Data Mining Practicals

Sunny Mishra | 2K17/CS/19

5/19/2020

Practical 1

Q1. Create a file "people.txt" with the following data:

Age	agegroup	height	status	yearsmarried
21	adult	6.0	single	-1
2	child	3	married	0
18	adult	5.7	married	20
221	elderly	5	widowed	2
34	child	-7	married	3

i) Read the data from the file "*people.txt*".
ii) Create a ruleset *E* that contain rules to check for the following conditions:

1. The age should be in the range 0-150.
2. The age should be greater than yearsmarried.
3. The status should be married or single or widowed.
4. If age is less than 18 the agegroup should be child, if age is between 18 and 65 the agegroup should be adult, if age is more than 65 the agegroup should be elderly.

iii) Check whether ruleset *E* is violated by the data in the file people.txt.
iv) Summarize the results obtained in part (iii)
v) Visualize the results obtained in part (iii)

```
library(editrules)
```

```
## Loading required package: igraph
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

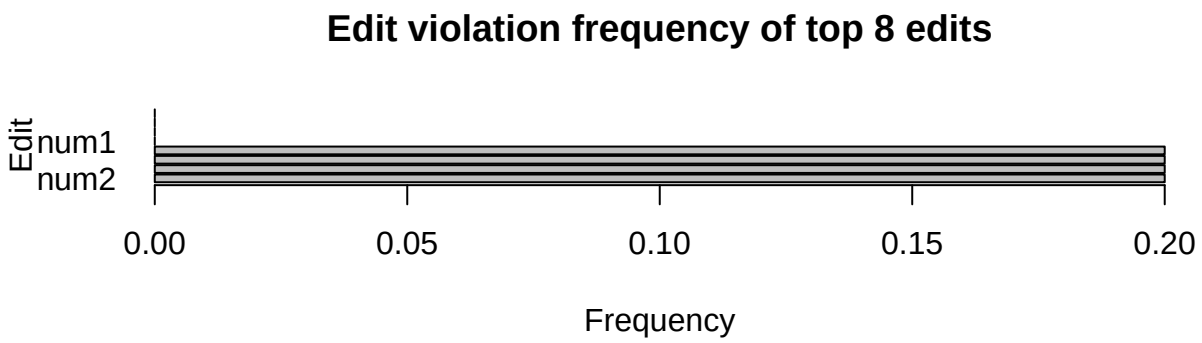
```
## The following object is masked from 'package:base':
##
```

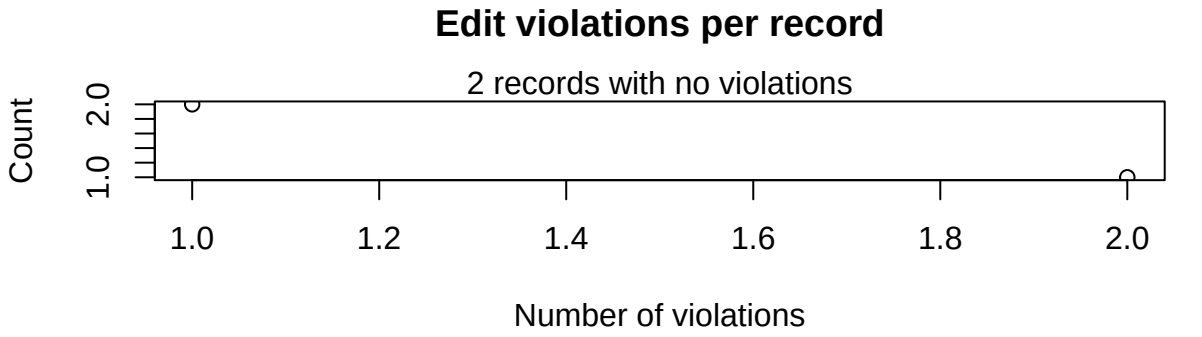
```
##     union
##
## Attaching package: 'editrules'
## The following objects are masked from 'package:igraph':
##
##     blocks, normalize
```

```
df <- read.table("practical1/people.txt", header=TRUE)
attach(df)
E <- editset(expression(
  age >= 0,
  age <= 150,
  age >= yearsmarried,
  status %in% c("married", "single", "widowed"),
  if (age <= 18) agegroup %in% c("child"),
  if (age >= 19 && age <= 64) agegroup %in% c("adult"),
  if (age >= 65) agegroup %in% c("elderly")
))

sm <- violatedEdits(E, df)
summary(sm)
```

```
## Edit violations, 5 observations, 0 completely missing (0%):
##
##  editname freq rel
##      num2    1 20%
##      num3    1 20%
##      mix4    1 20%
##      mix5    1 20%
##
## Edit violations per record:
##
##  errors freq rel
##       0    2 40%
##       1    2 40%
##       2    1 20%
```

```
plot(sm)
```


Edit violation frequency of top 8 edits
Edit
num1
num2
0.00
0.05
0.10
0.15
0.20
Frequency


Edit violations per record
2 records with no violations
Count
1.0
2.0
1.0
1.2
1.4
1.6
1.8
2.0
Number of violations

Practical 2

Q2. Perform the following preprocessing tasks on the dirty_iris dataset[ii].

i) Calculate the number and percentage of observations that are complete.

ii) Replace all the special values in data with NA.

iii) Define these rules in a separate text file and read them.

(Use editfile function in R (package editrules). Use similar function in Python).

Print the resulting constraint object.

- Species should be one of the following values: setosa, versicolor or virginica.
- All measured numerical properties of an iris should be positive.
- The petal length of an iris is at least 2 times its petal width.
- The sepal length of an iris cannot exceed 30 cm.
- The sepals of an iris are longer than its petals.

iv) Determine how often each rule is broken (violatedEdits). Also summarize and plot the result.

v) Find outliers in sepal length using boxplot and boxplot.stats

```
library(editrules)

df <- read.csv("./practical2/dirty_iris.csv")
df.complete <- df[complete.cases(df), ]
print(paste(
  "Complete cases are: ",
  nrow(df.complete),
  " and their percentage: ",
  nrow(df.complete) / nrow(df) * 100,
  "%",
  sep = ""
))
```

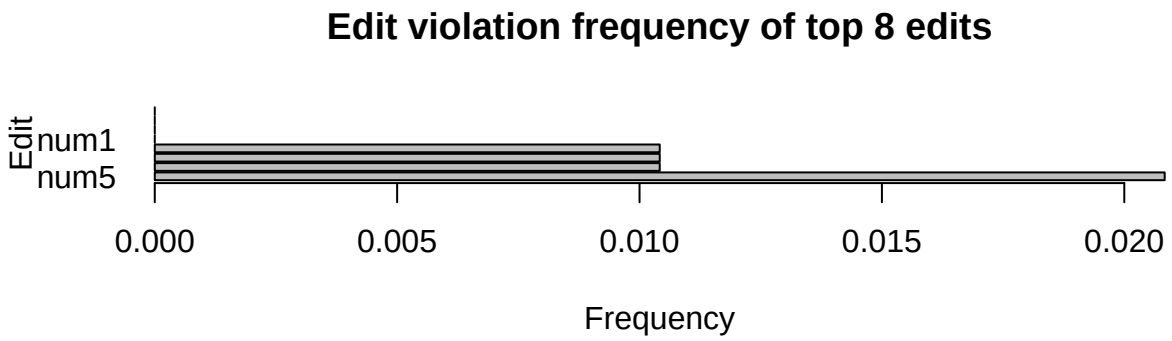
```
## [1] "Complete cases are: 96 and their percentage: 64%"
```

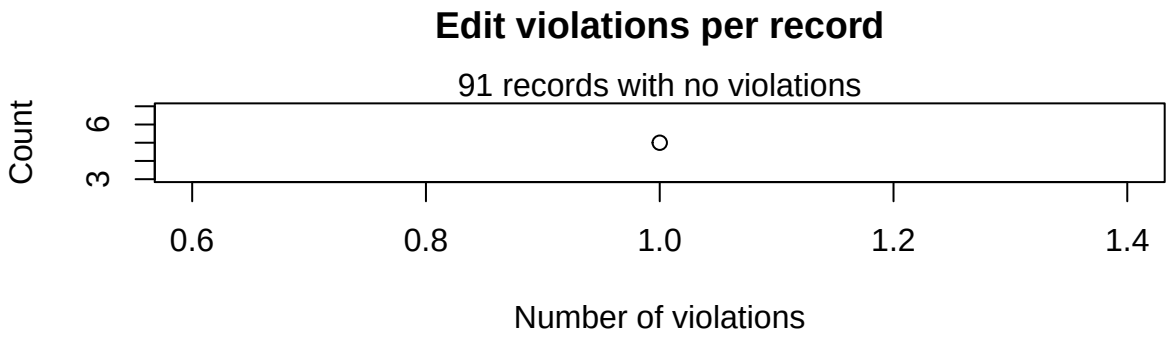
```
attach(df.complete)

E <- editfile("practical2/rules.txt")

violations <- violatedEdits(E, df.complete)
summary(violations)
```

```
## Edit violations, 96 observations, 0 completely missing (0%):
```

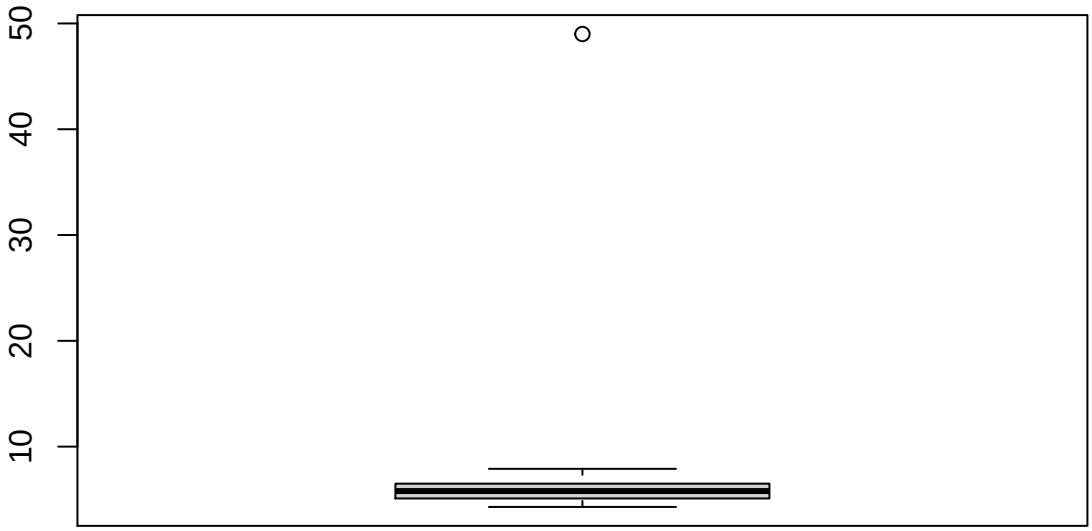
```
##
##  editname freq  rel
##      num5    2 2.1%
##      num2    1   1%
##      num6    1   1%
##      num7    1   1%
##
## Edit violations per record:
##
##  errors freq   rel
##       0   91 94.8%
##       1    5  5.2%
```

```
plot(violations)
```






```
boxplot(df.complete$Sepal.Length)
```



Practical 3

Q3. Load the data from wine dataset. Check whether all attributes are standardized or not (mean is 0 and standard deviation is 1). If not, standardize the attributes. Do the same with Iris dataset.

```
df <- read.csv(file = "practical3/wine.csv", sep = ";")

isNormalized <- function(df) {
  normalized <- TRUE
  for(i in 1:length(df)) {
    if (floor(mean(df[, i])) != 0 && sd(df[, i]) !=1) {
      normalized <- FALSE
    }
  }
  return(normalized)
}

normalizeTransformation <- function(x) {
  return ((x-mean(x))/sd(x))
}

if(isNormalized(df[, -12])) {
  cat("Dataset is normalized")
} else {
  cat("Dataset is not normalized")
  cat("\nNormalizing now")
  df.normalized <- data.frame(sapply(df[, -12], normalizeTransformation))
  df.normalized$quality <- df[, 12]
  if(isNormalized(df.normalized[, -12])) {
    cat("\nDataset is now normalized")
  } else {
    cat("\nNormalization failed")
  }
```

```
}
```

```
## Dataset is not normalized
## Normalizing now
## Dataset is now normalized
```

Practical 4

Q4. Run Apriori algorithm to find frequent itemsets and association rules

4.1 Use minimum support as 50% and minimum confidence as 75%

4.2 Use minimum support as 60% and minimum confidence as 60 %

```
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
data(Adult)

rules <- apriori(Adult, parameter = list(supp = 0.5, conf = .75, target = "rules"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.75    0.1    1 none FALSE            TRUE       5     0.5      1
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 24421
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[115 item(s), 48842 transaction(s)] done [0.05s].
## sorting and recoding items ... [9 item(s)] done [0.00s].
## creating transaction tree ... done [0.03s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [84 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rules)
```

```
## set of 84 rules
##
```

```
## rule length distribution (lhs + rhs):sizes
##  1  2  3  4
##  4 23 38 19
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.857   3.000   4.000
##
## summary of quality measures:
##     support         confidence        coverage           lift
##  Min.   :0.5084   Min.   :0.8504   Min.   :0.5406   Min.   :0.9789
##  1st Qu.:0.5415   1st Qu.:0.8888   1st Qu.:0.5931   1st Qu.:0.9943
##  Median :0.5897   Median :0.9132   Median :0.6640   Median :0.9988
##  Mean   :0.6433   Mean   :0.9110   Mean   :0.7070   Mean   :1.0034
##  3rd Qu.:0.7490   3rd Qu.:0.9422   3rd Qu.:0.8220   3rd Qu.:1.0077
##  Max.   :0.9533   Max.   :0.9583   Max.   :1.0000   Max.   :1.0586
##      count
##  Min.   :24832
##  1st Qu.:26447
##  Median :28803
##  Mean   :31422
##  3rd Qu.:36585
##  Max.   :46560
##
## mining info:
##   data ntransactions support confidence
##  Adult         48842     0.5       0.75
```

```
inspect(head(rules))
```

```
##     lhs                            rhs                                support confidence  coverage
## [1] {}                          => {race=White}                     0.8550428  0.8550428 1.0000000 1.000
## [2] {}                          => {native-country=United-States} 0.8974243  0.8974243 1.0000000 1.000
## [3] {}                          => {capital-gain=None}              0.9173867  0.9173867 1.0000000 1.000
## [4] {}                          => {capital-loss=None}              0.9532779  0.9532779 1.0000000 1.000
## [5] {hours-per-week=Full-time} => {native-country=United-States} 0.5179559  0.8852574 0.5850907 0.980
## [6] {hours-per-week=Full-time} => {capital-gain=None}              0.5435895  0.9290688 0.5850907 1.011
```

Practical 5

Q5. Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers. Divide the data set into training and test set. Compare the accuracy of the different classifiers under the following situations:

5.1 a) Training set = 75% Test set = 25% b) Training set = 66.6% (2/3rd of total), Test set = 33.3%

5.2 Training set is chosen by i) hold out method ii) Random subsampling iii) Cross-Validation. Compare the accuracy of the classifiers obtained.

5.3 Data is scaled to standard format.

```
library(rpart)
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
library(e1071)
library(class)
```

```
##
## Attaching package: 'class'
```

```
## The following object is masked from 'package:igraph':
##
##     knn
```

```
data(iris)

#Holdout method
smp_size <- floor(0.75 * nrow(iris))
train <- iris[1:smp_size, ]
test <- iris[-(1:smp_size), ]

model <- naiveBayes(Species ~ ., data = train)
prediction <- predict(model, test)
confusionMatrix(prediction, test[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa          0          0         0
##   versicolor      0          0         5
##   virginica       0          0        33
##
## Overall Statistics
##
##                Accuracy : 0.8684
##                  95% CI : (0.7191, 0.9559)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                     NA                NA           0.8684
## Specificity                      1            0.8684               NA
## Pos Pred Value                  NA                NA               NA
## Neg Pred Value                  NA                NA               NA
## Prevalence                       0            0.0000           1.0000
## Detection Rate                   0            0.0000           0.8684
## Detection Prevalence             0            0.1316           0.8684
## Balanced Accuracy               NA                NA               NA
```

```
model <- rpart(Species ~ ., data = train)
prediction <- predict(model, test, type = "class")
confusionMatrix(prediction, test[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa          0          0         0
##   versicolor      0          0         4
##   virginica       0          0        34
##
## Overall Statistics
##
##                Accuracy : 0.8947
##                  95% CI : (0.752, 0.9706)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                     NA                NA           0.8947
## Specificity                      1            0.8947               NA
## Pos Pred Value                  NA                NA               NA
## Neg Pred Value                  NA                NA               NA
## Prevalence                       0            0.0000           1.0000
## Detection Rate                   0            0.0000           0.8947
## Detection Prevalence             0            0.1053           0.8947
## Balanced Accuracy               NA                NA               NA
```

```
prediction = knn(train[,-5], test[,-5], factor(train[,5]), k = 10)
confusionMatrix(prediction, test[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa          0          0         0
##   versicolor      0          0        15
##   virginica       0          0        23
##
## Overall Statistics
##
##                Accuracy : 0.6053
##                  95% CI : (0.4339, 0.7596)
##     No Information Rate : 1
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0
```

```
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                     NA                NA           0.6053
## Specificity                      1            0.6053               NA
## Pos Pred Value                  NA                NA               NA
## Neg Pred Value                  NA                NA               NA
## Prevalence                       0            0.0000           1.0000
## Detection Rate                   0            0.0000           0.6053
## Detection Prevalence             0            0.3947           0.6053
## Balanced Accuracy               NA                NA               NA
```

```r
#Random Subsampling
smp_size <- floor(0.75 * nrow(iris))
set.seed(123)
train_ind <- sample(nrow(iris), size = smp_size)
train <- iris[train_ind, ]
test <- iris[-train_ind, ]

model <- naiveBayes(Species ~ ., data = train)
prediction <- predict(model, test)
confusionMatrix(prediction, test[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         17         0
##   virginica       0          0         9
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9075, 1)
##     No Information Rate : 0.4474
##     P-Value [Acc > NIR] : 5.312e-14
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           1.0000
## Specificity                 1.0000            1.0000           1.0000
## Pos Pred Value              1.0000            1.0000           1.0000
## Neg Pred Value              1.0000            1.0000           1.0000
## Prevalence                  0.3158            0.4474           0.2368
## Detection Rate              0.3158            0.4474           0.2368
## Detection Prevalence        0.3158            0.4474           0.2368
```

```
## Balanced Accuracy           1.0000            1.0000           1.0000
```

```
model <- rpart(Species ~ ., data = train)
prediction <- predict(model, test, type = "class")
confusionMatrix(prediction, test[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         17         1
##   virginica       0          0         8
##
## Overall Statistics
##
##                Accuracy : 0.9737
##                  95% CI : (0.8619, 0.9993)
##     No Information Rate : 0.4474
##     P-Value [Acc > NIR] : 2.547e-12
##
##                   Kappa : 0.9588
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.8889
## Specificity                 1.0000            0.9524           1.0000
## Pos Pred Value              1.0000            0.9444           1.0000
## Neg Pred Value              1.0000            1.0000           0.9667
## Prevalence                  0.3158            0.4474           0.2368
## Detection Rate              0.3158            0.4474           0.2105
## Detection Prevalence        0.3158            0.4737           0.2105
## Balanced Accuracy           1.0000            0.9762           0.9444
```

```
prediction = knn(train[,-5], test[,-5], factor(train[,5]), k = 10)
confusionMatrix(prediction, test[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         16         0
##   virginica       0          1         9
##
## Overall Statistics
##
##                Accuracy : 0.9737
##                  95% CI : (0.8619, 0.9993)
##     No Information Rate : 0.4474
##     P-Value [Acc > NIR] : 2.547e-12
##
```

```
##                  Kappa : 0.9595
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9412           1.0000
## Specificity                 1.0000            1.0000           0.9655
## Pos Pred Value              1.0000            1.0000           0.9000
## Neg Pred Value              1.0000            0.9545           1.0000
## Prevalence                  0.3158            0.4474           0.2368
## Detection Rate              0.3158            0.4211           0.2368
## Detection Prevalence        0.3158            0.4211           0.2632
## Balanced Accuracy           1.0000            0.9706           0.9828
```

```
train_control <- trainControl(method="cv", number=10)
model <- train(Species~., data=iris, trControl=train_control, method="nb")
prediction <- predict(model, test)
confusionMatrix(prediction, test[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         17         0
##   virginica       0          0         9
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9075, 1)
##     No Information Rate : 0.4474
##     P-Value [Acc > NIR] : 5.312e-14
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           1.0000
## Specificity                 1.0000            1.0000           1.0000
## Pos Pred Value              1.0000            1.0000           1.0000
## Neg Pred Value              1.0000            1.0000           1.0000
## Prevalence                  0.3158            0.4474           0.2368
## Detection Rate              0.3158            0.4474           0.2368
## Detection Prevalence        0.3158            0.4474           0.2368
## Balanced Accuracy           1.0000            1.0000           1.0000
```

```
train_control <- trainControl(method="cv", number=10)
model <- train(Species~., data=iris, trControl=train_control, method="rpart")
prediction <- predict(model, test)
```

```r
confusionMatrix(prediction, test[,5])$table
```

```
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         17         1
##   virginica       0          0         8
```

```r
train_control <- trainControl(method="cv", number=10)
model <- train(Species~., data=iris, trControl=train_control, method="knn")
prediction <- predict(model, test)
confusionMatrix(prediction, test[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         12          0         0
##   versicolor      0         16         0
##   virginica       0          1         9
##
## Overall Statistics
##
##                Accuracy : 0.9737
##                  95% CI : (0.8619, 0.9993)
##     No Information Rate : 0.4474
##     P-Value [Acc > NIR] : 2.547e-12
##
##                   Kappa : 0.9595
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9412           1.0000
## Specificity                 1.0000            1.0000           0.9655
## Pos Pred Value              1.0000            1.0000           0.9000
## Neg Pred Value              1.0000            0.9545           1.0000
## Prevalence                  0.3158            0.4474           0.2368
## Detection Rate              0.3158            0.4211           0.2368
## Detection Prevalence        0.3158            0.4211           0.2632
## Balanced Accuracy           1.0000            0.9706           0.9828
```

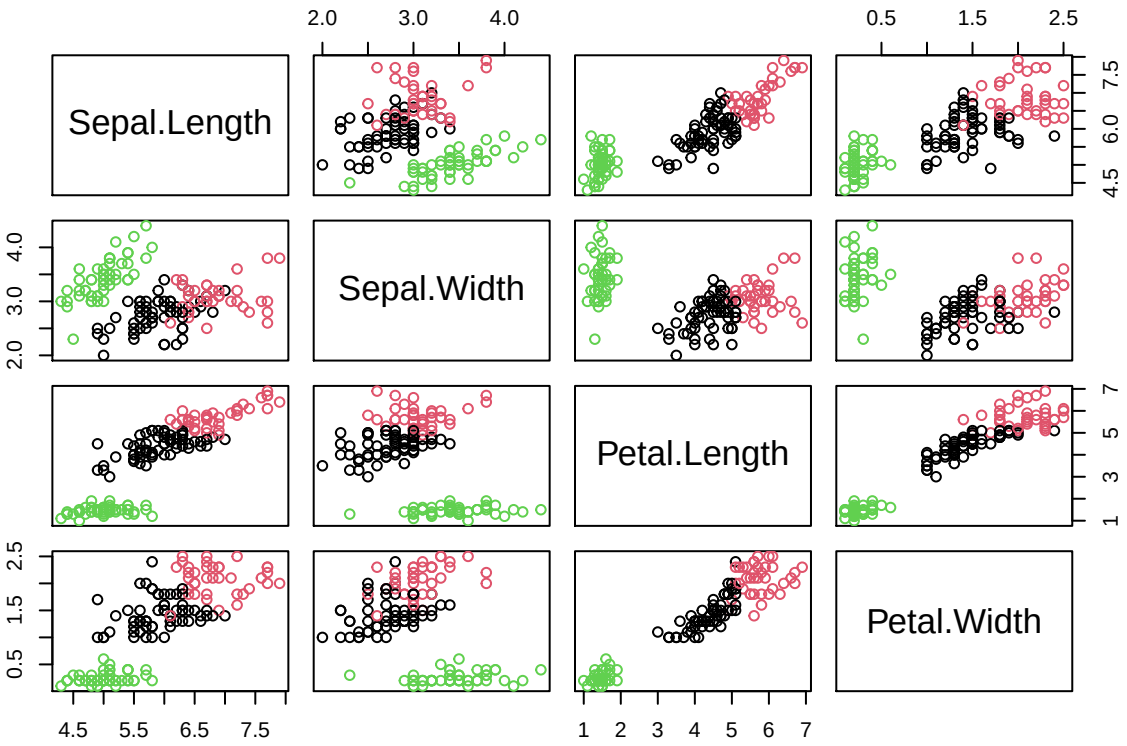
Practical 5

Q6. Use Simple Kmeans, DBScan, Hierachical clustering algorithms for clustering. Compare the performance of clusters by changing the parameters involved in the algorithms.

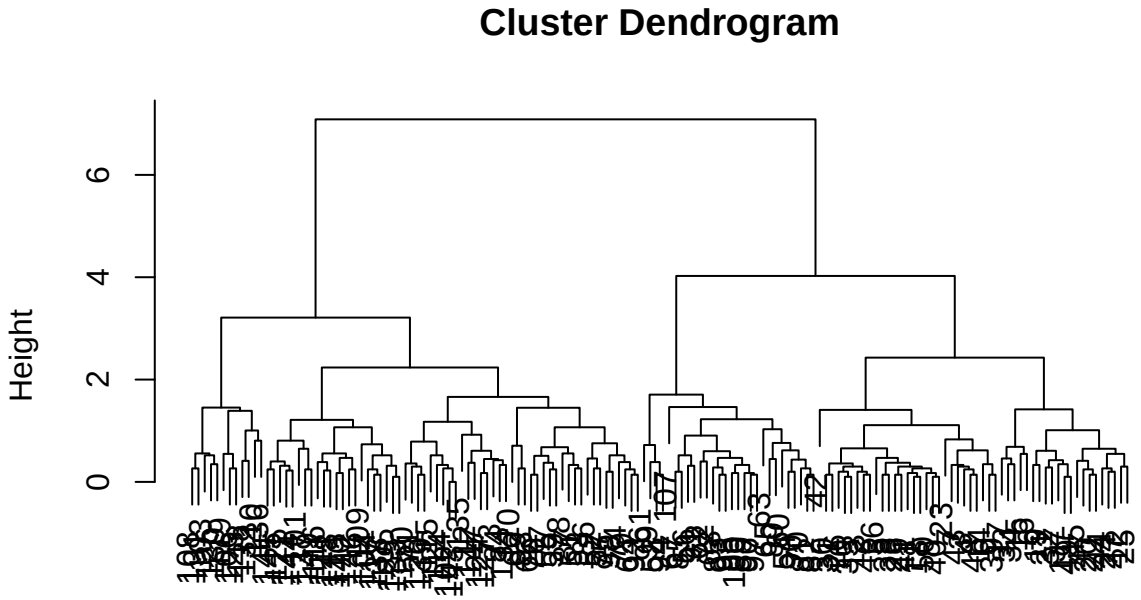
```r
library(dbscan)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```r
#kmeans
cl <- kmeans(iris[,-5], 3)
plot(iris[,-5], col = cl$cluster)
points(cl$centers, col = 1:3, pch = 8)
```




```r
#heirarchical
clusters <- hclust(dist(iris[, -5]))
plot(clusters)
```



dist(iris[, −5])
hclust (*, "complete")

```
#DBScan
cl <- dbscan(iris[,-5], eps = .5, minPts = 5)
plot(iris[,-5], col = cl$cluster)
```