DS8002 - Machine Learning Project 2 - Unsupervised and Supervised Learning (December 2016)

Najlis, Bernardo

December 2nd, 2016

This is the R code, illustrations and examples that go together with the report for DS8002 - Project 2.

0 - Data Preparation

Load required libraries, data sets, split train and test sets, label sets, etc.

```
library(ggplot2)
library(e1071)
library(caret)
```

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

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.3.2
```

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

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(RWeka)
library(NbClust)
```

```
## Warning: package 'NbClust' was built under R version 3.3.2
```

```
library(rpart)
library(partykit)
```

```
## Loading required package: grid
```

```
data(iris)  #load iris data
nrow(iris)
```

```
## [1] 150
```

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
iris.data <- iris[, 1:4]  # features
iris.class <- iris[, 5] # labels

ind.iris <- sample(2, nrow(iris), replace=TRUE, prob=c(0.67, 0.33)) # get random indices for training /
iris.training <- iris[ind.iris==1,1:4]                              # get training set
iris.trainingWithLabels <- iris[ind.iris==1,]                       # training set with labels
iris.test <- iris[ind.iris==2,]                                     # get test set
iris.trainLabels <- iris[ind.iris==1, 5]                            # get labels for training
iris.testLabels <- iris[ind.iris==2, 5]                             # get labels for set

lenses <- read.table("lenses.data", # name of file reading, this requires setting the working directory
                     header= FALSE, # header is not included in first line
                     col.names =    # to provide names for columns
                       c("id", "age", "spectacle_prescription", "astigmatic", "tear_production_rate", "
                     colClasses=    # data types for columns
                       c("NULL",    # as first column is specified as "NULL", read.table will skip this
                         rep("integer", 4), # all other attributes are integer
                          "factor"          # the last column is the class, typified as factor
                         ))
nrow(lenses)
```

```
## [1] 24
```

```
head(lenses)
```

```
##   age spectacle_prescription astigmatic tear_production_rate class
## 1   1                      1          1                    1     3
## 2   1                      1          1                    2     2
## 3   1                      1          2                    1     3
## 4   1                      1          2                    2     1
## 5   1                      2          1                    1     3
## 6   1                      2          1                    2     2
```

```
lenses.data <- lenses[, 1:4]
lenses.class <- lenses[, 5]

ind.lenses <- sample(2, nrow(lenses), replace=TRUE, prob=c(0.8, 0.2)) # get random indices for training
lenses.training <- lenses[ind.lenses==1,1:4]                          # get training set
lenses.trainingWithLabels <- lenses[ind.lenses==1,]                   # training set with labels
lenses.test <- lenses[ind.lenses==2,]                                 # get test set
lenses.trainLabels <- lenses[ind.lenses==1, 5]                        # get labels for training
lenses.testLabels <- lenses[ind.lenses==2, 5]                         # get labels for set
```

1 - SVM

Run SVM with different kernels and then compare.

Iris

```
# First use tune to select best model parameters

iris.svm.linear.tuned <- tune.svm(Species~.,                                # class and features
                                  data=iris.trainingWithLabels,             # data frame
                                  kernel="linear",                          # kernel
                                  cost=c(0.001, 0.01, 0.1, 1, 10, 100)    # parameter values to try mode
                                  )

iris.svm.linear.tuned <- tune.svm(Species~.,                                # class and features
                                  data=iris.trainingWithLabels,                      # data frame
                                  kernel="linear",                          # kernel
                                  cost=c(0.001, 0.01, 0.1, 1, 10, 100)    # parameter values to try mode
                                  )
summary(iris.svm.linear.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.04222222
##
## - Detailed performance results:
##    cost      error dispersion
## 1 1e-03 0.61555556 0.14686481
## 2 1e-02 0.28555556 0.10975093
## 3 1e-01 0.06444444 0.07640769
## 4 1e+00 0.04333333 0.07670334
## 5 1e+01 0.04222222 0.05463434
## 6 1e+02 0.04222222 0.05463434
```

```
iris.svm.polynomial.tuned <- tune.svm(Species~., data=iris.trainingWithLabels, kernel="polynomial",
                                      degree = c(3, 4, 5),              # degree of polynomial
                                      coef0=c(0.1, 0.5, 1, 2, 3, 4))    # kernel coefficient

summary(iris.svm.polynomial.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
```

```
##
## - best parameters:
##  degree coef0
##       3   0.5
##
## - best performance: 0.04333333
##
## - Detailed performance results:
##    degree coef0      error dispersion
## 1       3   0.1 0.07444444 0.08733990
## 2       4   0.1 0.09555556 0.09416444
## 3       5   0.1 0.12666667 0.08367420
## 4       3   0.5 0.04333333 0.07670334
## 5       4   0.5 0.04333333 0.07670334
## 6       5   0.5 0.05444444 0.07776896
## 7       3   1.0 0.04333333 0.07670334
## 8       4   1.0 0.05444444 0.07776896
## 9       5   1.0 0.05444444 0.07776896
## 10      3   2.0 0.05444444 0.07776896
## 11      4   2.0 0.05444444 0.07776896
## 12      5   2.0 0.05444444 0.07776896
## 13      3   3.0 0.05444444 0.07776896
## 14      4   3.0 0.05444444 0.07776896
## 15      5   3.0 0.05333333 0.07694439
## 16      3   4.0 0.05444444 0.07776896
## 17      4   4.0 0.06555556 0.07706018
## 18      5   4.0 0.07333333 0.08689113
```

```
iris.svm.radial.tuned <- tune.svm(Species~., data=iris.trainingWithLabels, kernel="radial",
                                  gamma = c(0.1, 0.5, 1, 2, 3, 4))          # gamma coefficient
summary(iris.svm.radial.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma
##      3
##
## - best performance: 0.04222222
##
## - Detailed performance results:
##   gamma      error dispersion
## 1   0.1 0.05333333 0.07694439
## 2   0.5 0.04333333 0.07670334
## 3   1.0 0.04333333 0.07670334
## 4   2.0 0.04333333 0.07670334
## 5   3.0 0.04222222 0.07568616
## 6   4.0 0.05333333 0.07694439
```

```r
iris.svm.sigmoid.tuned <- tune.svm(Species~., data=iris.trainingWithLabels, kernel="sigmoid",
                                   gamma = c(0.1, 0.5, 1, 2, 3, 4),          # gamma
                                   coef0=c(0.1, 0.5, 1, 2, 3, 4))            # kernel coefficient
summary(iris.svm.sigmoid.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma coef0
##    0.1   0.1
##
## - best performance: 0.05111111
##
## - Detailed performance results:
##    gamma coef0      error dispersion
## 1    0.1   0.1 0.05111111 0.08577893
## 2    0.5   0.1 0.29777778 0.16603171
## 3    1.0   0.1 0.40666667 0.23136738
## 4    2.0   0.1 0.40111111 0.12983581
## 5    3.0   0.1 0.38666667 0.13102948
## 6    4.0   0.1 0.38888889 0.15449373
## 7    0.1   0.5 0.05333333 0.05636448
## 8    0.5   0.5 0.30666667 0.15583749
## 9    1.0   0.5 0.41222222 0.18632611
## 10   2.0   0.5 0.35777778 0.13984118
## 11   3.0   0.5 0.36777778 0.16105363
## 12   4.0   0.5 0.37777778 0.14487116
## 13   0.1   1.0 0.11555556 0.16247401
## 14   0.5   1.0 0.31444444 0.17870514
## 15   1.0   1.0 0.33444444 0.16088319
## 16   2.0   1.0 0.29333333 0.20529346
## 17   3.0   1.0 0.32555556 0.19647441
## 18   4.0   1.0 0.32555556 0.13878268
## 19   0.1   2.0 0.29333333 0.17948257
## 20   0.5   2.0 0.32333333 0.16156386
## 21   1.0   2.0 0.36555556 0.14791185
## 22   2.0   2.0 0.36555556 0.20009943
## 23   3.0   2.0 0.36444444 0.18934585
## 24   4.0   2.0 0.36333333 0.20102959
## 25   0.1   3.0 0.61000000 0.15726137
## 26   0.5   3.0 0.38888889 0.22246900
## 27   1.0   3.0 0.45333333 0.20772463
## 28   2.0   3.0 0.40555556 0.16457536
## 29   3.0   3.0 0.37555556 0.21505733
## 30   4.0   3.0 0.40444444 0.18711586
## 31   0.1   4.0 0.61000000 0.15726137
## 32   0.5   4.0 0.61000000 0.15726137
## 33   1.0   4.0 0.44777778 0.20448000
## 34   2.0   4.0 0.48111111 0.20641628
## 35   3.0   4.0 0.38555556 0.15538350
```

```
## 36   4.0   4.0 0.41555556 0.22105868
```

```
# Now use the best model with the best cost as selected by tune()

iris.svm.linear.best <- iris.svm.linear.tuned$best.model
iris.svm.linear.best.pred <- predict(iris.svm.linear.best, iris.test)
confusionMatrix(iris.svm.linear.best.pred, iris.testLabels)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         12         0
##   virginica       0          1        22
##
## Overall Statistics
##
##                Accuracy : 0.9818
##                  95% CI : (0.9028, 0.9995)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.972
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9231           1.0000
## Specificity                 1.0000            1.0000           0.9697
## Pos Pred Value              1.0000            1.0000           0.9565
## Neg Pred Value              1.0000            0.9767           1.0000
## Prevalence                  0.3636            0.2364           0.4000
## Detection Rate              0.3636            0.2182           0.4000
## Detection Prevalence        0.3636            0.2182           0.4182
## Balanced Accuracy           1.0000            0.9615           0.9848
```

```
iris.svm.polynomial.best <- iris.svm.polynomial.tuned$best.model
iris.svm.polynomial.best.pred <- predict(iris.svm.polynomial.best, iris.test)
confusionMatrix(iris.svm.polynomial.best.pred, iris.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         13         2
##   virginica       0          0        20
##
## Overall Statistics
##
##                Accuracy : 0.9636
```

```
##                  95% CI : (0.8747, 0.9956)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9447
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9091
## Specificity                 1.0000            0.9524           1.0000
## Pos Pred Value              1.0000            0.8667           1.0000
## Neg Pred Value              1.0000            1.0000           0.9429
## Prevalence                  0.3636            0.2364           0.4000
## Detection Rate              0.3636            0.2364           0.3636
## Detection Prevalence        0.3636            0.2727           0.3636
## Balanced Accuracy           1.0000            0.9762           0.9545
```

```
iris.svm.radial.best <- iris.svm.radial.tuned$best.model
iris.svm.radial.best.pred <- predict(iris.svm.radial.best, iris.test)
confusionMatrix(iris.svm.radial.best.pred, iris.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         19          0         0
##   versicolor      0         12         2
##   virginica       1          1        20
##
## Overall Statistics
##
##                Accuracy : 0.9273
##                  95% CI : (0.8241, 0.9798)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : 2.361e-16
##
##                   Kappa : 0.8888
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 0.9500            0.9231           0.9091
## Specificity                 1.0000            0.9524           0.9394
## Pos Pred Value              1.0000            0.8571           0.9091
## Neg Pred Value              0.9722            0.9756           0.9394
## Prevalence                  0.3636            0.2364           0.4000
## Detection Rate              0.3455            0.2182           0.3636
## Detection Prevalence        0.3455            0.2545           0.4000
## Balanced Accuracy           0.9750            0.9377           0.9242
```

```r
iris.svm.sigmoid.best <- iris.svm.sigmoid.tuned$best.model
iris.svm.sigmoid.best.pred <- predict(iris.svm.sigmoid.best, iris.test)
confusionMatrix(iris.svm.sigmoid.best.pred, iris.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         13         2
##   virginica       0          0        20
##
## Overall Statistics
##
##                Accuracy : 0.9636
##                  95% CI : (0.8747, 0.9956)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9447
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9091
## Specificity                 1.0000            0.9524           1.0000
## Pos Pred Value              1.0000            0.8667           1.0000
## Neg Pred Value              1.0000            1.0000           0.9429
## Prevalence                  0.3636            0.2364           0.4000
## Detection Rate              0.3636            0.2364           0.3636
## Detection Prevalence        0.3636            0.2727           0.3636
## Balanced Accuracy           1.0000            0.9762           0.9545
```

Contact Lenses

```r
lenses.svm.linear.tuned <- tune.svm(class~., data=lenses.trainingWithLabels, kernel="linear", cost=c(0.
summary(lenses.svm.linear.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##    10
##
## - best performance: 0.2833333
##
## - Detailed performance results:
```

```
##    cost     error dispersion
## 1 1e-03 0.3666667  0.3122993
## 2 1e-02 0.3666667  0.3122993
## 3 1e-01 0.4166667  0.2859897
## 4 1e+00 0.3333333  0.4082483
## 5 1e+01 0.2833333  0.2490724
## 6 1e+02 0.2833333  0.2490724
```

```
lenses.svm.polynomial.tuned <- tune.svm(class~., data=lenses.trainingWithLabels, kernel="polynomial", d
summary(lenses.svm.polynomial.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  degree coef0
##       3   0.5
##
## - best performance: 0.1833333
##
## - Detailed performance results:
##    degree coef0     error dispersion
## 1       3   0.1 0.2833333  0.2490724
## 2       4   0.1 0.3166667  0.2283867
## 3       5   0.1 0.3333333  0.3333333
## 4       3   0.5 0.1833333  0.2415229
## 5       4   0.5 0.2000000  0.2581989
## 6       5   0.5 0.2000000  0.2581989
## 7       3   1.0 0.2333333  0.2509242
## 8       4   1.0 0.1833333  0.2415229
## 9       5   1.0 0.2000000  0.2581989
## 10      3   2.0 0.2333333  0.2509242
## 11      4   2.0 0.2333333  0.2509242
## 12      5   2.0 0.1833333  0.2415229
## 13      3   3.0 0.2333333  0.2509242
## 14      4   3.0 0.2333333  0.2509242
## 15      5   3.0 0.2333333  0.2509242
## 16      3   4.0 0.2333333  0.2509242
## 17      4   4.0 0.2333333  0.2509242
## 18      5   4.0 0.2333333  0.2509242
```

```
lenses.svm.radial.tuned <- tune.svm(class~., data=lenses.trainingWithLabels, kernel="radial", gamma = c
summary(lenses.svm.radial.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma
```

```
##    0.5
##
## - best performance: 0.3333333
##
## - Detailed performance results:
##   gamma     error dispersion
## 1   0.1 0.3833333  0.3147603
## 2   0.5 0.3333333  0.3333333
## 3   1.0 0.3333333  0.3333333
## 4   2.0 0.3333333  0.3333333
## 5   3.0 0.3333333  0.3333333
## 6   4.0 0.3333333  0.3333333
```

```
lenses.svm.sigmoid.tuned <- tune.svm(class~., data=lenses.trainingWithLabels, kernel="sigmoid", gamma =
summary(lenses.svm.sigmoid.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma coef0
##      3     3
##
## - best performance: 0.3333333
##
## - Detailed performance results:
##    gamma coef0     error dispersion
## 1    0.1   0.1 0.4000000  0.3258417
## 2    0.5   0.1 0.4000000  0.3258417
## 3    1.0   0.1 0.4000000  0.3258417
## 4    2.0   0.1 0.4333333  0.3351801
## 5    3.0   0.1 0.4333333  0.3351801
## 6    4.0   0.1 0.4333333  0.3351801
## 7    0.1   0.5 0.3500000  0.2539807
## 8    0.5   0.5 0.3500000  0.3464992
## 9    1.0   0.5 0.4500000  0.2944969
## 10   2.0   0.5 0.4000000  0.3258417
## 11   3.0   0.5 0.4333333  0.3351801
## 12   4.0   0.5 0.4333333  0.3351801
## 13   0.1   1.0 0.3500000  0.2539807
## 14   0.5   1.0 0.4500000  0.2944969
## 15   1.0   1.0 0.4500000  0.2944969
## 16   2.0   1.0 0.4500000  0.2944969
## 17   3.0   1.0 0.4500000  0.2944969
## 18   4.0   1.0 0.4833333  0.2986596
## 19   0.1   2.0 0.3500000  0.2539807
## 20   0.5   2.0 0.3500000  0.2539807
## 21   1.0   2.0 0.5000000  0.2484520
## 22   2.0   2.0 0.3833333  0.3147603
## 23   3.0   2.0 0.4500000  0.2944969
## 24   4.0   2.0 0.4500000  0.2944969
## 25   0.1   3.0 0.3500000  0.2539807
```

```
## 26   0.5   3.0 0.3500000  0.2539807
## 27   1.0   3.0 0.3500000  0.2539807
## 28   2.0   3.0 0.5000000  0.2484520
## 29   3.0   3.0 0.3333333  0.3333333
## 30   4.0   3.0 0.4500000  0.2944969
## 31   0.1   4.0 0.3500000  0.2539807
## 32   0.5   4.0 0.3500000  0.2539807
## 33   1.0   4.0 0.3833333  0.3147603
## 34   2.0   4.0 0.5000000  0.2484520
## 35   3.0   4.0 0.5000000  0.2484520
## 36   4.0   4.0 0.3666667  0.3496029
```

```
# Now create the best model with the best cost as selected by tune()

lenses.svm.linear.best <- lenses.svm.linear.tuned$best.model
lenses.svm.linear.best.pred <- predict(lenses.svm.linear.best, lenses.test)
confusionMatrix(lenses.svm.linear.best.pred, lenses.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 1 0 0
##          2 0 0 0
##          3 0 0 1
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.1581, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.25
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity               1.0       NA      1.0
## Specificity               1.0        1      1.0
## Pos Pred Value            1.0       NA      1.0
## Neg Pred Value            1.0       NA      1.0
## Prevalence                0.5        0      0.5
## Detection Rate            0.5        0      0.5
## Detection Prevalence      0.5        0      0.5
## Balanced Accuracy         1.0       NA      1.0
```

```
lenses.svm.polynomial.best <- lenses.svm.polynomial.tuned$best.model
lenses.svm.polynomial.best.pred <- predict(lenses.svm.polynomial.best, lenses.test)
confusionMatrix(lenses.svm.polynomial.best.pred, lenses.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction 1 2 3
##          1 1 0 0
##          2 0 0 0
##          3 0 0 1
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.1581, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.25
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity               1.0       NA      1.0
## Specificity               1.0        1      1.0
## Pos Pred Value            1.0       NA      1.0
## Neg Pred Value            1.0       NA      1.0
## Prevalence                0.5        0      0.5
## Detection Rate            0.5        0      0.5
## Detection Prevalence      0.5        0      0.5
## Balanced Accuracy         1.0       NA      1.0
```

```
lenses.svm.radial.best <- lenses.svm.radial.tuned$best.model
lenses.svm.radial.best.pred <- predict(lenses.svm.radial.best, lenses.test)
confusionMatrix(lenses.svm.radial.best.pred, lenses.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 0 0 0
##          2 0 0 0
##          3 1 0 1
##
## Overall Statistics
##
##                Accuracy : 0.5
##                  95% CI : (0.0126, 0.9874)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.75
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
```

```
## Sensitivity               0.0       NA      1.0
## Specificity               1.0        1      0.0
## Pos Pred Value            NaN       NA      0.5
## Neg Pred Value            0.5       NA      NaN
## Prevalence                0.5        0      0.5
## Detection Rate            0.0        0      0.5
## Detection Prevalence      0.0        0      1.0
## Balanced Accuracy         0.5       NA      0.5
```

```
lenses.svm.sigmoid.best <- lenses.svm.sigmoid.tuned$best.model
lenses.svm.sigmoid.best.pred <- predict(lenses.svm.sigmoid.best, lenses.test)
confusionMatrix(lenses.svm.sigmoid.best.pred, lenses.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 0 0 0
##          2 0 0 0
##          3 1 0 1
##
## Overall Statistics
##
##                Accuracy : 0.5
##                  95% CI : (0.0126, 0.9874)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.75
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity               0.0       NA      1.0
## Specificity               1.0        1      0.0
## Pos Pred Value            NaN       NA      0.5
## Neg Pred Value            0.5       NA      NaN
## Prevalence                0.5        0      0.5
## Detection Rate            0.0        0      0.5
## Detection Prevalence      0.0        0      1.0
## Balanced Accuracy         0.5       NA      0.5
```

2 - PCA - SVM

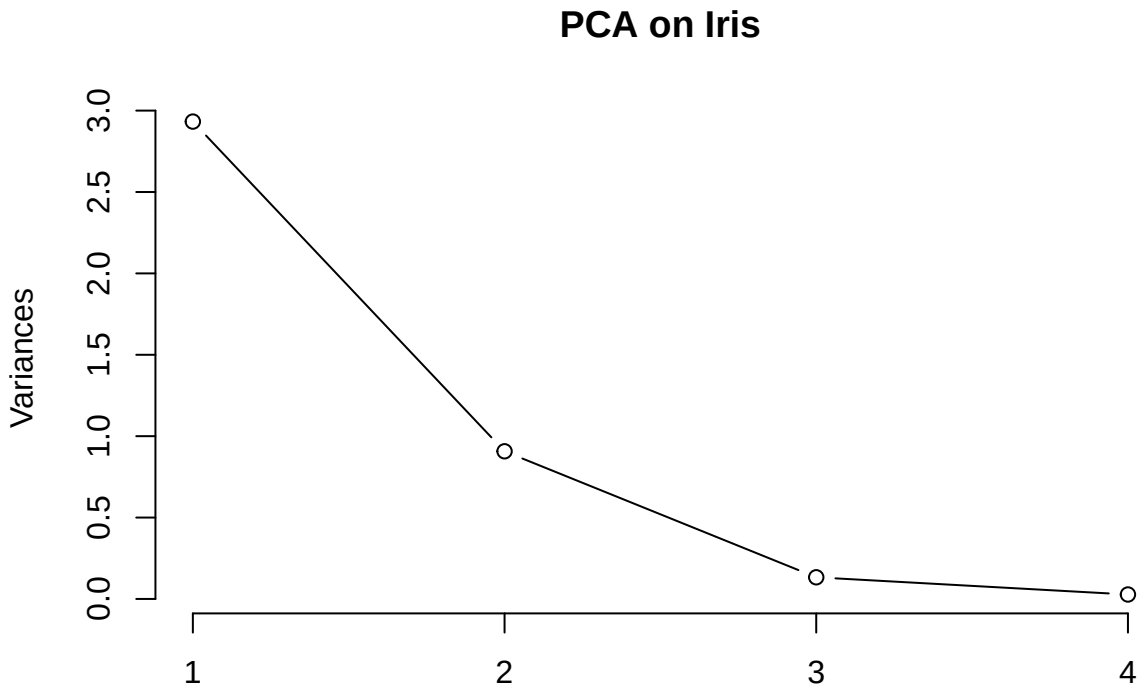
Run PCA and then run SVM on the reduced data.

Iris

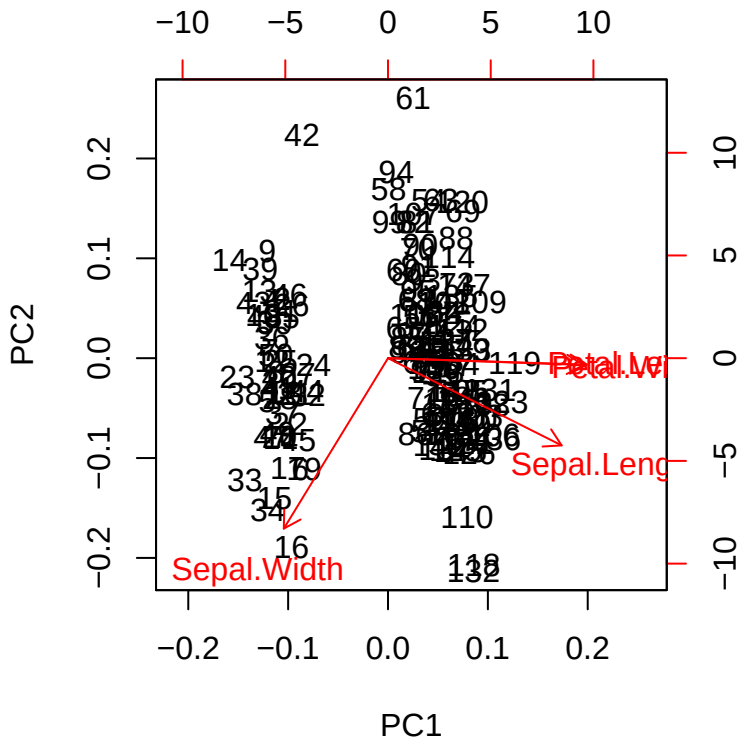
```r
iris.log <- log(iris.data)
iris.pca <- prcomp(iris.log, center=TRUE, scale. = TRUE) # do PCA analysis on iris data
summary(iris.pca)
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.7125 0.9524 0.36470 0.16568
## Proportion of Variance 0.7331 0.2268 0.03325 0.00686
## Cumulative Proportion  0.7331 0.9599 0.99314 1.00000
```

```r
plot(iris.pca, main="PCA on Iris", type="l")                # plot PCA comparison
```




```r
biplot(iris.pca)
```




```
# SVM on reduced set

iris.training.reduced <- cbind.data.frame(iris.pca$x[ind.iris==1, c(1,2)], Species=iris.trainLabels) #
iris.test.reduced <- cbind.data.frame(iris.pca$x[ind.iris==2, c(1,2)], Species=iris.testLabels)#reduced

iris.svm.polynomial.reduced.tuned <- tune.svm(Species~., data=iris.training.reduced, kernel="polynomial
summary(iris.svm.polynomial.reduced.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  degree coef0
##       4     1
##
## - best performance: 0.1177778
##
## - Detailed performance results:
##    degree coef0     error dispersion
## 1       3   0.1 0.1688889 0.17652335
## 2       4   0.1 0.1800000 0.17745341
## 3       5   0.1 0.1900000 0.16207248
## 4       3   0.5 0.1688889 0.17652335
## 5       4   0.5 0.1588889 0.13709208
```

```
## 6       5   0.5 0.1688889 0.12752794
## 7       3   1.0 0.1477778 0.15135873
## 8       4   1.0 0.1177778 0.13159359
## 9       5   1.0 0.1477778 0.12744724
## 10      3   2.0 0.1777778 0.14083817
## 11      4   2.0 0.1366667 0.08607427
## 12      5   2.0 0.1366667 0.08607427
## 13      3   3.0 0.1677778 0.15021247
## 14      4   3.0 0.1255556 0.08081376
## 15      5   3.0 0.1477778 0.07258692
## 16      3   4.0 0.1577778 0.15115468
## 17      4   4.0 0.1366667 0.08607427
## 18      5   4.0 0.1477778 0.07258692
```

```
# Now create polynomial SVM with optimal parameters

iris.svm.polynomial.reduced.best <- iris.svm.polynomial.reduced.tuned$best.model
iris.svm.polynomial.reduced.best.pred <- predict(iris.svm.polynomial.reduced.best, iris.test.reduced)

confusionMatrix(iris.svm.polynomial.reduced.best.pred, iris.testLabels)  # Confusion matrix
```

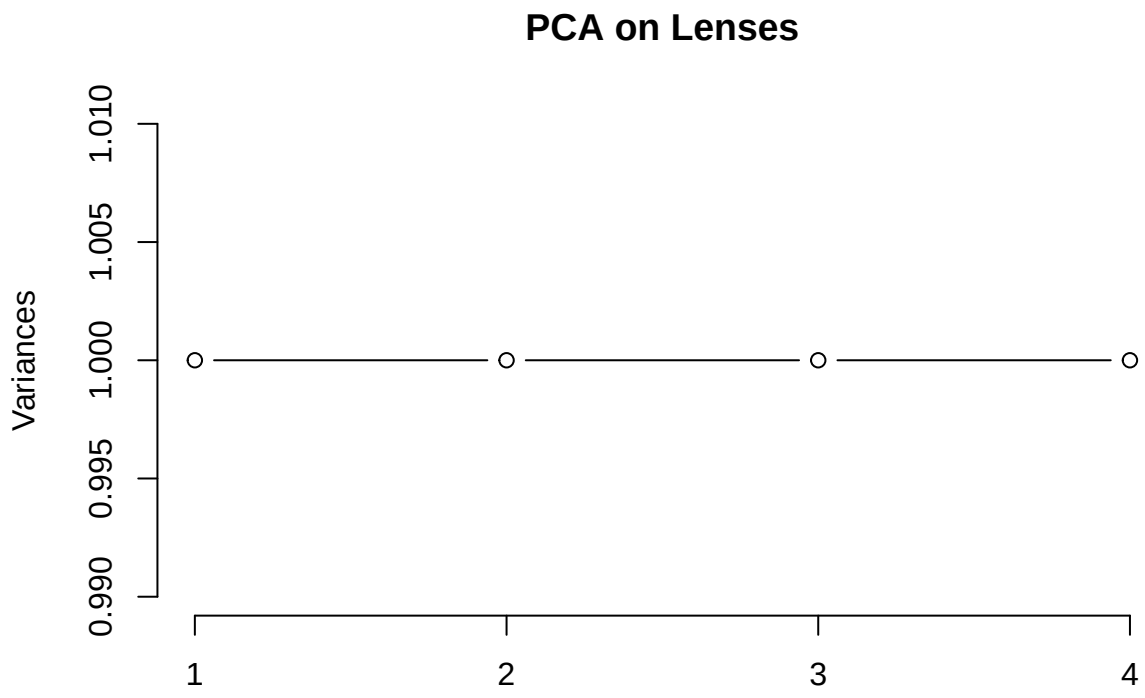
```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         11         4
##   virginica       0          2        18
##
## Overall Statistics
##
##                Accuracy : 0.8909
##                  95% CI : (0.7775, 0.9589)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : 4.653e-14
##
##                   Kappa : 0.8342
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.8462           0.8182
## Specificity                 1.0000            0.9048           0.9394
## Pos Pred Value              1.0000            0.7333           0.9000
## Neg Pred Value              1.0000            0.9500           0.8857
## Prevalence                  0.3636            0.2364           0.4000
## Detection Rate              0.3636            0.2000           0.3273
## Detection Prevalence        0.3636            0.2727           0.3636
## Balanced Accuracy           1.0000            0.8755           0.8788
```

Contact Lenses

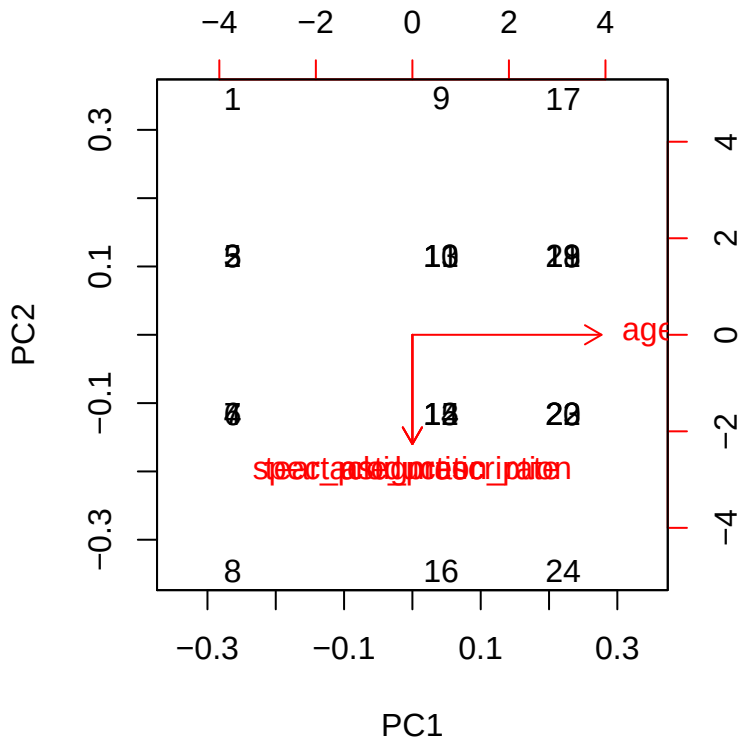
```
lenses.log <- log(lenses.data)
lenses.pca <- prcomp(lenses.log, center=TRUE, scale. = TRUE) # do PCA analysis on iris data
summary(lenses.pca)
```

```
## Importance of components:
##                          PC1  PC2  PC3  PC4
## Standard deviation     1.00 1.00 1.00 1.00
## Proportion of Variance 0.25 0.25 0.25 0.25
## Cumulative Proportion  0.25 0.50 0.75 1.00
```

```
plot(lenses.pca, main="PCA on Lenses", type="l")                # plot PCA comparison
```




```
biplot(lenses.pca)
```




```
lenses.training.reduced <- cbind.data.frame(lenses.pca$x[ind.lenses==1, c(1,2,3)], class=lenses.trainLa
lenses.test.reduced <- cbind.data.frame(lenses.pca$x[ind.lenses==2, c(1,2, 3)], class=lenses.testLabels

lenses.svm.polynomial.reduced.tuned <- tune.svm(class~., data=lenses.training.reduced, kernel="polynomi
summary(lenses.svm.polynomial.reduced.tuned)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  degree coef0
##       3   0.1
##
## - best performance: 0.3333333
##
## - Detailed performance results:
##    degree coef0     error dispersion
## 1       3   0.1 0.3333333  0.3767961
## 2       4   0.1 0.3833333  0.3604695
## 3       5   0.1 0.3833333  0.3604695
## 4       3   0.5 0.4333333  0.4097575
## 5       4   0.5 0.4333333  0.4097575
## 6       5   0.5 0.4833333  0.3804643
```

```
## 7       3   1.0 0.4833333  0.3804643
## 8       4   1.0 0.4833333  0.3804643
## 9       5   1.0 0.4833333  0.3804643
## 10      3   2.0 0.4833333  0.3804643
## 11      4   2.0 0.4833333  0.3804643
## 12      5   2.0 0.4833333  0.3804643
## 13      3   3.0 0.4833333  0.3804643
## 14      4   3.0 0.4833333  0.3804643
## 15      5   3.0 0.4833333  0.3804643
## 16      3   4.0 0.4833333  0.3804643
## 17      4   4.0 0.4833333  0.3804643
## 18      5   4.0 0.4833333  0.3804643
```

```
# Now create polynomial SVM with optimal parameters

lenses.svm.polynomial.reduced.best <- lenses.svm.polynomial.reduced.tuned$best.model
lenses.svm.polynomial.reduced.best.pred <- predict(lenses.svm.polynomial.reduced.best, lenses.test.redu

confusionMatrix(lenses.svm.polynomial.reduced.best.pred, lenses.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 1 0 0
##          2 0 0 0
##          3 0 0 1
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.1581, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.25
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity               1.0       NA      1.0
## Specificity               1.0        1      1.0
## Pos Pred Value            1.0       NA      1.0
## Neg Pred Value            1.0       NA      1.0
## Prevalence                0.5        0      0.5
## Detection Rate            0.5        0      0.5
## Detection Prevalence      0.5        0      0.5
## Balanced Accuracy         1.0       NA      1.0
```

3 - Random Forest

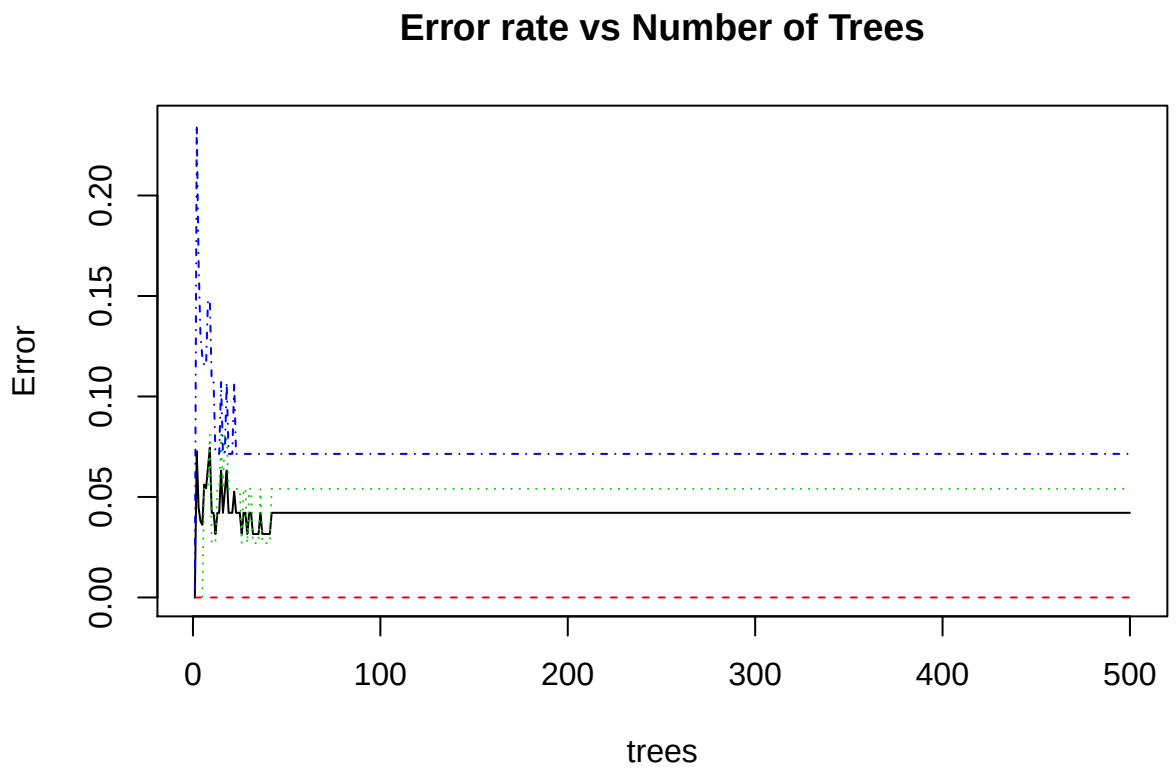
How did the boosting or bagging compare to the J48 results from Project 1?

Iris

```
set.seed(1234)
iris.rf <- randomForest(Species~., data=iris.trainingWithLabels)
print(iris.rf)
```

```
##
## Call:
##  randomForest(formula = Species ~ ., data = iris.trainingWithLabels)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 4.21%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         30          0         0  0.00000000
## versicolor      0         35         2  0.05405405
## virginica       0          2        26  0.07142857
```

```
plot(iris.rf, main="Error rate vs Number of Trees")
```



```r
which.min(iris.rf$err.rate[,1])     # this returns how mamy trees are needed
```

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

```r
iris.rf.ntree <- randomForest(Species~., data=iris.trainingWithLabels,
                              ntree = which.min(iris.rf$err.rate[,1]) ) # specify the number of trees t

print(iris.rf.ntree)
```

```
##
## Call:
##  randomForest(formula = Species ~ ., data = iris.trainingWithLabels,      ntree = which.min(iris.rf$
##                Type of random forest: classification
##                      Number of trees: 1
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 8.33%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         11          0         0   0.0000000
## versicolor      0         14         0   0.0000000
## virginica       0          3         8   0.2727273
```

```r
iris.rf.ntree.pred <- predict(iris.rf.ntree, newdata = iris.test)
confusionMatrix(iris.rf.ntree.pred, iris.testLabels)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         12         2
##   virginica       0          1        20
##
## Overall Statistics
##
##                Accuracy : 0.9455
##                  95% CI : (0.8488, 0.9886)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9167
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9231           0.9091
## Specificity                 1.0000            0.9524           0.9697
## Pos Pred Value              1.0000            0.8571           0.9524
## Neg Pred Value              1.0000            0.9756           0.9412
```

```
## Prevalence                 0.3636           0.2364           0.4000
## Detection Rate             0.3636           0.2182           0.3636
## Detection Prevalence       0.3636           0.2545           0.3818
## Balanced Accuracy          1.0000           0.9377           0.9394
```

```
# J48 (from Project 1) to compare with Random Forest

weka_j48 <- make_Weka_classifier("weka/classifiers/trees/J48")
# non-prunned version of J48 tree
iris.j48 <- weka_j48(Species~., data=iris.trainingWithLabels, control=Weka_control(U=TRUE))
evaluate_Weka_classifier(iris.j48, newdata = iris.test, class=TRUE)
```

```
##
## === Summary ===
##
## Correctly Classified Instances          51               92.7273 %
## Incorrectly Classified Instances         4                7.2727 %
## Kappa statistic                          0.8893
## Mean absolute error                      0.0521
## Root mean squared error                  0.216
## Relative absolute error                 11.9789 %
## Root relative squared error             46.3404 %
## Total Number of Instances               55
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
##                  0.950    0.000    1.000      0.950    0.974      0.961    0.975     0.968     setosa
##                  0.923    0.071    0.800      0.923    0.857      0.812    0.926     0.757     versi
##                  0.909    0.030    0.952      0.909    0.930      0.886    0.956     0.921     virgi
## Weighted Avg.    0.927    0.029    0.934      0.927    0.929      0.896    0.956     0.899
##
## === Confusion Matrix ===
##
##    a  b  c   <-- classified as
##   19  1  0 |  a = setosa
##    0 12  1 |  b = versicolor
##    0  2 20 |  c = virginica
```

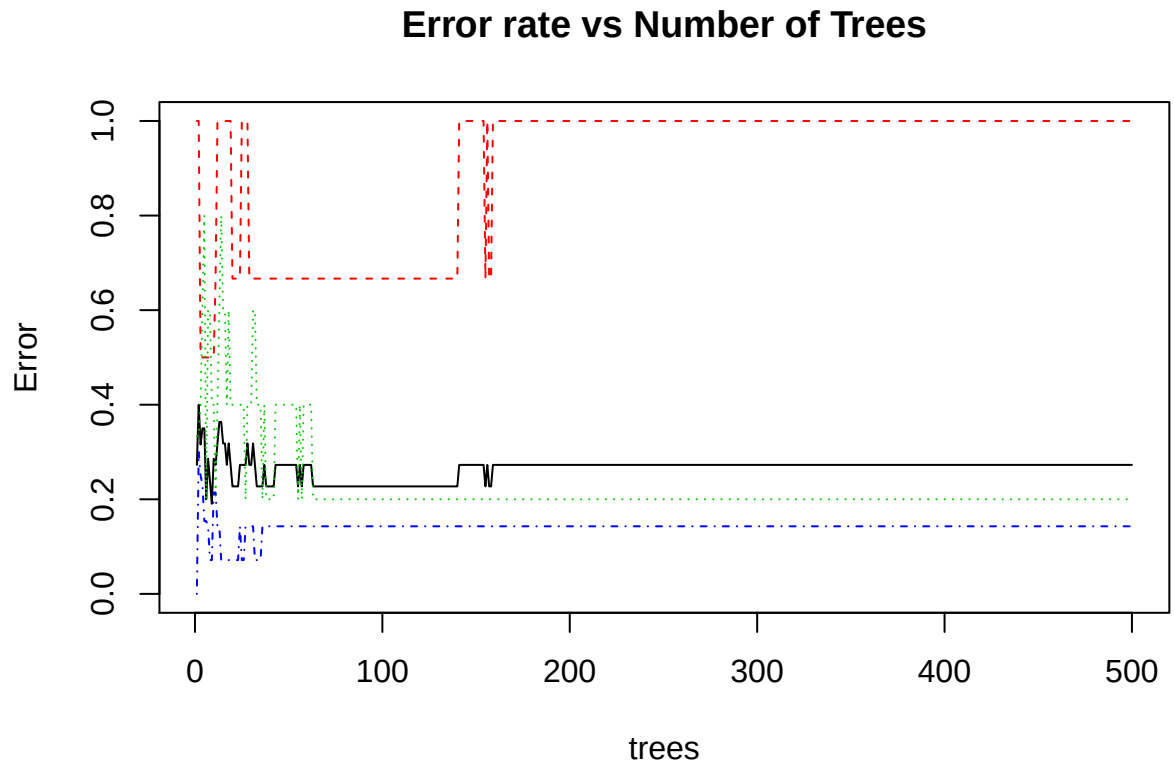
Contact Lenses

```
set.seed(1234)
lenses.rf <- randomForest(class~., data=lenses.trainingWithLabels)
print(lenses.rf)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = lenses.trainingWithLabels)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
```

```
##
##         OOB estimate of  error rate: 27.27%
## Confusion matrix:
##   1 2  3 class.error
## 1 0 0  3   1.0000000
## 2 0 4  1   0.2000000
## 3 1 1 12   0.1428571
```

```
plot(lenses.rf, main="Error rate vs Number of Trees")
```




```
which.min(lenses.rf$err.rate[,1])     # this returns how mamy trees are needed
```

```
## [1] 9
```

```
lenses.rf.ntree <- randomForest(class~., data=lenses.trainingWithLabels,
                                ntree = which.min(lenses.rf$err.rate[,1]) ) # specify the number of trees

print(lenses.rf.ntree)
```

```
##
## Call:
##  randomForest(formula = class ~ ., data = lenses.trainingWithLabels,      ntree = which.min(lenses.r
##                Type of random forest: classification
##                      Number of trees: 9
```

```
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 33.33%
## Confusion matrix:
##   1 2  3 class.error
## 1 1 0  2   0.6666667
## 2 0 2  3   0.6000000
## 3 1 1 11   0.1538462
```

```
lenses.rf.ntree.pred <- predict(lenses.rf.ntree, newdata = lenses.test)
confusionMatrix(lenses.rf.ntree.pred, lenses.testLabels)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 1 0 0
##          2 0 0 0
##          3 0 0 1
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.1581, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.25
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity               1.0       NA      1.0
## Specificity               1.0        1      1.0
## Pos Pred Value            1.0       NA      1.0
## Neg Pred Value            1.0       NA      1.0
## Prevalence                0.5        0      0.5
## Detection Rate            0.5        0      0.5
## Detection Prevalence      0.5        0      0.5
## Balanced Accuracy         1.0       NA      1.0
```

```
# J48 (from Project 1) to compare with Random Forest

weka_j48 <- make_Weka_classifier("weka/classifiers/trees/J48")
# non-prunned version of J48 tree
lenses.j48 <- weka_j48(class~., data=lenses, control=Weka_control(U=TRUE))
evaluate_Weka_classifier(lenses.j48, numFolds = 3, class=TRUE)
```

```
## === 3 Fold Cross Validation ===
##
## === Summary ===
##
```

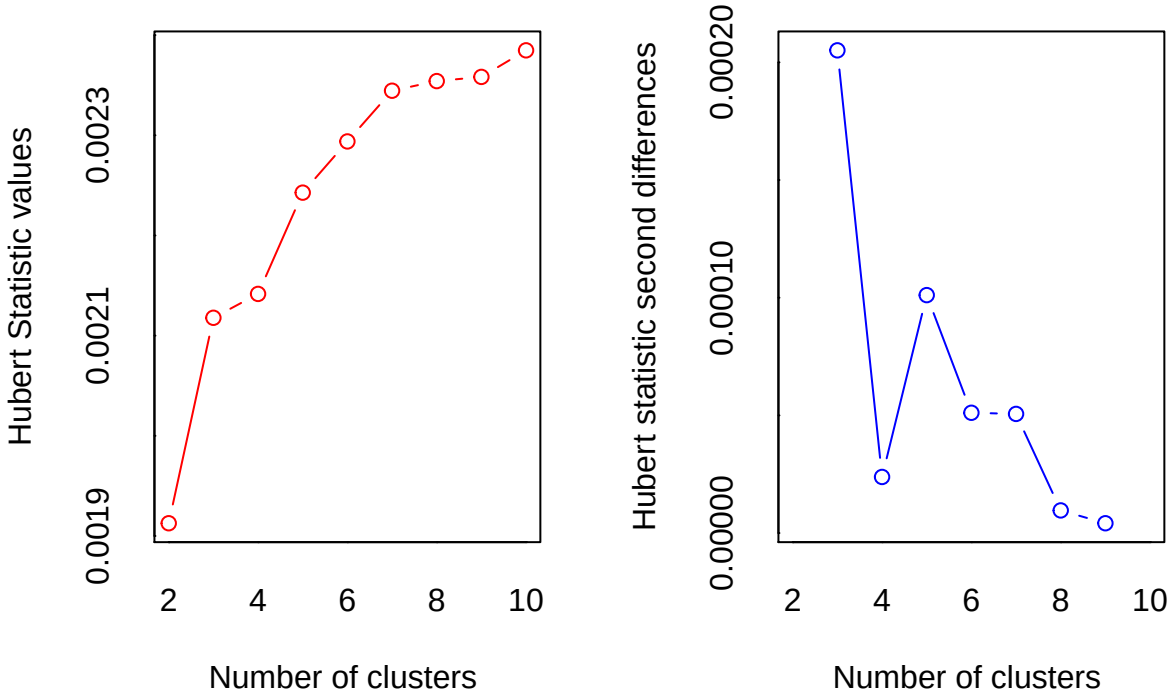
```
## Correctly Classified Instances          20               83.3333 %
## Incorrectly Classified Instances         4               16.6667 %
## Kappa statistic                          0.71
## Mean absolute error                      0.1398
## Root mean squared error                  0.3099
## Relative absolute error                 37.3568 %
## Root relative squared error             72.4701 %
## Total Number of Instances               24
##
## === Detailed Accuracy By Class ===
##
##                  TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
##                  0.750    0.100    0.600      0.750    0.667      0.596    0.819     0.467     1
##                  1.000    0.053    0.833      1.000    0.909      0.889    0.958     0.750     2
##                  0.800    0.111    0.923      0.800    0.857      0.669    0.848     0.881     3
## Weighted Avg.    0.833    0.097    0.851      0.833    0.836      0.703    0.866     0.785
##
## === Confusion Matrix ===
##
##   a  b  c   <-- classified as
##   3  0  1 |  a = 1
##   0  5  0 |  b = 2
##   2  1 12 |  c = 3
```

4 - Clustering (k-means) / Decision Tree / SVM

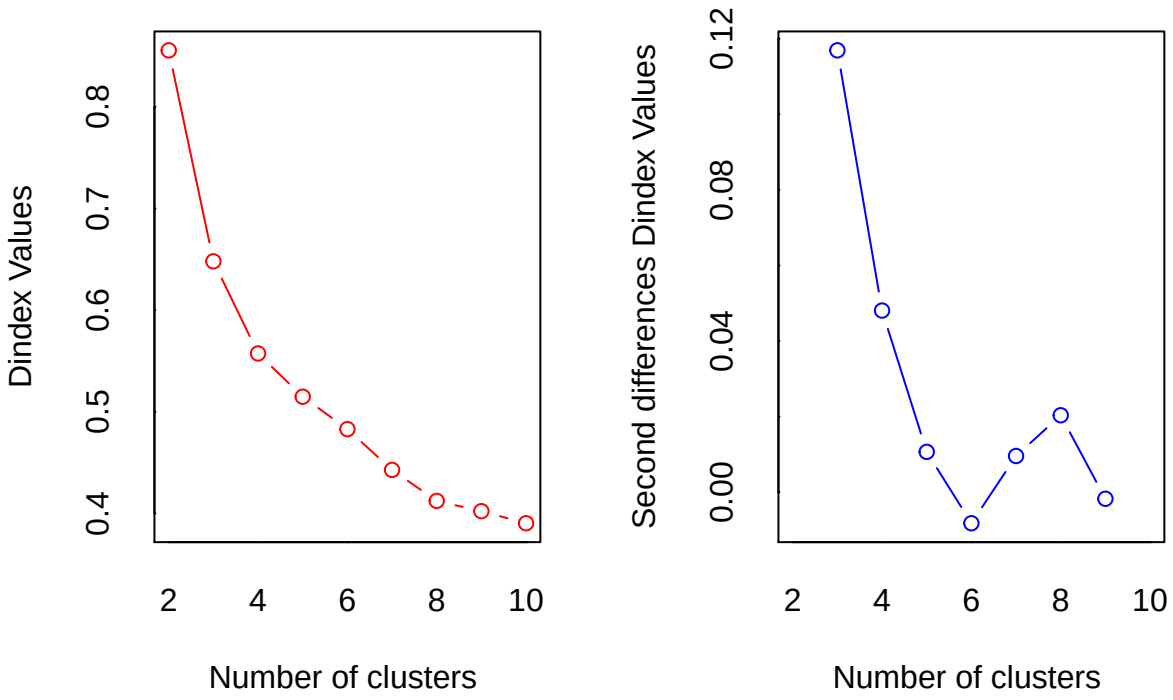
Run clustering (k-means) ant then apply decision tree and SVM on clustered data.

Iris

```
NbClust(iris.data,          # using the complete set with no labels
        min.nc = 2,         # minimum number of clusters
        max.nc = 10,        # maximum number of clusters
        method = "kmeans")
```




```
## *** : The Hubert index is a graphical method of determining the number of clusters.
##                 In the plot of Hubert index, we seek a significant knee that corresponds to a
##                 significant increase of the value of the measure i.e the significant peak in Hubert
##                 index second differences plot.
##
```

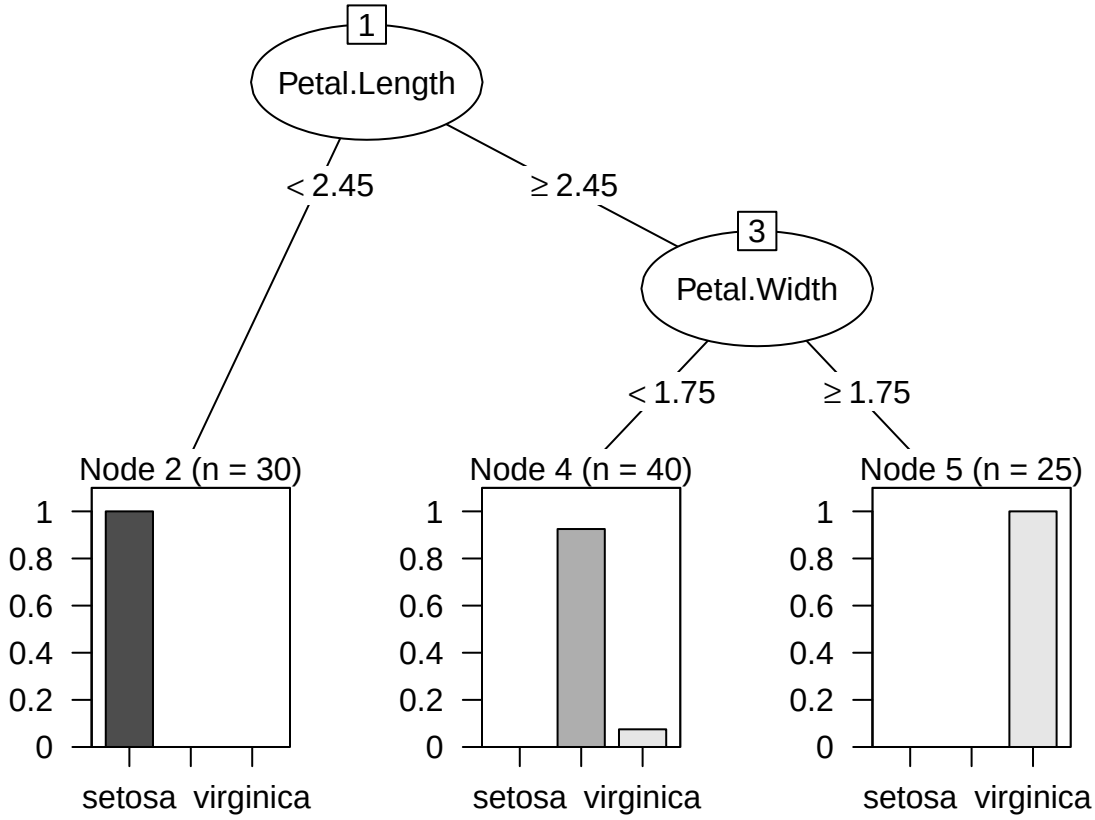



```
## *** : The D index is a graphical method of determining the number of clusters.
##                 In the plot of D index, we seek a significant knee (the significant peak in Dindex
##                 second differences plot) that corresponds to a significant increase of the value of
##                 the measure.
##
## *******************************************************************
## * Among all indices:
## * 10 proposed 2 as the best number of clusters
## * 8 proposed 3 as the best number of clusters
## * 2 proposed 4 as the best number of clusters
## * 1 proposed 5 as the best number of clusters
## * 1 proposed 7 as the best number of clusters
## * 1 proposed 8 as the best number of clusters
## * 1 proposed 10 as the best number of clusters
##
##                   ***** Conclusion *****
##
## * According to the majority rule, the best number of clusters is  2
##
##
## *******************************************************************

## $All.index
##        KL       CH Hartigan     CCC    Scott  Marriot    TrCovW   TraceW
## 2  5.9068 513.9245 137.9491 35.9428 1044.605 467371.6 1045.9696 152.3480
```

```
## 3  3.5663 561.6278  55.5419 37.6701 1246.668 273408.6  248.9814  78.8514
## 4  7.2495 530.7658  21.7032 36.4682 1359.280 229428.4  173.8973  57.2285
## 5  0.4117 459.5058  25.1578 34.3409 1415.662 246163.9  140.6758  49.8223
## 6  0.6156 433.4067  31.8988 33.3747 1502.213 199064.5   87.7420  42.4561
## 7  1.6869 443.3948  22.7349 33.4645 1583.366 157734.8   79.7038  34.7567
## 8  5.3825 440.6205   6.2958 33.2318 1653.207 129330.2   49.7924  29.9889
## 9  1.3278 400.5825   9.3700 31.9307 1683.279 133948.2   38.7655  28.7158
## 10 2.2038 378.0763   7.6187 31.1083 1712.825 135802.5   35.2030  26.9264
##     Friedman    Rubin Cindex     DB Silhouette   Duda Pseudot2   Beale
## 2   732.8086  62.6152 0.2728 0.4744     0.6810 1.9253 -52.8667 -1.1380
## 3   801.6490 120.9780 0.3450 0.7256     0.5528 1.1915  -9.3224 -0.3776
## 4   874.3981 166.6878 0.3211 0.8436     0.4981 0.5112  45.9014  2.2615
## 5   936.2996 191.4664 0.3327 0.9987     0.3728 1.1340  -5.1981 -0.2746
## 6  1033.8843 224.6862 0.3594 1.0923     0.3263 0.8469   5.2430  0.4175
## 7  1173.6099 274.4586 0.3965 1.0070     0.3462 3.9365 -17.9033 -1.5008
## 8  1289.5807 318.0936 0.4007 1.0403     0.3519 0.9269   1.5780  0.1799
## 9  1381.7100 332.1968 0.3919 1.0573     0.3536 0.7926   6.5424  0.6084
## 10 1443.8573 354.2725 0.3831 1.0993     0.3114 1.0204  -0.5210 -0.0452
##    Ratkowsky    Ball Ptbiserial    Frey McClain   Dunn Hubert SDindex
## 2     0.5462 76.1740     0.8345  1.7571  0.2723 0.0765 0.0019  1.2820
## 3     0.4967 26.2838     0.7146  1.5949  0.5255 0.0988 0.0021  1.7259
## 4     0.4413 14.3071     0.6361  6.0985  0.7120 0.1365 0.0021  2.4707
## 5     0.4067  9.9645     0.5521  1.3762  0.9903 0.0624 0.0022  3.1993
## 6     0.3737  7.0760     0.5023 -0.1138  1.2099 0.0739 0.0023  3.3704
## 7     0.3498  4.9652     0.5119  1.1261  1.1407 0.0872 0.0023  3.4409
## 8     0.3302  3.7486     0.4690  1.3658  1.3416 0.0974 0.0024  3.8586
## 9     0.3131  3.1906     0.4567  3.0876  1.4108 0.0974 0.0024  4.4503
## 10    0.2989  2.6926     0.4249  0.6407  1.6368 0.0974 0.0024  4.9241
##    Dindex   SDbw
## 2  0.8556 0.1618
## 3  0.6480 0.2257
## 4  0.5574 0.3186
## 5  0.5148 0.1542
## 6  0.4829 0.1158
## 7  0.4428 0.1341
## 8  0.4123 0.0713
## 9  0.4022 0.0612
## 10 0.3904 0.0311
##
## $All.CriticalValues
##    CritValue_Duda CritValue_PseudoT2 Fvalue_Beale
## 2          0.5633            85.2756       1.0000
## 3          0.5131            55.0440       1.0000
## 4          0.5551            38.4707       0.0640
## 5          0.4590            51.8597       1.0000
## 6          0.4284            38.6922       0.7956
## 7          0.0772           287.0296       1.0000
## 8          0.3773            33.0071       0.9481
## 9          0.4590            29.4657       0.6575
## 10         0.3357            51.4462       1.0000
##
## $Best.nc
##                       KL       CH Hartigan     CCC    Scott  Marriot
## Number_clusters 4.0000   3.0000   3.0000  3.0000   3.0000      3.0
```

```
## Value_Index     7.2495 561.6278  82.4072 37.6701 202.0631 149982.9
##                    TrCovW  TraceW Friedman    Rubin Cindex     DB
## Number_clusters   3.0000  3.0000   7.0000   8.0000 2.0000 2.0000
## Value_Index     796.9882 51.8735 139.7256 -29.5317 0.2728 0.4744
##                 Silhouette   Duda PseudoT2  Beale Ratkowsky    Ball
## Number_clusters      2.000 2.0000   2.0000  2.000    2.0000  3.0000
## Value_Index          0.681 1.9253 -52.8667 -1.138    0.5462 49.8902
##                 PtBiserial   Frey McClain   Dunn Hubert SDindex Dindex
## Number_clusters     2.0000 5.0000  2.0000 4.0000      0   2.000      0
## Value_Index         0.8345 1.3762  0.2723 0.1365      0   1.282      0
##                    SDbw
## Number_clusters 10.0000
## Value_Index      0.0311
##
## $Best.partition
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [36] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2
##  [71] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2
## [106] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
## [141] 2 2 2 2 2 2 2 2 2 2
```

```
## Two centroids

iris.kmeans.2 <- kmeans(iris.data, centers = 2) # two centroids as suggested by NbClust
iris.with.kmeans.2 <- cbind.data.frame(iris, Cluster = iris.kmeans.2$cluster) # add cluster feature
iris.with.kmeans.2.training = iris.with.kmeans.2[ind.iris==1,]
iris.with.kmeans.2.test = iris.with.kmeans.2[ind.iris==2,]

## Decision tree on clustered data
iris.kmeans.2.dtree <- rpart(Species~., data = iris.with.kmeans.2.training)
plot(as.party(iris.kmeans.2.dtree))
```

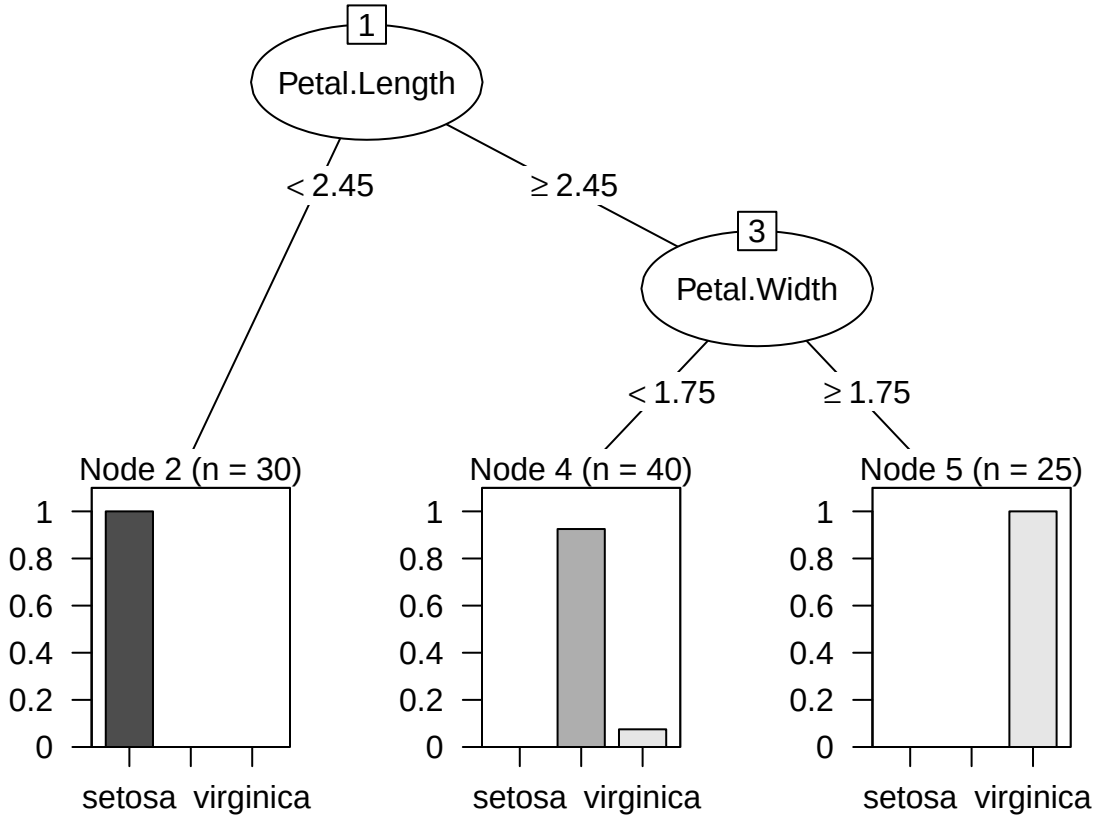



```
## SVM on clustered data

# tune svm once more
iris.kmeans.2.svm.polynomial.tuned <- tune.svm(Species~., data=iris.with.kmeans.2.training, kernel="poly
# select best model
iris.kmeans.2.svm.polynomial.best <- iris.kmeans.2.svm.polynomial.tuned$best.model
iris.kmeans.2.svm.polynomial.best.pred <- predict(iris.kmeans.2.svm.polynomial.best, iris.with.kmeans.2

confusionMatrix(iris.kmeans.2.svm.polynomial.best.pred, iris.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         13         2
##   virginica       0          0        20
##
## Overall Statistics
##
##                Accuracy : 0.9636
##                  95% CI : (0.8747, 0.9956)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9447
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9091
## Specificity                 1.0000            0.9524           1.0000
## Pos Pred Value              1.0000            0.8667           1.0000
## Neg Pred Value              1.0000            1.0000           0.9429
## Prevalence                  0.3636            0.2364           0.4000
## Detection Rate              0.3636            0.2364           0.3636
## Detection Prevalence        0.3636            0.2727           0.3636
## Balanced Accuracy           1.0000            0.9762           0.9545
```

```
## Three centroids

iris.kmeans.3 = kmeans(iris.data, centers = 3) # three centroids as data is really 3 classes
iris.with.kmeans.3 <- cbind.data.frame(iris.data, Cluster = iris.kmeans.3$cluster)

iris.with.kmeans.3 <- cbind.data.frame(iris, Cluster = iris.kmeans.3$cluster) # add cluster feature
iris.with.kmeans.3.training = iris.with.kmeans.3[ind.iris==1,]
iris.with.kmeans.3.test = iris.with.kmeans.3[ind.iris==2,]

## Decision tree on clustered data
iris.kmeans.3.dtree <- rpart(Species~., data = iris.with.kmeans.3.training)
plot(as.party(iris.kmeans.3.dtree))
```

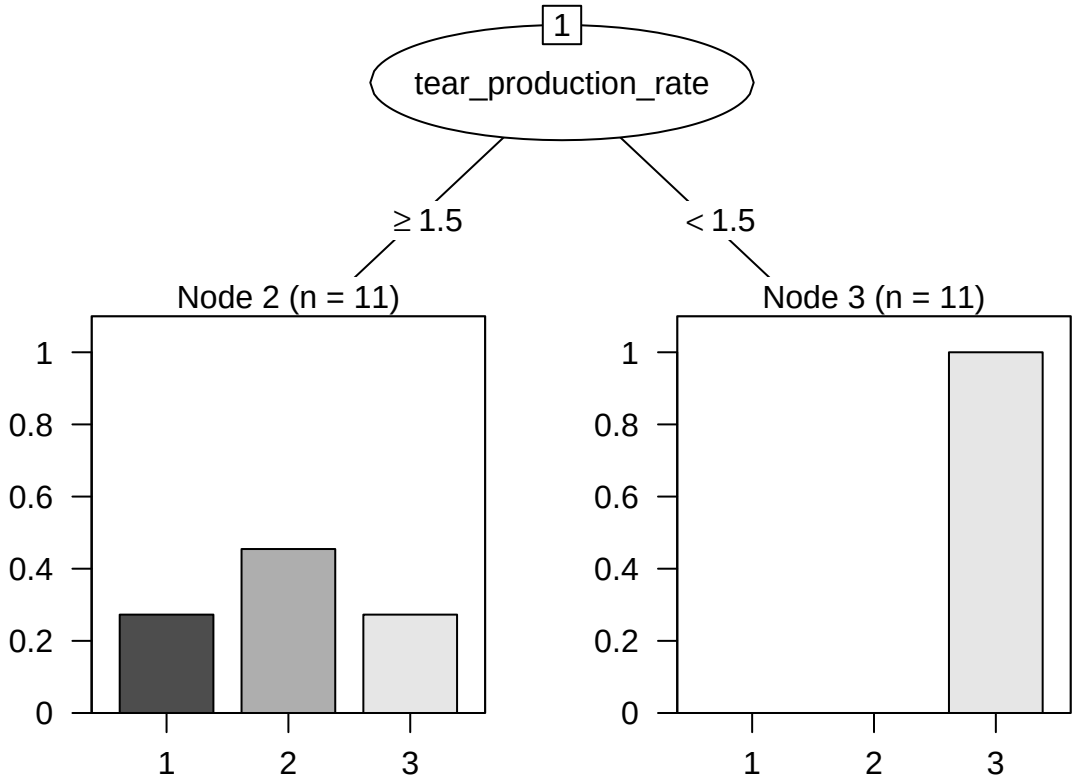


```
## SVM on clustered data

# tune svm once more
iris.kmeans.3.svm.polynomial.tuned <- tune.svm(Species~., data=iris.with.kmeans.3.training, kernel="poly
# select best model
iris.kmeans.3.svm.polynomial.best <- iris.kmeans.3.svm.polynomial.tuned$best.model
iris.kmeans.3.svm.polynomial.best.pred <- predict(iris.kmeans.3.svm.polynomial.best, iris.with.kmeans.3

confusionMatrix(iris.kmeans.3.svm.polynomial.best.pred, iris.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         20          0         0
##   versicolor      0         13         2
##   virginica       0          0        20
##
## Overall Statistics
##
##                Accuracy : 0.9636
##                  95% CI : (0.8747, 0.9956)
##     No Information Rate : 0.4
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9447
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            1.0000           0.9091
## Specificity                 1.0000            0.9524           1.0000
## Pos Pred Value              1.0000            0.8667           1.0000
## Neg Pred Value              1.0000            1.0000           0.9429
## Prevalence                  0.3636            0.2364           0.4000
## Detection Rate              0.3636            0.2364           0.3636
## Detection Prevalence        0.3636            0.2727           0.3636
## Balanced Accuracy           1.0000            0.9762           0.9545
```

Contact Lenses

```
## Can't apply NbClust for this dataset, will use 2 and 3 centroids as with iris
## Two centroids
lenses.kmeans.2 <- kmeans(lenses.data, centers = 2)
lenses.with.kmeans.2 <- cbind.data.frame(lenses, cluster=lenses.kmeans.2$cluster)
lenses.with.kmeans.2.training = lenses.with.kmeans.2[ind.lenses==1,]
lenses.with.kmeans.2.test = lenses.with.kmeans.2[ind.lenses==2,]

## Decision tree on clustered data
lenses.kmeans.2.dtree <- rpart(class~., data = lenses.with.kmeans.2.training)
plot(as.party(lenses.kmeans.2.dtree))
```

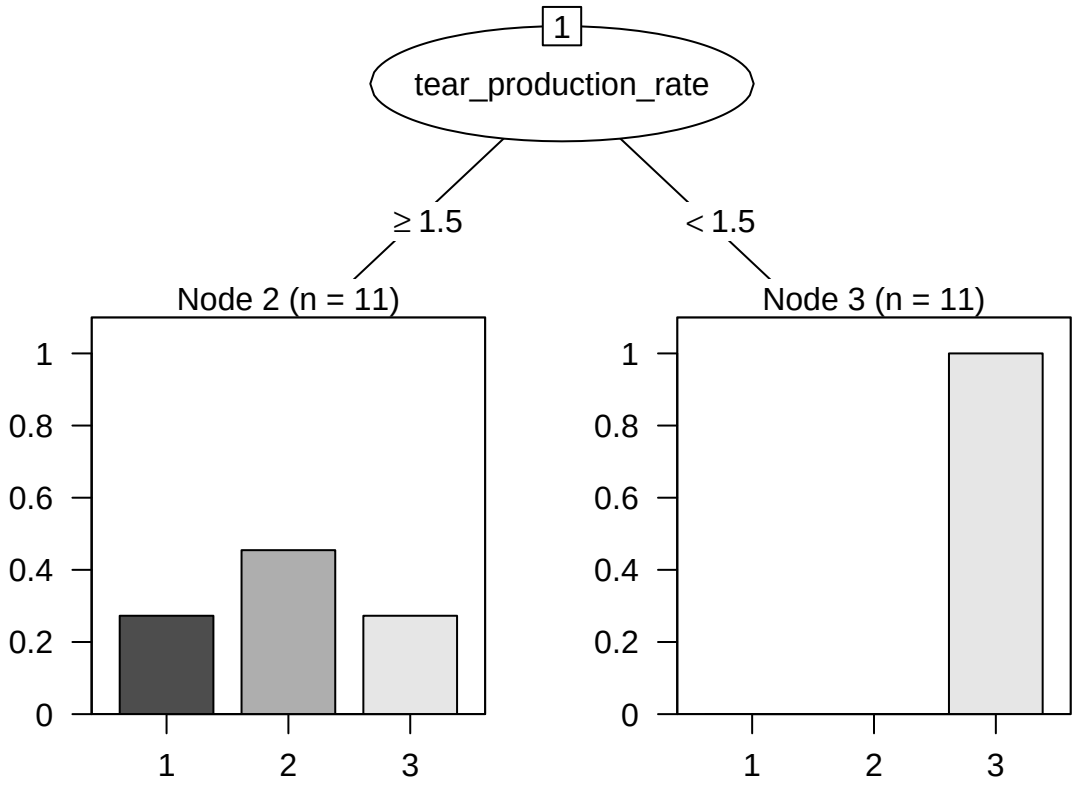



```
## SVM on clustered data

# tune svm once more
lenses.kmeans.2.svm.polynomial.tuned <- tune.svm(class~., data=lenses.with.kmeans.2.training, kernel="p
# select best model
lenses.kmeans.2.svm.polynomial.best <- lenses.kmeans.2.svm.polynomial.tuned$best.model
lenses.kmeans.2.svm.polynomial.best.pred <- predict(lenses.kmeans.2.svm.polynomial.best, lenses.with.km

confusionMatrix(lenses.kmeans.2.svm.polynomial.best.pred, lenses.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 1 0 0
##          2 0 0 0
##          3 0 0 1
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.1581, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.25
##
##                   Kappa : 1
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity               1.0       NA      1.0
## Specificity               1.0        1      1.0
## Pos Pred Value            1.0       NA      1.0
## Neg Pred Value            1.0       NA      1.0
## Prevalence                0.5        0      0.5
## Detection Rate            0.5        0      0.5
## Detection Prevalence      0.5        0      0.5
## Balanced Accuracy         1.0       NA      1.0
```

```
## Three centroids
lenses.kmeans.3 <- kmeans(lenses.data, centers = 3)
lenses.with.kmeans.3 <- cbind.data.frame(lenses, cluster=lenses.kmeans.3$cluster)
lenses.with.kmeans.3.training = lenses.with.kmeans.3[ind.lenses==1,]
lenses.with.kmeans.3.test = lenses.with.kmeans.3[ind.lenses==2,]

## Decision tree on clustered data
lenses.kmeans.3.dtree <- rpart(class~., data = lenses.with.kmeans.3.training)
plot(as.party(lenses.kmeans.3.dtree))
```



```
## SVM on clustered data
# tune svm once more
lenses.kmeans.3.svm.polynomial.tuned <- tune.svm(class~., data=lenses.with.kmeans.3.training, kernel="p
# select best model
lenses.kmeans.3.svm.polynomial.best <- lenses.kmeans.3.svm.polynomial.tuned$best.model
lenses.kmeans.3.svm.polynomial.best.pred <- predict(lenses.kmeans.3.svm.polynomial.best, lenses.with.km

confusionMatrix(lenses.kmeans.3.svm.polynomial.best.pred, lenses.testLabels)  # Confusion matrix
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2 3
##          1 1 0 0
##          2 0 0 0
##          3 0 0 1
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.1581, 1)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.25
##
##                   Kappa : 1
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3
## Sensitivity               1.0       NA      1.0
## Specificity               1.0        1      1.0
## Pos Pred Value            1.0       NA      1.0
## Neg Pred Value            1.0       NA      1.0
## Prevalence                0.5        0      0.5
## Detection Rate            0.5        0      0.5
## Detection Prevalence      0.5        0      0.5
## Balanced Accuracy         1.0       NA      1.0
```