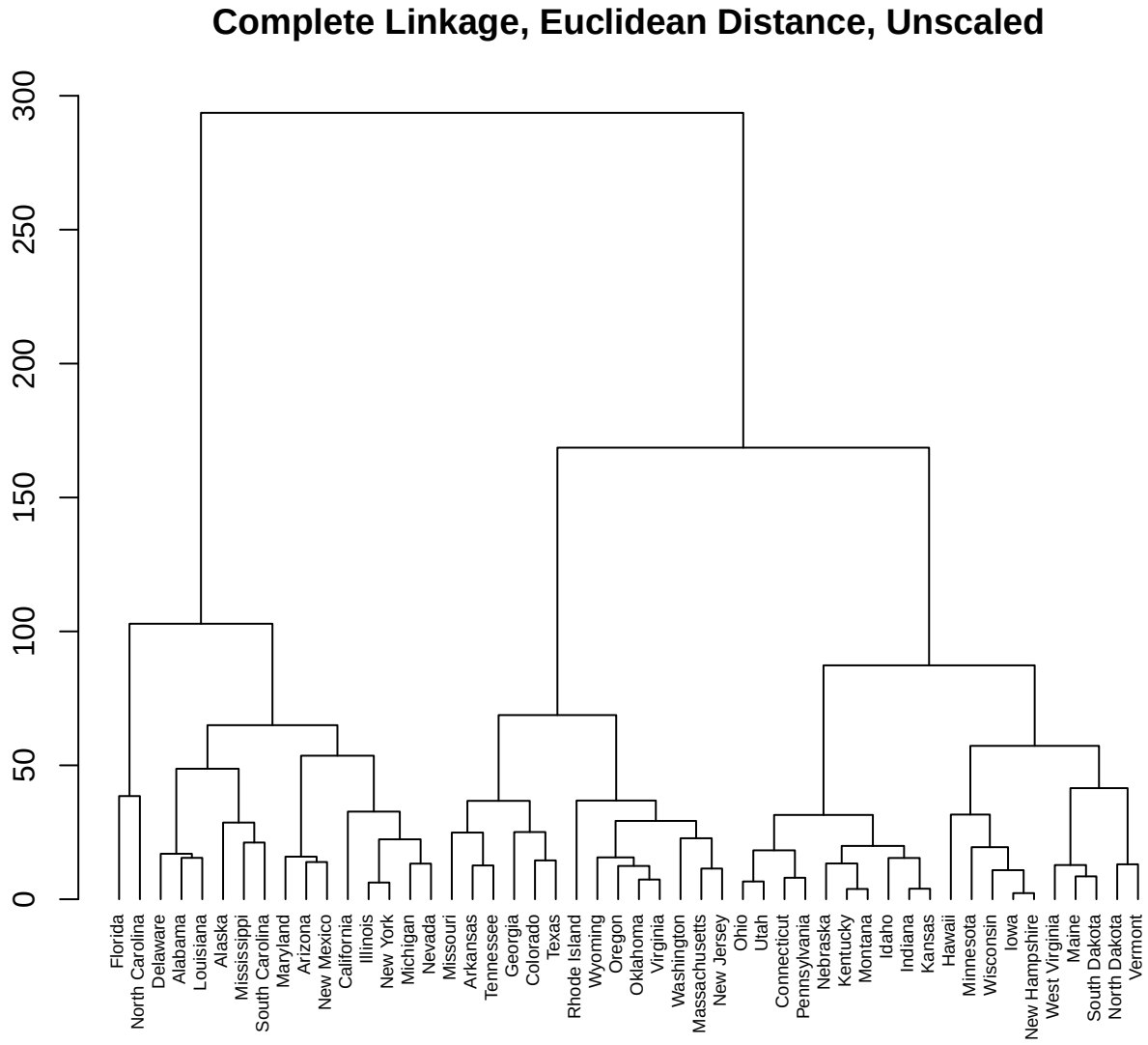
```
#Area under the ROC curve for the radial svm model
get_AUROC_Value(radial_svm_model)
```

```
## [1] 0.7483896
```

4. Hierarchical clustering

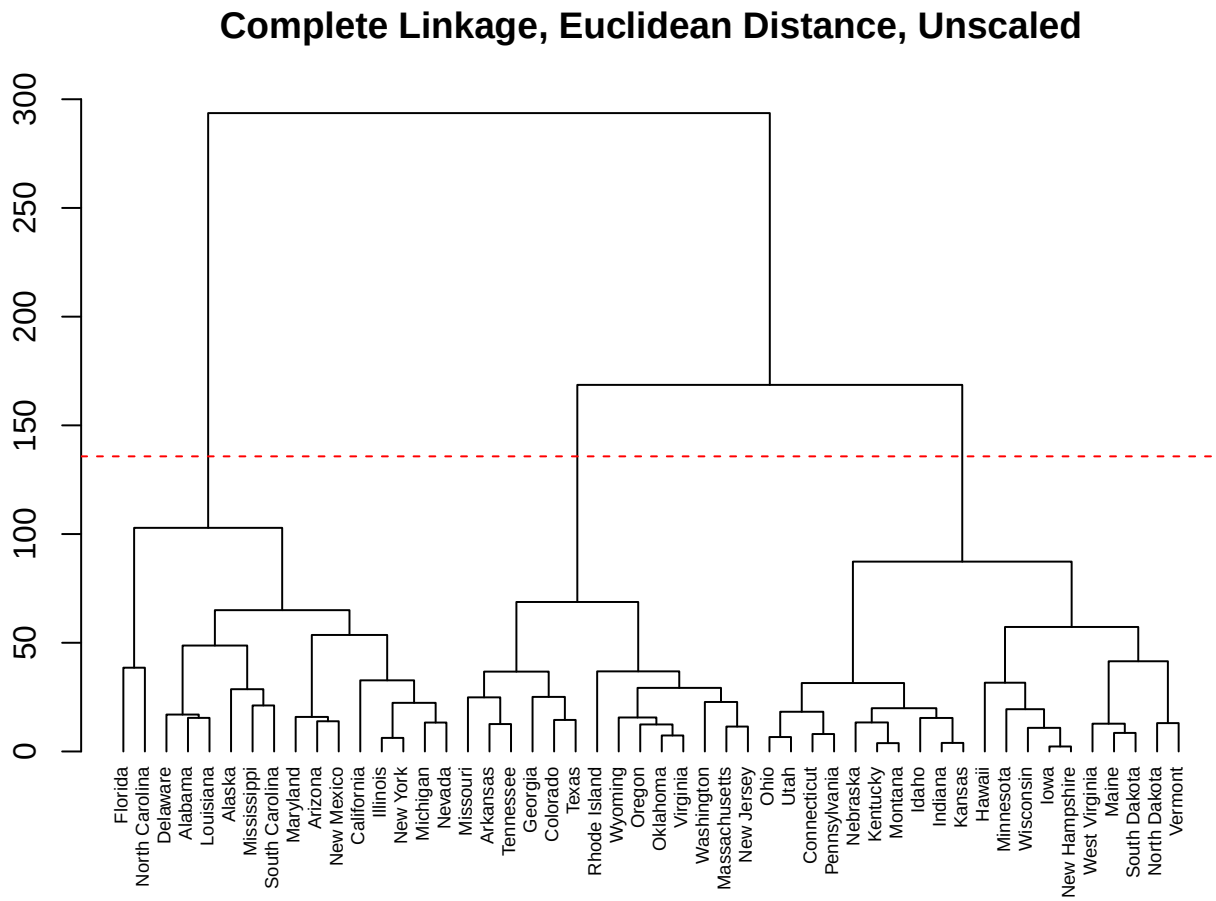
4.a) Hierarchical clustering complete linkage and Euclidean distance

```
hc = hclust(dist(USArrests),method='complete')
par(mar=c(0,3,3,2))
plot(hc, main='Complete Linkage, Euclidean Distance, Unscaled',
     xlab='',sub='',ylab="", hang=-1, cex=0.6)
```



4.b) Cut dendrogram into three clusters. Which states in each cluster?

```
par(mar=c(0,3,3,2))
plot(hc, main='Complete Linkage, Euclidean Distance, Unscaled',
     xlab='',sub='',ylab="", hang=-1, cex=0.6)
abline(h=mean(rev(hc$height)[2:3]),lty=2,col=2)
```



States that belong to each cluster

```
ct = cutree(hc,3)
# First Cluster
rownames(USArrests)[ct==1]
```

```
##  [1] "Alabama"        "Alaska"         "Arizona"        "California"
##  [5] "Delaware"       "Florida"        "Illinois"       "Louisiana"
##  [9] "Maryland"       "Michigan"       "Mississippi"    "Nevada"
## [13] "New Mexico"     "New York"       "North Carolina" "South Carolina"
```

```
# Second Cluster
rownames(USArrests)[ct==2]
```

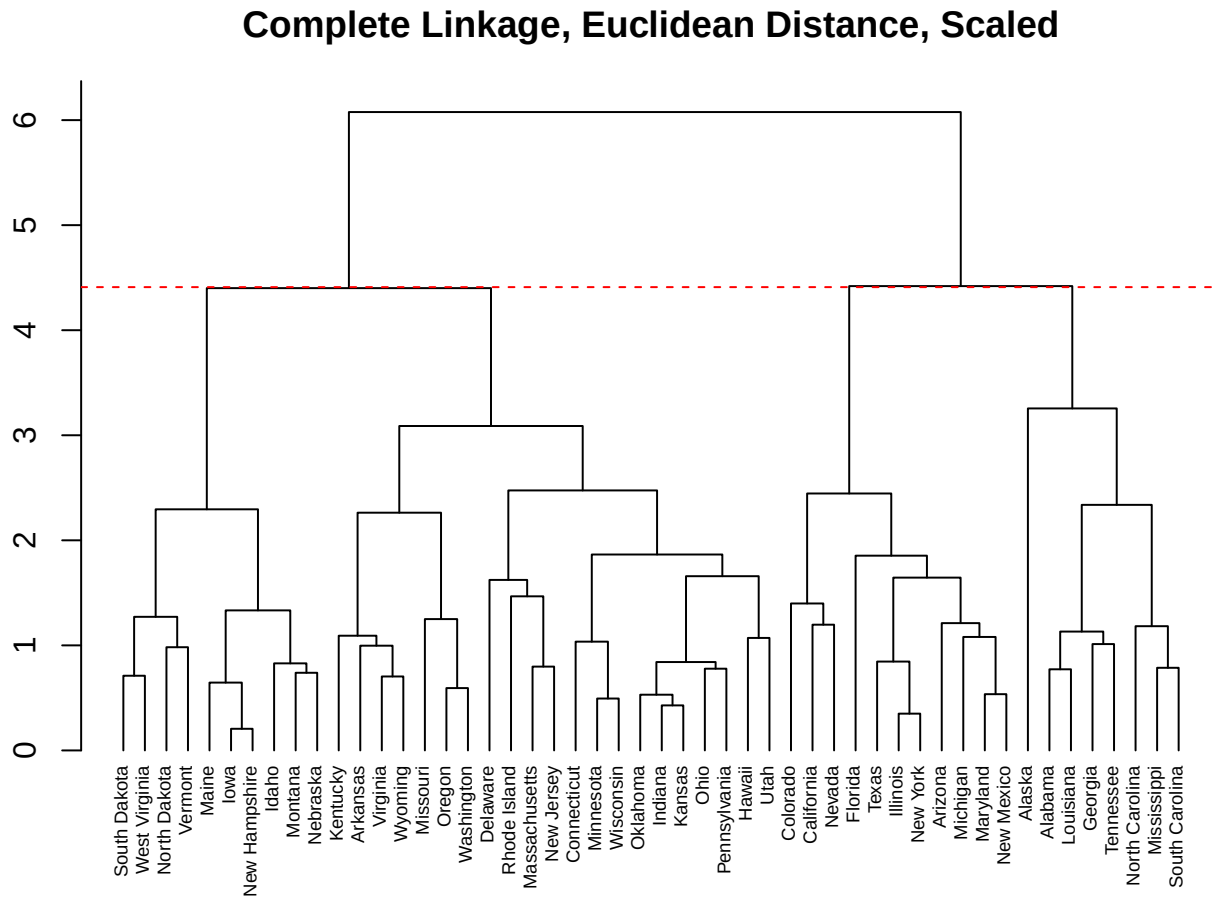
```
##  [1] "Arkansas"      "Colorado"      "Georgia"       "Massachusetts"
##  [5] "Missouri"      "New Jersey"    "Oklahoma"      "Oregon"
##  [9] "Rhode Island"  "Tennessee"     "Texas"         "Virginia"
## [13] "Washington"    "Wyoming"
```

```
# Third Cluster
rownames(USArrests)[ct==3]
```

```
##  [1] "Connecticut"   "Hawaii"        "Idaho"         "Indiana"
##  [5] "Iowa"          "Kansas"        "Kentucky"      "Maine"
##  [9] "Minnesota"     "Montana"       "Nebraska"      "New Hampshire"
## [13] "North Dakota"  "Ohio"          "Pennsylvania"  "South Dakota"
## [17] "Utah"          "Vermont"       "West Virginia" "Wisconsin"
```

4.c) Hierarchical clustering with complete linkage and Euclidean distance after scaling

```
hc.scaled = hclust(dist(scale(USArrests)),method='complete')

par(mar=c(0,3,3,2))
plot(hc.scaled, main='Complete Linkage, Euclidean Distance, Scaled',
     xlab='',sub='',ylab="", hang=-1, cex=0.6)
abline(h=mean(rev(hc.scaled$height)[2:3]),lty=2,col=2)
```




```
ct.scaled = cutree(hc.scaled,3)
#First Cluster
rownames(USArrests)[ct.scaled==1]
```

```
## [1] "Alabama"        "Alaska"         "Georgia"        "Louisiana"
## [5] "Mississippi"    "North Carolina" "South Carolina" "Tennessee"
```

```
#Second Cluster
rownames(USArrests)[ct.scaled==2]
```

```
##  [1] "Arizona"    "California" "Colorado"   "Florida"    "Illinois"
##  [6] "Maryland"   "Michigan"   "Nevada"     "New Mexico" "New York"
## [11] "Texas"
```

```
#Third Cluster
rownames(USArrests)[ct.scaled==3]
```

```
##  [1] "Arkansas"      "Connecticut"   "Delaware"      "Hawaii"
##  [5] "Idaho"         "Indiana"       "Iowa"          "Kansas"
```

```
##  [9] "Kentucky"      "Maine"         "Massachusetts" "Minnesota"
## [13] "Missouri"      "Montana"       "Nebraska"      "New Hampshire"
## [17] "New Jersey"    "North Dakota"  "Ohio"          "Oklahoma"
## [21] "Oregon"        "Pennsylvania"  "Rhode Island"  "South Dakota"
## [25] "Utah"          "Vermont"       "Virginia"      "Washington"
## [29] "West Virginia" "Wisconsin"     "Wyoming"
```

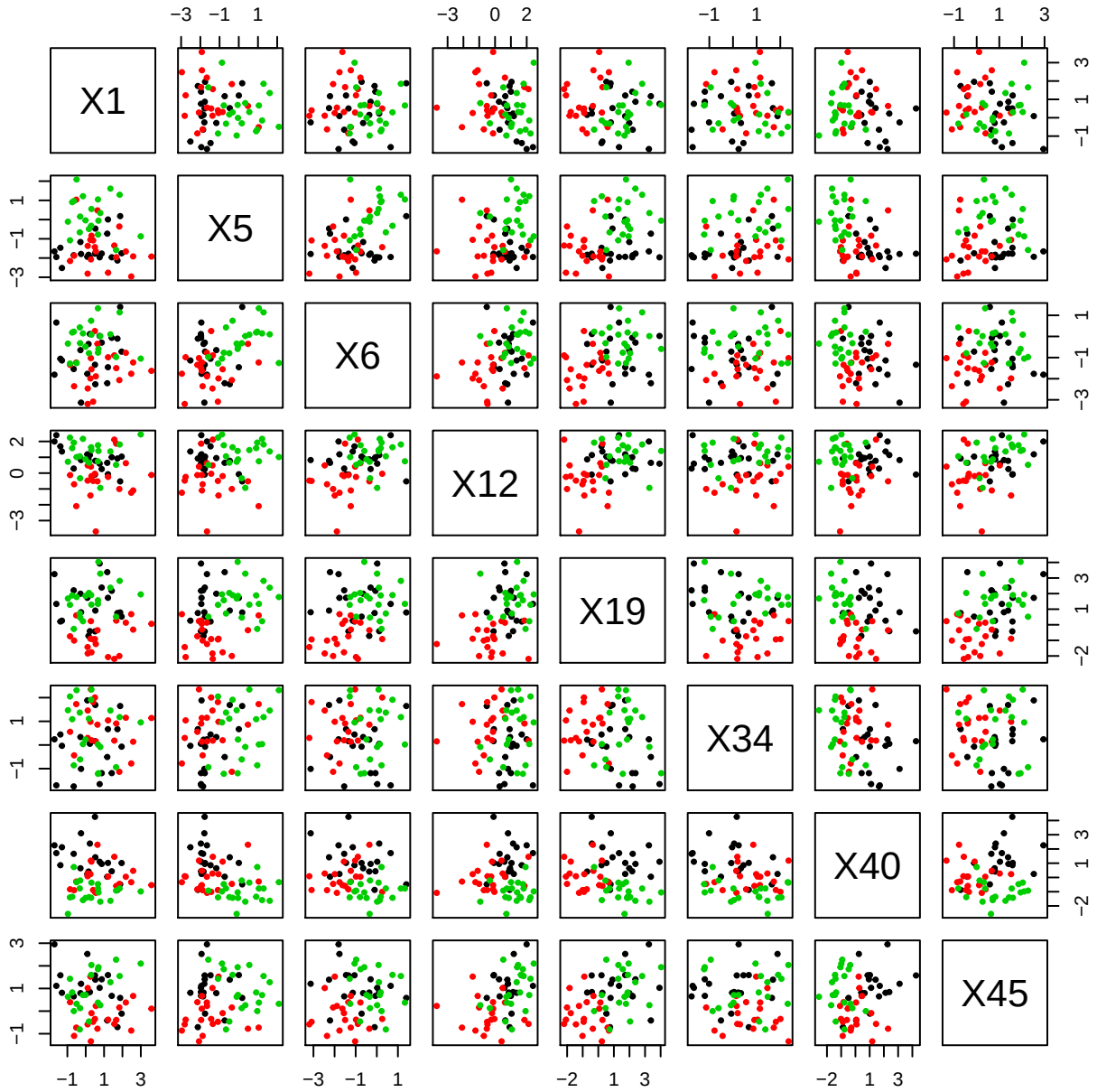
4.d) Effect of scaling

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5. PCA and K-Means Clustering

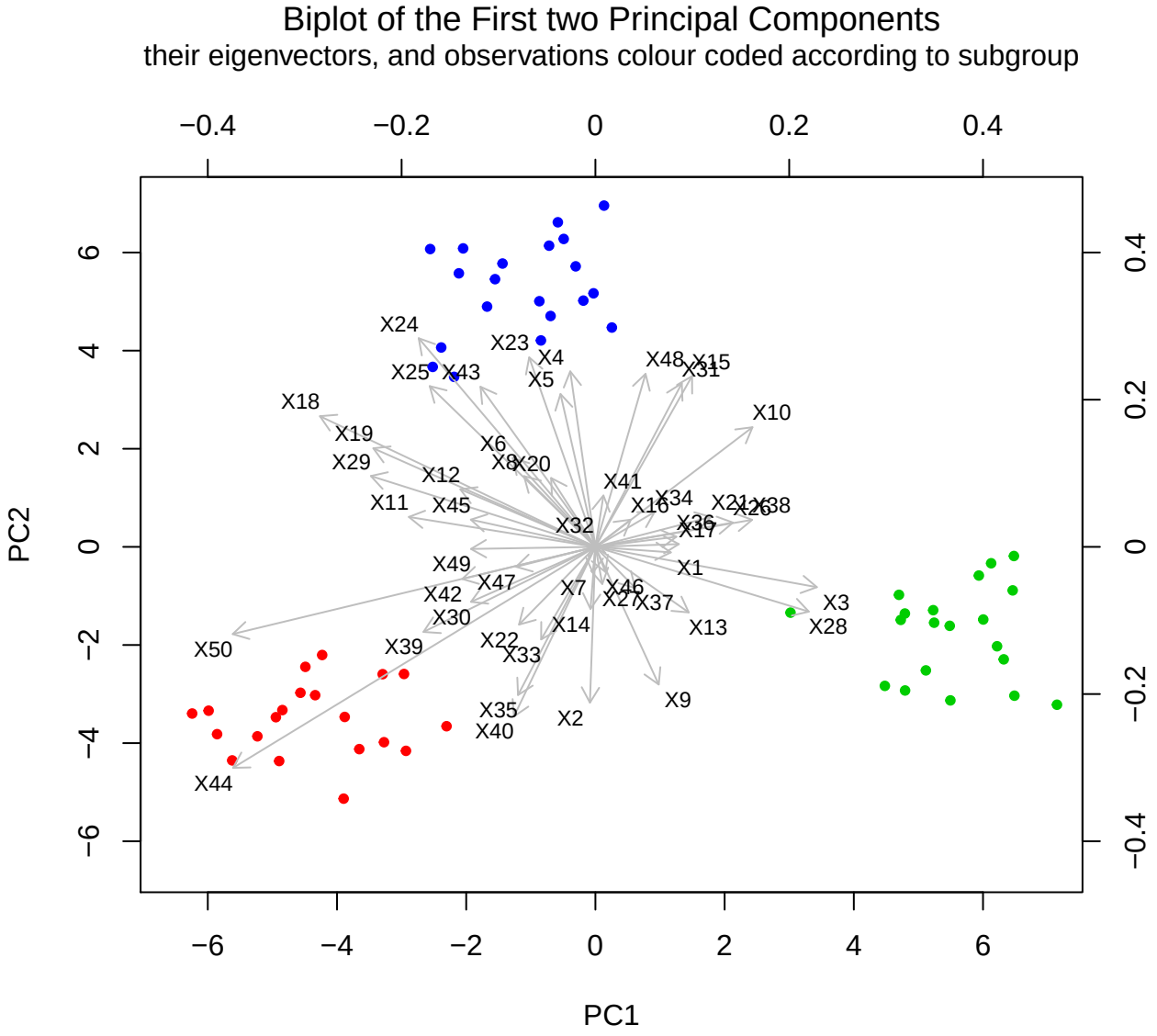
5.a Generate a simulated data set

```
set.seed(102)
randoms = matrix(rnorm(60*50),nrow=60,ncol=50)
categories = rep(1:3,each=20)
offsets = matrix(rnorm(3*50),3,50)[categories,]
dat = data.frame(randoms+offsets)

#pairs plot of some example dimensions to show shifted means
pairs(dat[c(1,5,6,12,19,34,40,45)],col=categories,cex=0.8,pch=20)
```



5.b Perform PCA analysis, check classes appear separated

```
pr <- prcomp(dat, scale=FALSE)

par(mar=c(4,4,5.5,4))
plot(pr$x[,1], pr$x[,2], col=categories+1, pch=20,
     xlab = "PC1", ylab = "PC2", xlim=c(-6.5,7), ylim=c(-6.5,7))

arrows(rep(0,50), rep(0,50), pr$rotation[,1]*15, pr$rotation[,2]*15, col='grey',length = 0.1)

text((pr$rotation[,1]+0.02*sign(pr$rotation[,1]))*15, (pr$rotation[,2]+0.02*sign(pr$rotation[,2]))*15,
      labels=rownames(pr$rotation), cex=0.75)

for (s in 3:4){axis(side=s,at=seq(-6,6,by=3),labels=seq(-0.4,0.4,by=0.2))}

mtext("Biplot of the First two Principal Components", side = 3, line = 4, cex = 1.2)
mtext("their eigenvectors, and observations colour coded according to subgroup",
      side = 3, line = 3, cex = 1.0)
```



5.c K-means clustering on raw data with 3 clusters

```
set.seed(102)
km = kmeans(dat,centers=3,nstart=100)
table(actual=categories,predicted=km$cluster)
```

```
##       predicted
## actual  1  2  3
##      1 20  0  0
##      2  0  0 20
##      3  0 20  0
```

K-means clustering successfully separates the clusters into three distinct classes.

5.d K-means clustering on raw data with 2 clusters

```
set.seed(102)
km = kmeans(dat,centers=2,nstart=100)
table(actual=categories,predicted=km$cluster)
```

```
##       predicted
## actual  1  2
##      1  0 20
##      2 20  0
##      3  0 20
```

Two of the actual clusters have been grouped together into one predicted cluster.

5.e K-means clustering on raw data with 4 clusters

```
set.seed(102)
km = kmeans(dat,centers=4,nstart=100)
table(actual=categories,predicted=km$cluster)
```

```
##       predicted
## actual  1  2  3  4
##      1  0  0  0 20
##      2 11  0  9  0
##      3  0 20  0  0
```

Two clusters have retained their integrity, however, one cluster has been split into two.

5.f K-means clustering on principal components

```
set.seed(102)
km_pc=kmeans(pr$x[,1:2],3,100)
table(actual=categories,predicted=km_pc$cluster)
```

```
##       predicted
## actual  1  2  3
##      1  0  0 20
##      2 20  0  0
##      3  0 20  0
```

The clusters are clearly distinguishable in the biplots above, so it not surprising that kmeans easily distinguishes between them by looking only at two dimenstions.

5.g K-means clustering on scaled raw data

```
set.seed(102)
km_sc=kmeans(scale(dat),centers=3,nstart=100)
table(actual=categories,predicted=km_sc$cluster)
```

```
##       predicted
## actual  1  2  3
##      1 20  0  0
##      2  0  0 20
##      3  0 20  0
```

Results are similar to those in 5.c, different only by the arbitrary class label allocated to the predicted cluster.