Exercise Sheet 8 – Data Mining Wirtschaftsinformatik, HTW Berlin

Martin Spott

last revision 26 May 2020

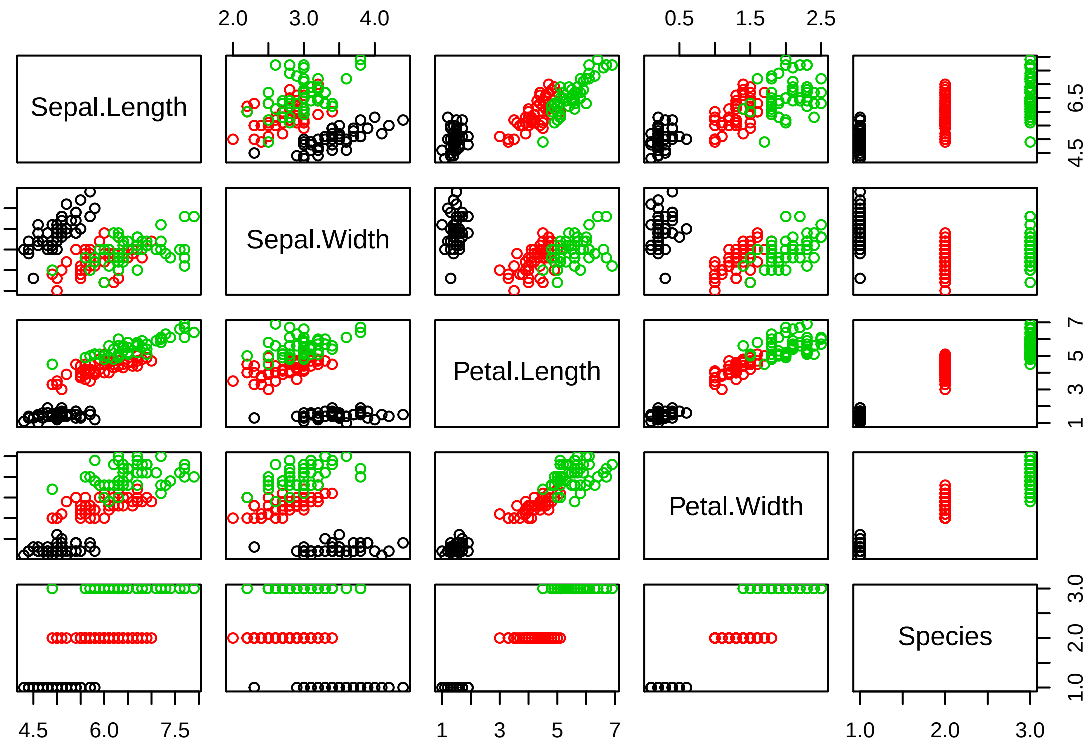
This exercise is about learning k-Nearest-Neighbour classifiers and Support Vector Machines (SVMs). We will use the Iris data set.

First load some libraries and attach the data.

```
# load some libraries; install first if necessary
library(e1071) # for SVMs
library(caret) # for learning control
data("iris")
```

Some data preparation:

```
# look at the Iris data set
plot(iris, col=iris$Species)
```




```
# test and training data sets
n <- nrow(iris)

train_indices <- sample(1:n, round(2/3 * n))
iris_train <- iris[train_indices,]
iris_test <- iris[-train_indices,]
```

```r
# for illustration train on two input features only
# pick two such that linear separation works well
features <- c("Petal.Length", "Petal.Width")
cols <- c(features, "Species")
```

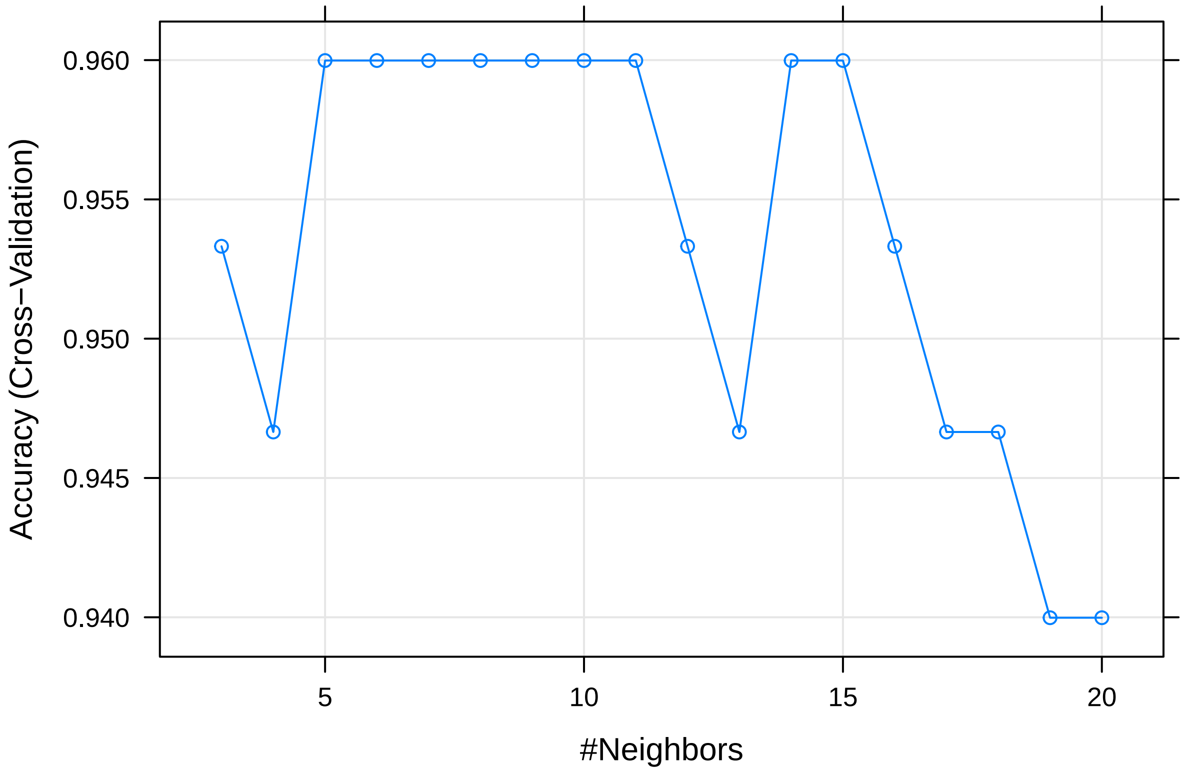
Exercise 8.1 (k-Nearest Neighbour)

We use the library `caret` to implement a learner for k-nearest neighbour that finds the best value for k.

```r
set.seed(50)
ctrl <- trainControl(method="cv", number = 3) # 3-fold cross validation

# tuneGrid contains the values for k which are tried
knnfit <- train(Species ~ ., data = iris[, cols], method = "knn", trControl = ctrl,
                preProcess = c("center","scale"),
                tuneGrid = expand.grid(k=3:20)) # try k=3,4, ... 20

#Output of kNN fit
knnfit
```

```
## k-Nearest Neighbors
##
## 150 samples
##   2 predictor
##   3 classes: 'setosa', 'versicolor', 'virginica'
##
## Pre-processing: centered (2), scaled (2)
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 100, 99, 101
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    3  0.9533173  0.9299844
##    4  0.9466507  0.9199503
##    5  0.9599840  0.9399824
##    6  0.9599840  0.9399824
##    7  0.9599840  0.9399824
##    8  0.9599840  0.9399824
##    9  0.9599840  0.9399824
##   10  0.9599840  0.9399824
##   11  0.9599840  0.9399824
##   12  0.9533173  0.9299724
##   13  0.9466507  0.9199503
##   14  0.9599840  0.9399824
##   15  0.9599840  0.9399824
##   16  0.9533173  0.9299724
##   17  0.9466507  0.9199503
##   18  0.9466507  0.9199503
##   19  0.9399840  0.9099162
##   20  0.9399840  0.9099162
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 15.
```

```r
plot(knnfit)
```

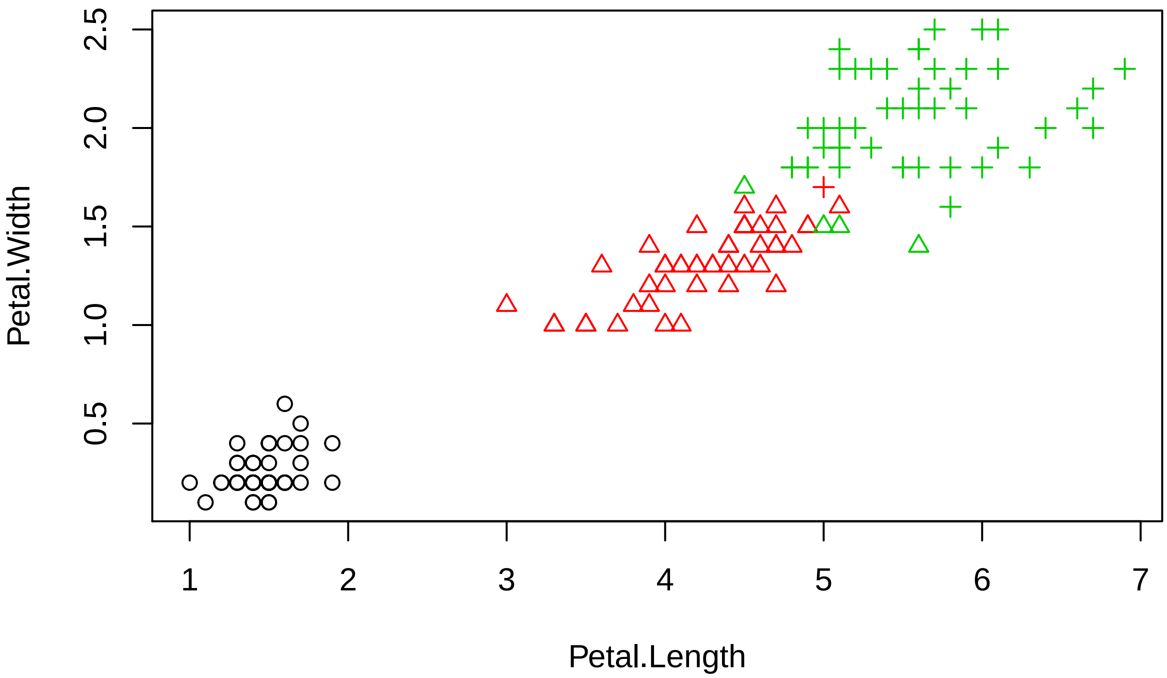



```r
# predict the classes in the entire data set using the best model
knnPredict <- predict(knnfit, newdata = iris[, cols] )

# Get the confusion matrix to see accuracy value and other quality measures
confusionMatrix(knnPredict, iris$Species)
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         4
##   virginica       0          2        46
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.915, 0.9852)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.94
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                          Class: setosa Class: versicolor Class: virginica
## Sensitivity                    1.0000            0.9600           0.9200
## Specificity                    1.0000            0.9600           0.9800
## Pos Pred Value                 1.0000            0.9231           0.9583
## Neg Pred Value                 1.0000            0.9796           0.9608
## Prevalence                     0.3333            0.3333           0.3333
## Detection Rate                 0.3333            0.3200           0.3067
## Detection Prevalence           0.3333            0.3467           0.3200
## Balanced Accuracy              1.0000            0.9600           0.9500
```

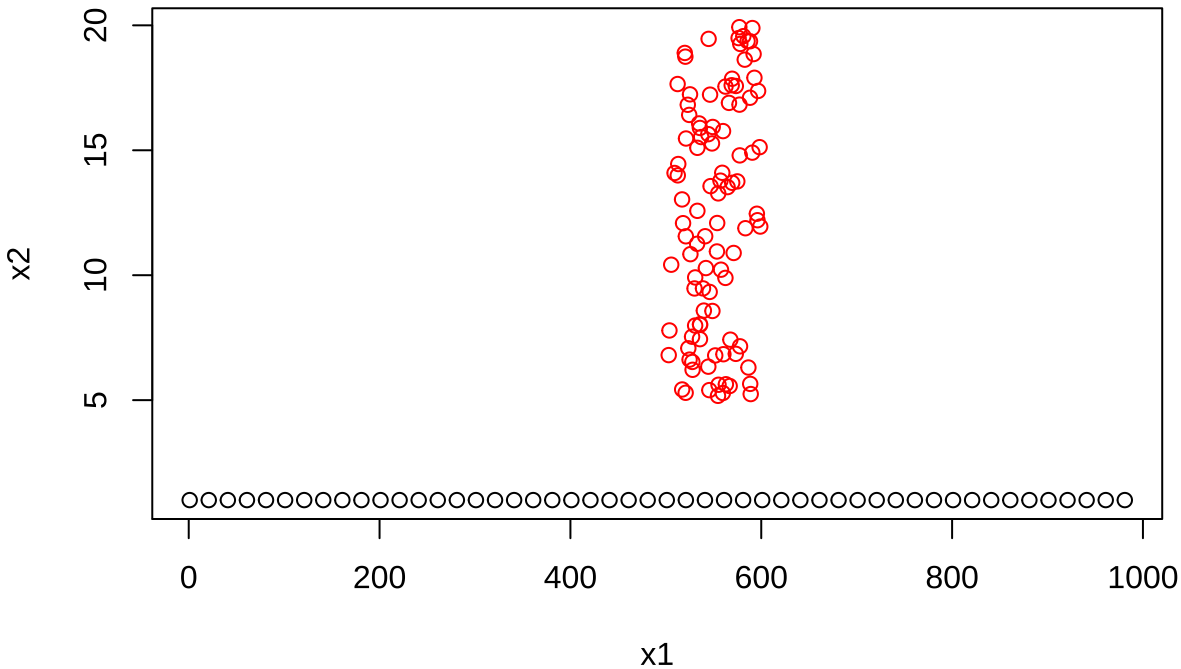
```r
# use colour for original class and symbol for predicted class
plot(iris[,features], col=iris$Species, pch=as.numeric(unclass(knnPredict)))
```



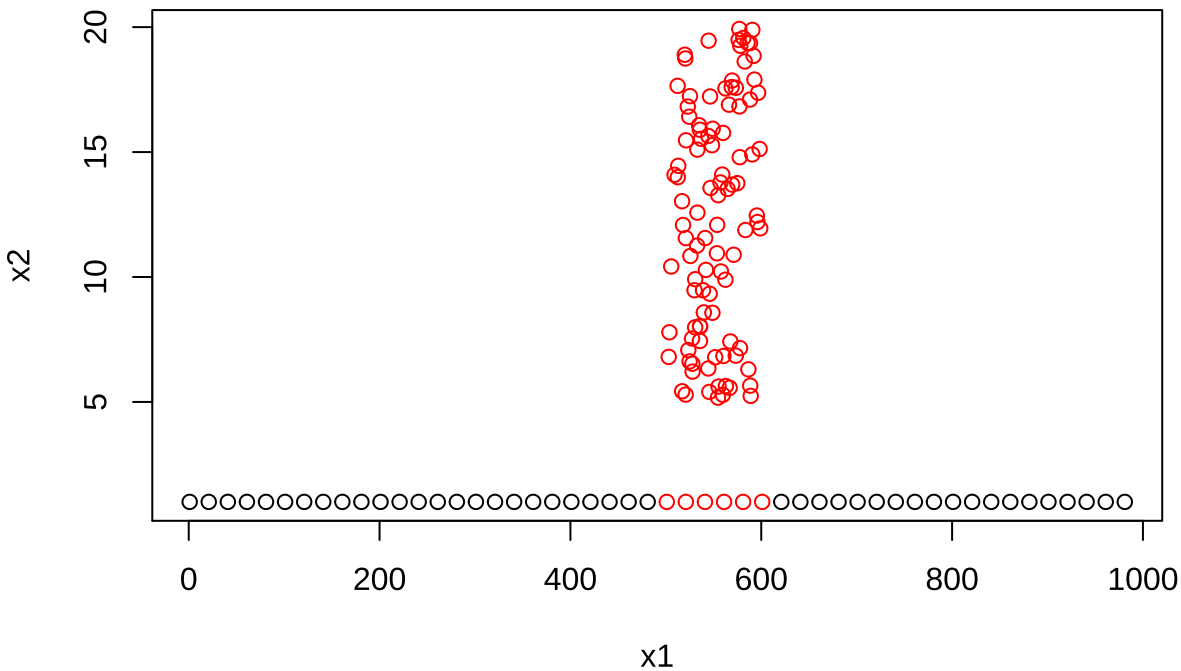
Importance of Scaling

Since the nearest neighbours are found using Euclidean distance (in this case), it is important that the distances in the different features are comparable, i.e. on a similar scale.

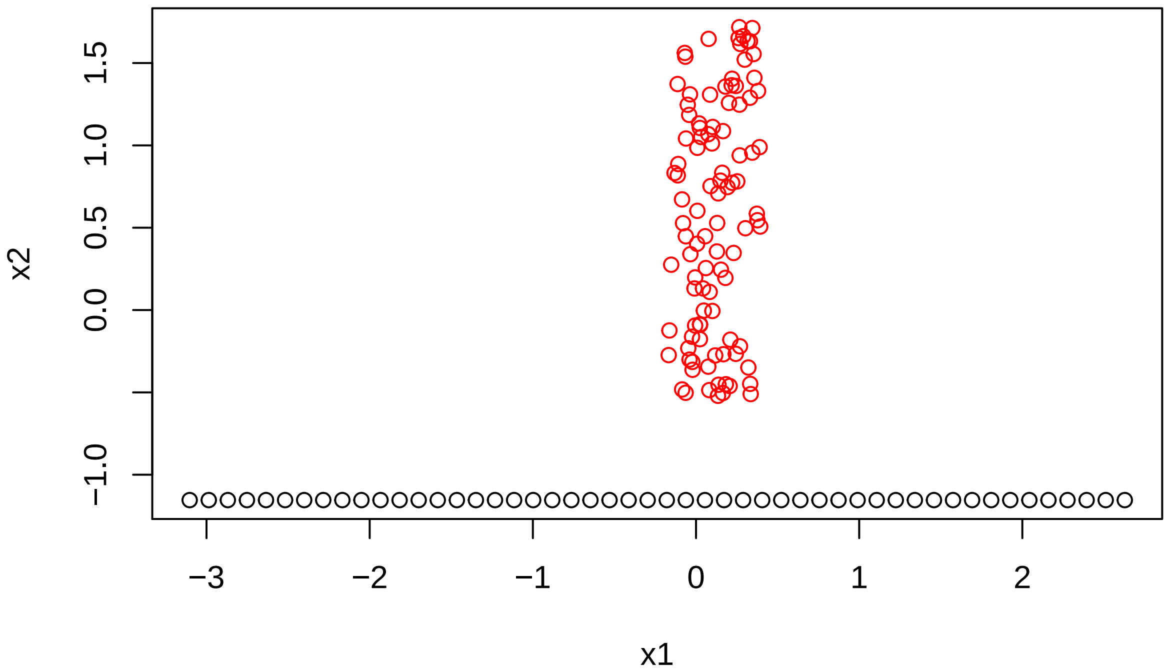
```r
# generate some data

set.seed(100)
x1 <- c(seq(1,1000,20), runif(100, min = 500, max = 600))
x2 <- c(rep(1,50), runif(100, min = 5, max = 20))
myclass <- factor(c(rep(1, 50), rep(2, 100)))
plot(x1, x2, col=myclass)
```




```r
x <- cbind(x1, x2)
pre_x <- preProcess(x, method=c("center", "scale"))
x_n <- predict(pre_x, x)

res <- knn3Train(x, x, myclass, k=5, prob = FALSE)

plot(x1, x2, col=res)
```




```r
res_n <- knn3Train(x_n, x_n, myclass, k=5, prob = FALSE)

plot(x_n[,1] , x_n[,2], col=res_n, xlab = "x1", ylab = "x2")
```



Exercise 8.2 (Support Vector Machine)

Split the data into training and test data set, learn a linear SVM and check the results in terms of accuracy and the confusion matrix. The cost parameter is related to the margin of the classifier. Each data point in the margin produces costs. The higher the cost the smaller the margin, i.e. the fewer data points lie in the margin.

```r
# fit an SVM with linear kernel
svmfit <- svm(Species ~ ., data = iris_train[,cols], kernel = "linear",
              cost = 1, scale = TRUE)

# print the results
print(svmfit)
```

```
##
## Call:
## svm(formula = Species ~ ., data = iris_train[, cols], kernel = "linear",
##     cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  22
```

```r
# more information about the model including the support vectors
summary(svmfit)
```

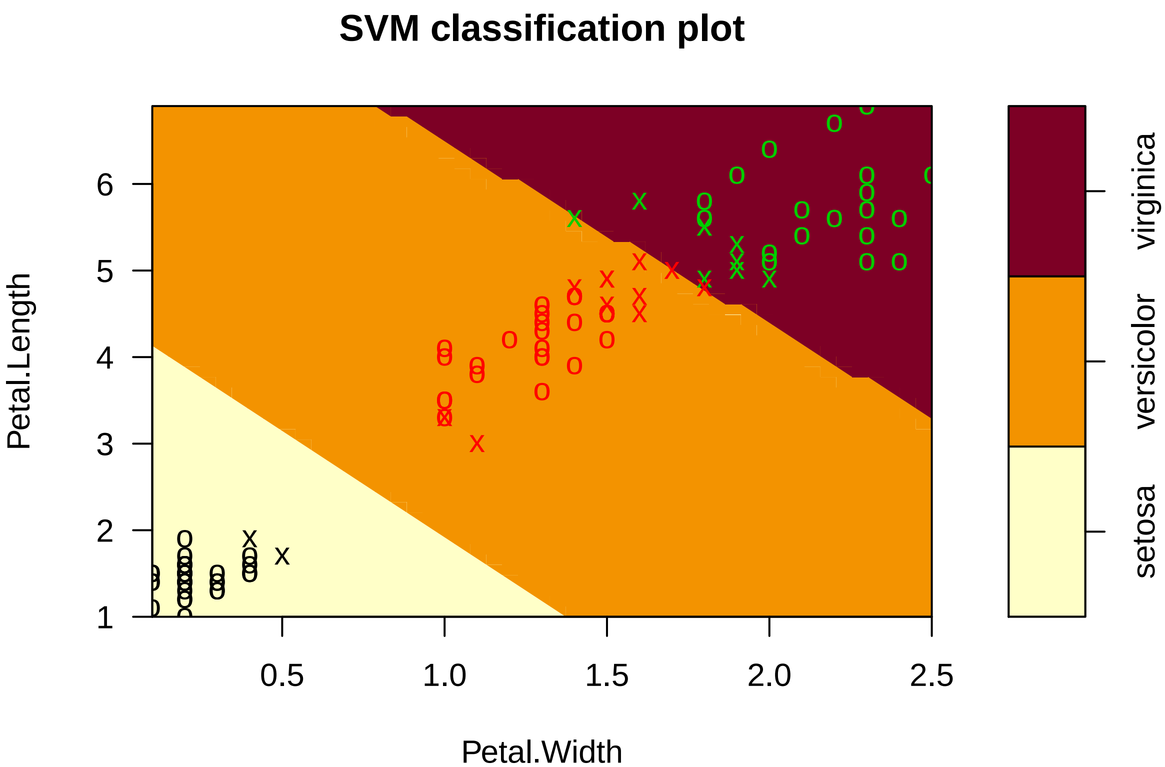
```
##
## Call:
```

```
## svm(formula = Species ~ ., data = iris_train[, cols], kernel = "linear",
##     cost = 1, scale = TRUE)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  22
##
##  ( 9 2 11 )
##
##
## Number of Classes:  3
##
## Levels:
##  setosa versicolor virginica
```

```
# indices of the support vectors
svmfit$index
```

```
##  [1] 19 23 28 37 43 56 72 77 83 14 48 16 35 52 61 67 75 80 81 91 96 97
```

```
# plot it including the separation lines (hyperplanes)
plot(svmfit, iris_train[,cols])
```




```
# predict on the test data set
prediction <- predict(svmfit, iris_test, type = "class")
```

```r
# confusion matrix with quality measures
confusionMatrix(prediction, iris_test[,"Species"])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         15          0         0
##   versicolor      0         13         3
##   virginica       0          0        19
##
## Overall Statistics
##
##                Accuracy : 0.94
##                  95% CI : (0.8345, 0.9875)
##     No Information Rate : 0.44
##     P-Value [Acc > NIR] : 6.318e-14
##
##                   Kappa : 0.909
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                    1.0            1.0000           0.8636
## Specificity                    1.0            0.9189           1.0000
## Pos Pred Value                 1.0            0.8125           1.0000
## Neg Pred Value                 1.0            1.0000           0.9032
## Prevalence                     0.3            0.2600           0.4400
## Detection Rate                 0.3            0.2600           0.3800
## Detection Prevalence           0.3            0.3200           0.3800
## Balanced Accuracy              1.0            0.9595           0.9318
```

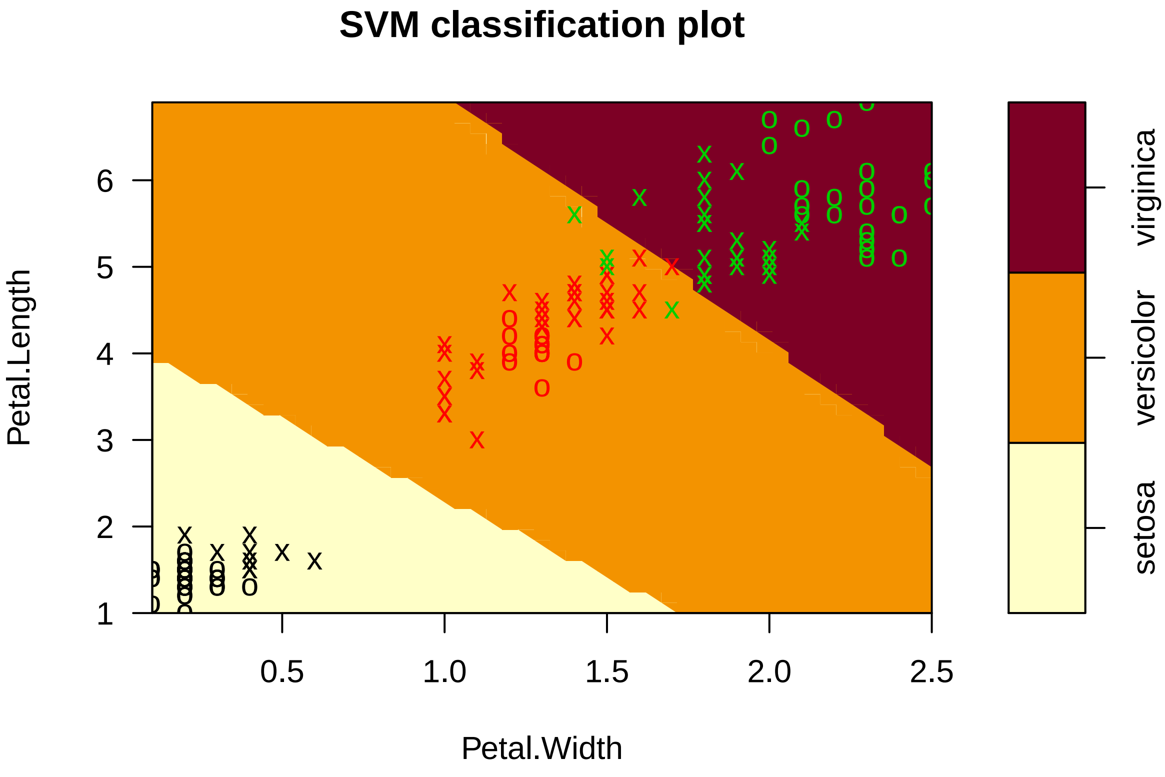
Exercise 8.3 (Support Vector Machine)

We would like to find the best value for the cost parameter using cross validation. We can use the function `tune()` in the library `e1071`. Alternatively, the package `caret` can be used as with kNN above, but we then may have to choose a different implementation of an SVM supported by `caret`.

```r
# function to find the best value for the cost parameter
# assume a column called Species in my_data representing the class to be predicted
run_iris_svm <- function(my_data, svm_type) {

    # automatically run five-fold cross validation to find the best value for cost
    # all the code related to the parameter gamma should be ignored by the functions
    # if svm_type = "linear", since a linear kernel only requires cost
    tuned_svm <- tune(svm, Species ~ ., data = my_data, kernel = svm_type,
              scale = TRUE, ranges = list(cost = c(0.01, 0.1, 1, 10, 100),
                                          gamma = c(0.25, 0.5, 1, 2)),
              tunecontrol = tune.control(cross=5))
    print(tuned_svm)
```

```
    # train on the entire data set using best value for cost
    best_cost <- tuned_svm$best.parameters[1]
    best_gamma <- tuned_svm$best.parameters[2]


    svmfit_best <- svm(Species ~ ., data = my_data, kernel = svm_type,
                  cost = best_cost, gamma = best_gamma, scale = TRUE)

    # plot the result
    plot(svmfit_best, my_data)

    # confusion matrix with quality measures
    print(confusionMatrix(svmfit_best$fitted, my_data[,"Species"]))

}

# learn a linear SVM
run_iris_svm(iris[,cols], "linear")
```

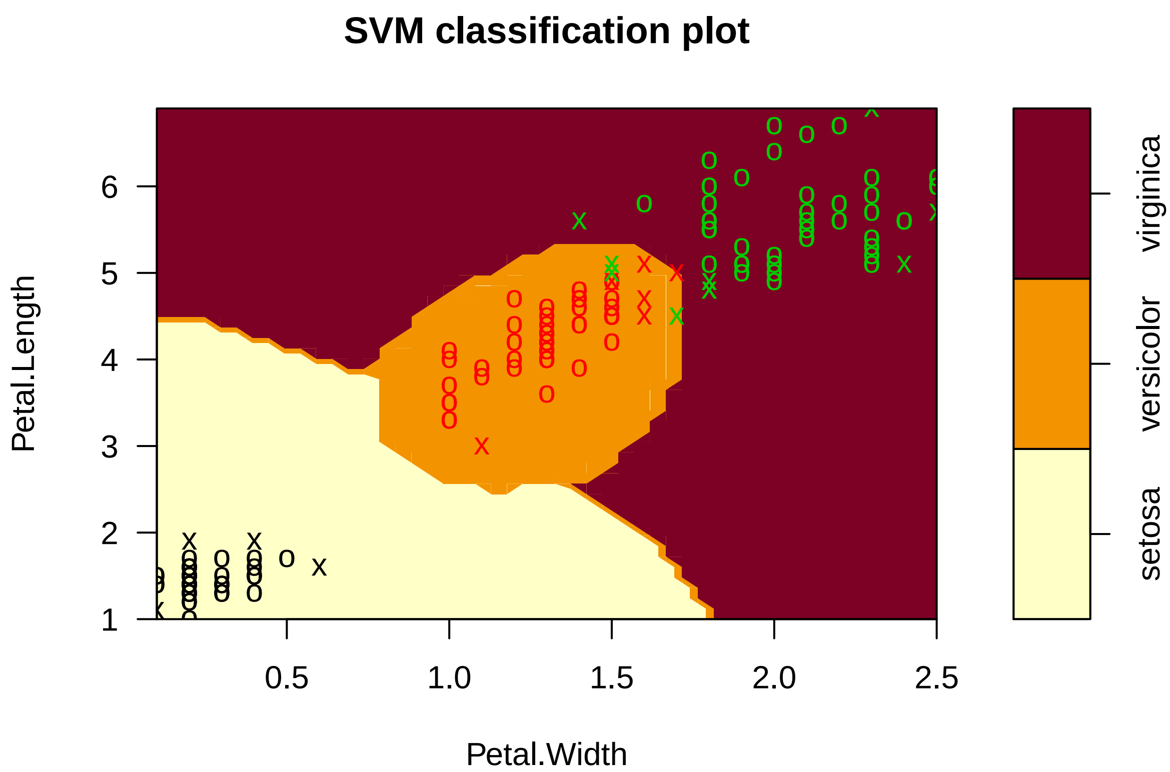
```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost gamma
##   0.1  0.25
##
## - best performance: 0.04
```



```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         4
##   virginica       0          2        46
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.915, 0.9852)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.94
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9600           0.9200
## Specificity                 1.0000            0.9600           0.9800
## Pos Pred Value              1.0000            0.9231           0.9583
## Neg Pred Value              1.0000            0.9796           0.9608
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3200           0.3067
## Detection Prevalence        0.3333            0.3467           0.3200
## Balanced Accuracy           1.0000            0.9600           0.9500
```

```r
# learn an SVM with a radial basis function kernel
# radial requires a value for the parameter gamma
# which should be optimised as well; we just go for the default here
run_iris_svm(iris[,cols], "radial")
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost gamma
##   100   0.5
##
## - best performance: 0.03333333
```

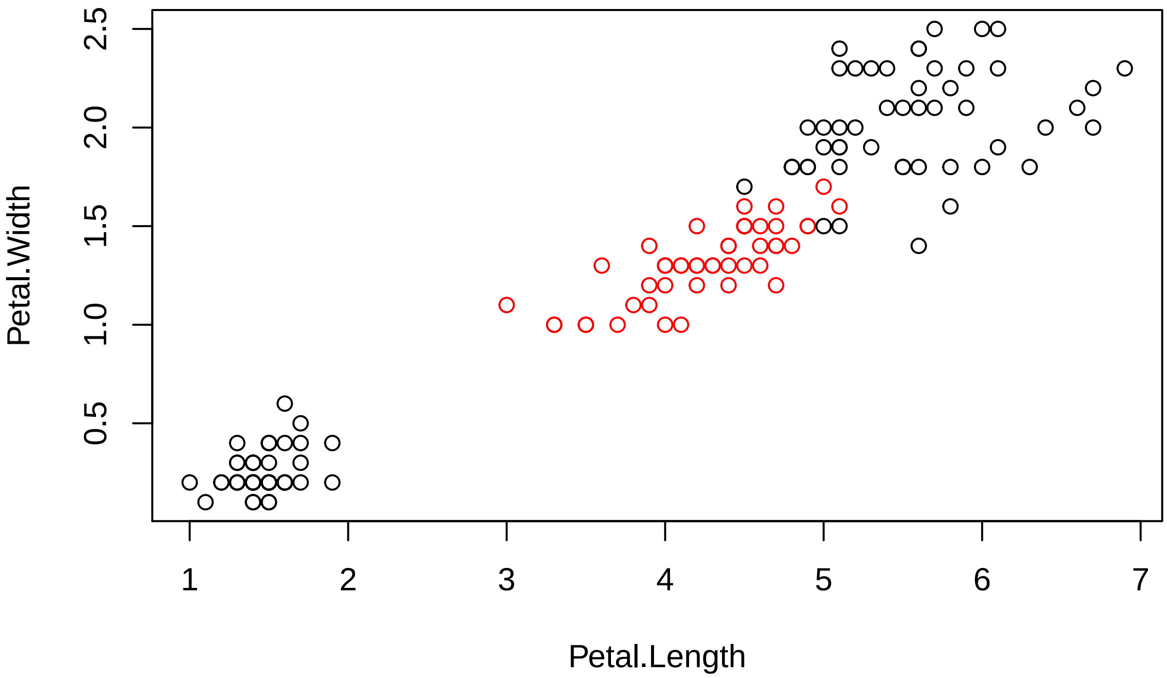



```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         48         3
##   virginica       0          2        47
##
## Overall Statistics
##
##                Accuracy : 0.9667
##                  95% CI : (0.9239, 0.9891)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.95
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9600           0.9400
## Specificity                 1.0000            0.9700           0.9800
## Pos Pred Value              1.0000            0.9412           0.9592
## Neg Pred Value              1.0000            0.9798           0.9703
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3200           0.3133
## Detection Prevalence        0.3333            0.3400           0.3267
## Balanced Accuracy           1.0000            0.9650           0.9600
```

Exercise 8.4

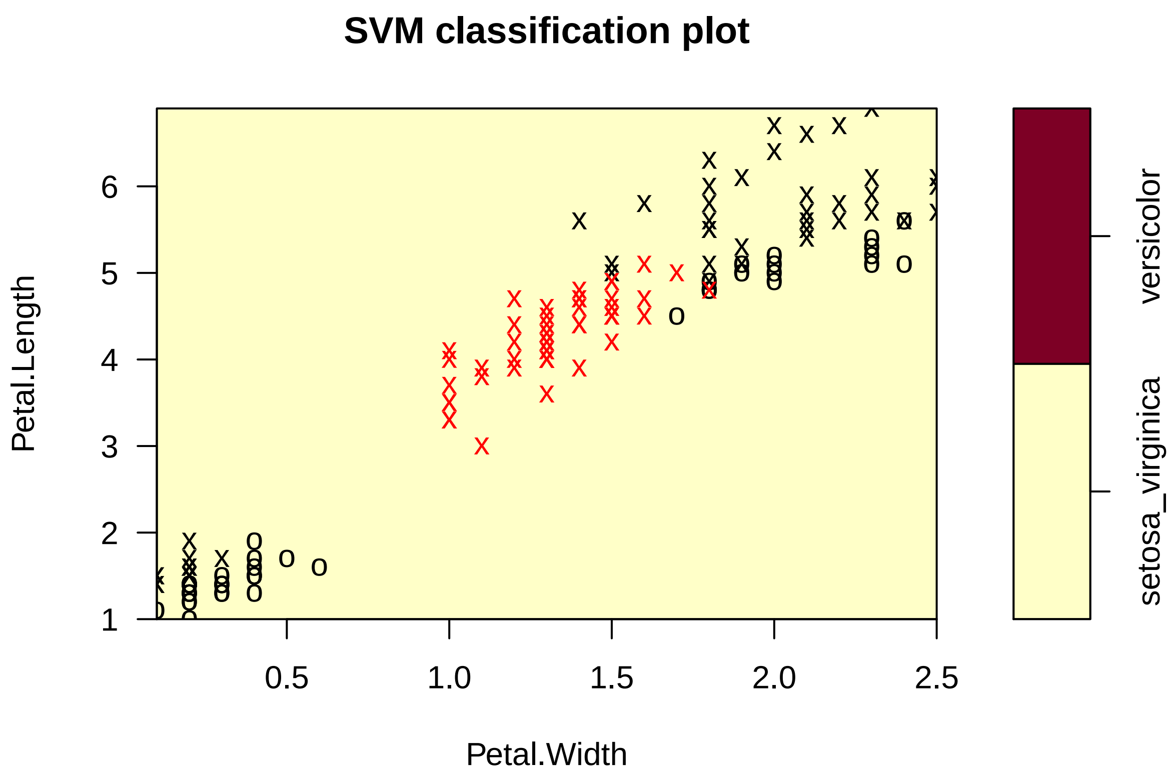
Join the classes *setosa* and *virginica* in a new class *setosa_virginica* and learn a classifier to separate it from *versicolor*. This cannot be achieved by linear separation.

```
iris2 <- iris
iris2$Species <- as.factor(ifelse(iris2$Species=="versicolor", "versicolor",
                                  "setosa_virginica"))

plot(iris2[,features], col=iris2$Species)
```




```
# linear separation must fail
run_iris_svm(iris2[,cols], "linear")
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost gamma
##  0.01  0.25
##
## - best performance: 0.3333333
```

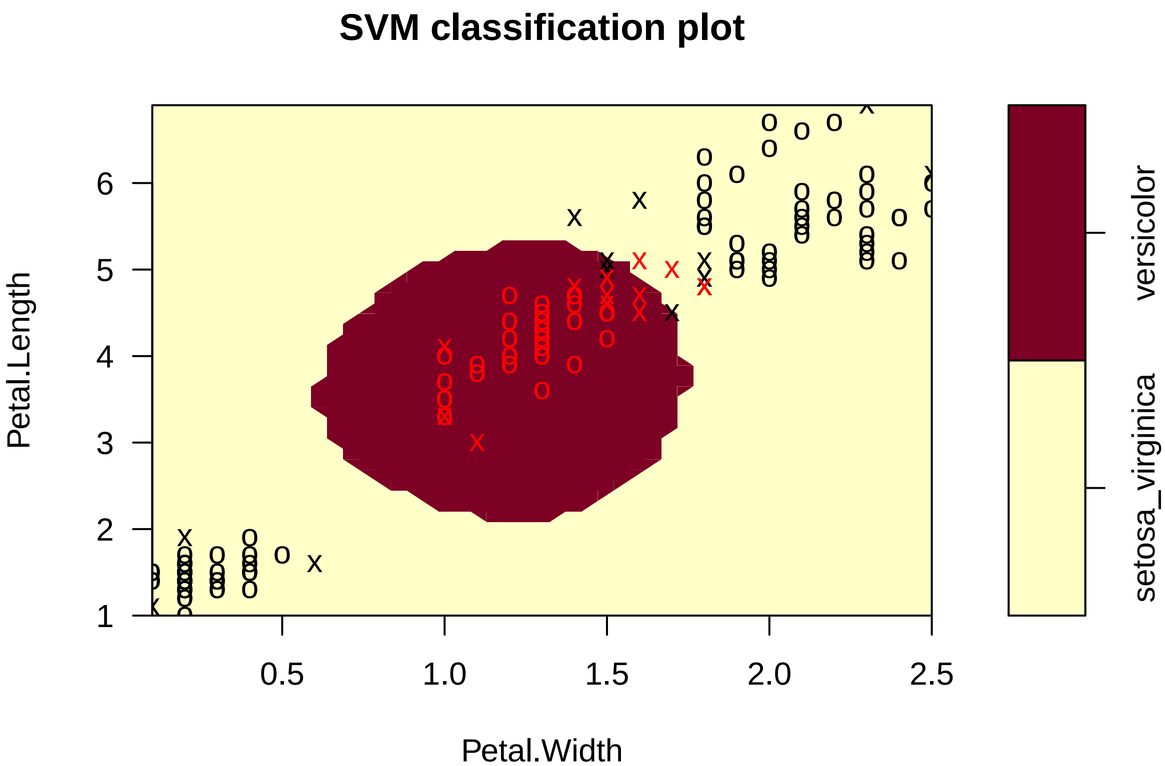



```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction         setosa_virginica versicolor
##   setosa_virginica              100         50
##   versicolor                      0          0
##
##                Accuracy : 0.6667
##                  95% CI : (0.5852, 0.7414)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : 0.5383
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 4.219e-12
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.6667
##          Neg Pred Value :    NaN
##              Prevalence : 0.6667
##          Detection Rate : 0.6667
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : setosa_virginica
##
```

As expected, linear separation fails and all cases are assigned to the same class, i.e. the majority class *setosa_virginica*.

```r
# radial kernel works nicely
run_iris_svm(iris2[,cols], "radial")
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 5-fold cross validation
##
## - best parameters:
##  cost gamma
##     1     2
##
## - best performance: 0.03333333
```




```
## Confusion Matrix and Statistics
##
##                   Reference
## Prediction          setosa_virginica versicolor
##   setosa_virginica                97          3
##   versicolor                       3         47
##
##                Accuracy : 0.96
##                  95% CI : (0.915, 0.9852)
##     No Information Rate : 0.6667
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.91
##
##  Mcnemar's Test P-Value : 1
##
```

```
##             Sensitivity : 0.9700
##             Specificity : 0.9400
##          Pos Pred Value : 0.9700
##          Neg Pred Value : 0.9400
##              Prevalence : 0.6667
##          Detection Rate : 0.6467
##    Detection Prevalence : 0.6667
##       Balanced Accuracy : 0.9550
##
##        'Positive' Class : setosa_virginica
##
```

Exercise 8.5

Re-run Exercises 1 and 2 with two different features like `Sepal.Length` and `Sepal.Width` that do not allow a linear separation. See, how kNN and SVMs perform. Finally, use all four features.