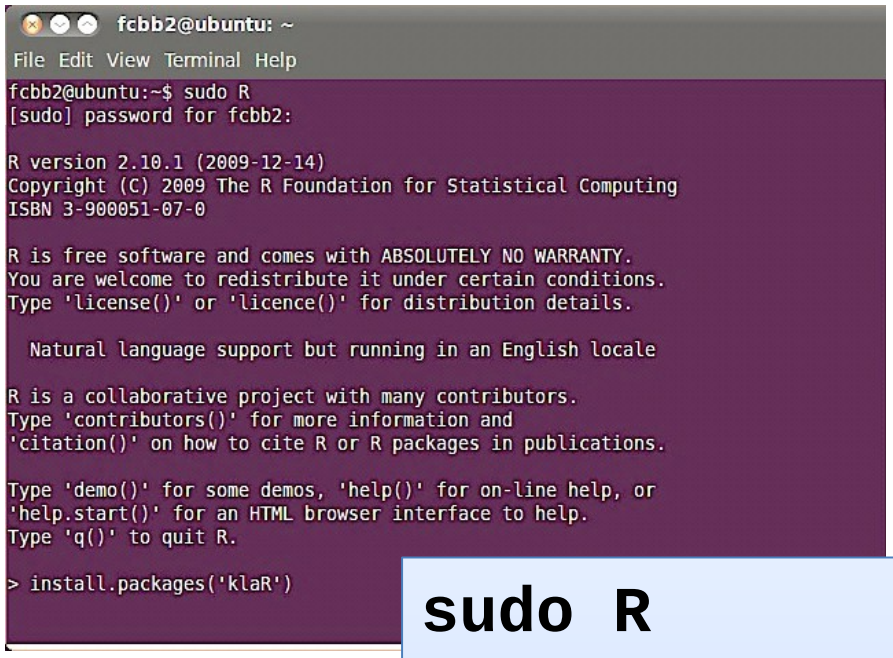
Machine Learning Tools in R

FCBB Practical Session

Overview

- Setting up R machine learning packages
- R code overview
  - Setting up machine learning packages
  - ROC analysis
  - Plotting and saving plots
- The Iris Dataset
  - Decision tree example
  - Naïve Bayes example
  - SVM examples
- ROC curves
- k-fold Cross-validation

Setting up R packages



- "sudo" allows you to run R as root
- install.packages downloads external packages and automatically installs them into R

```
sudo R
install.packages('getopt')
install.packages('klaR')
install.packages('e1071')
install.packages('ROCR')
library(kLaR) #load package
library(ROCR)
```

Machine Learning Commands

```
library(packageName)

modelName<-classifierFunc(labelId~featureIds, trainset)

predictionScores<- predict(modelName, testset)
```

- Naïve Bayes: **NaiveBayes(formula, dataset)** **klaR**
- Decision Trees: **rpart(formula, dataset),** **rpart**
- Support Vector Machines: **svm(formula, dataset),** **e1071**

ROC curves and AUC

```
library(ROCR)
…
pred <- prediction(prediction_scores, true_labels)
perf <- performance(pred, “tpr”, “fpr”)
plot(perf)# plots ROC curve

   perf.auc <- performance(pred,'auc’) #calculates AUC
   auc <- perf.auc@y.values
```

- **`prediction_scores`** should contain the raw scores, i.e. svm_scores, probabilities, log likelihood, …

http://rocr.bioinf.mpi-sb.mpg.de/

Saving R plots

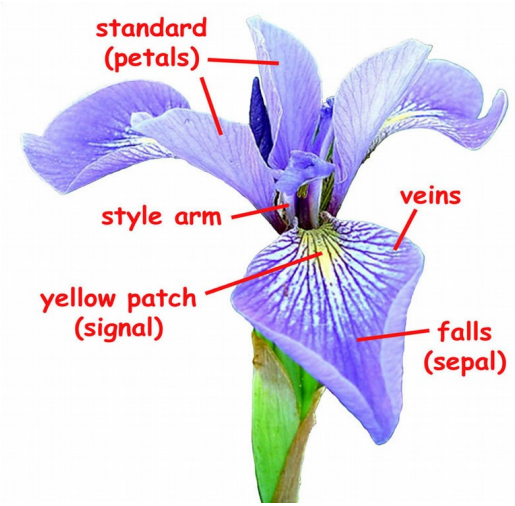
- Different formats:
  - **`jpeg(), png(), pdf(), postscript()`**
- Open, plot, save, then close:

```
pdf('myplot.pdf')
plot(x,y)
dev.off()
```

http://www.stat.berkeley.edu/users/spector/s133/saving.html

The iris dataset


```
> data(iris)
> 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
```

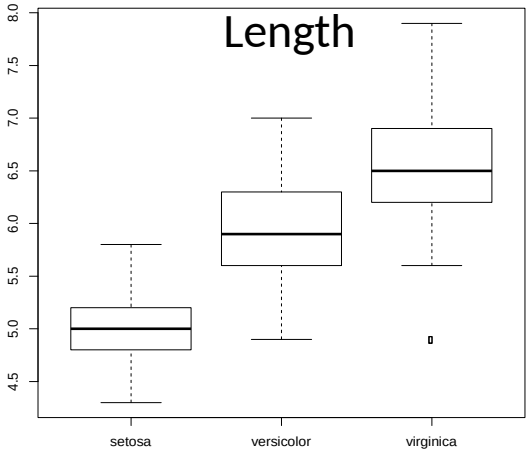


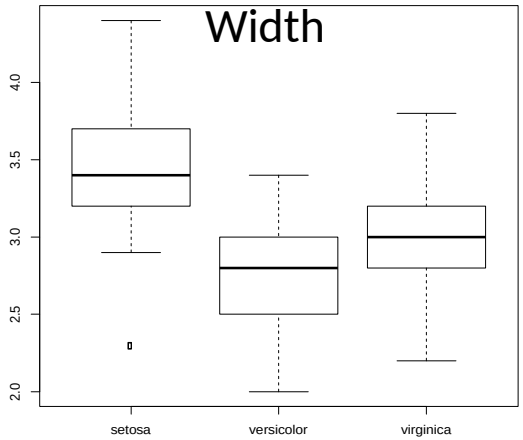

```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300000   Min.   :2.000000   Min.   :1.000   Min.   :0.100000
 1st Qu.:5.100000   1st Qu.:2.800000   1st Qu.:1.600   1st Qu.:0.300000
 Median :5.800000   Median :3.000000   Median :4.350   Median :1.300000
 Mean   :5.843333   Mean   :3.057333   Mean   :3.758   Mean   :1.199333
 3rd Qu.:6.400000   3rd Qu.:3.300000   3rd Qu.:5.100   3rd Qu.:1.800000
 Max.   :7.900000   Max.   :4.400000   Max.   :6.900   Max.   :2.500000
    Species
 setosa    :50
 versicolor:50
 virginica :50
```

The iris dataset

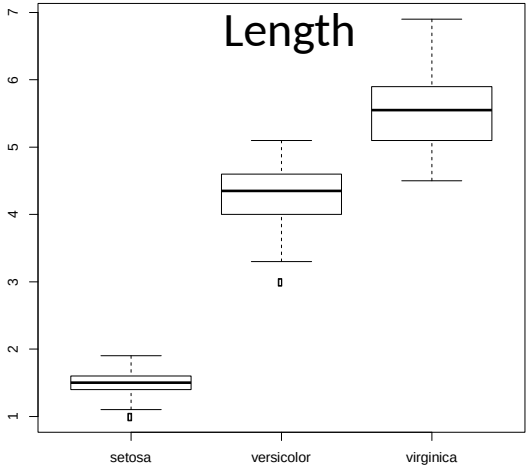

```
> boxplot(Sepal.Length~Species, data=iris)
```

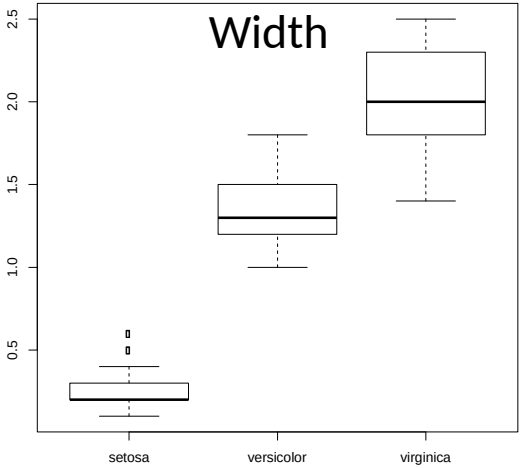
Sepal

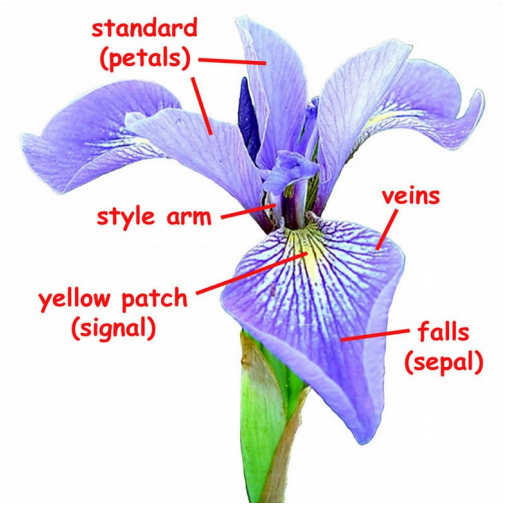




Petal

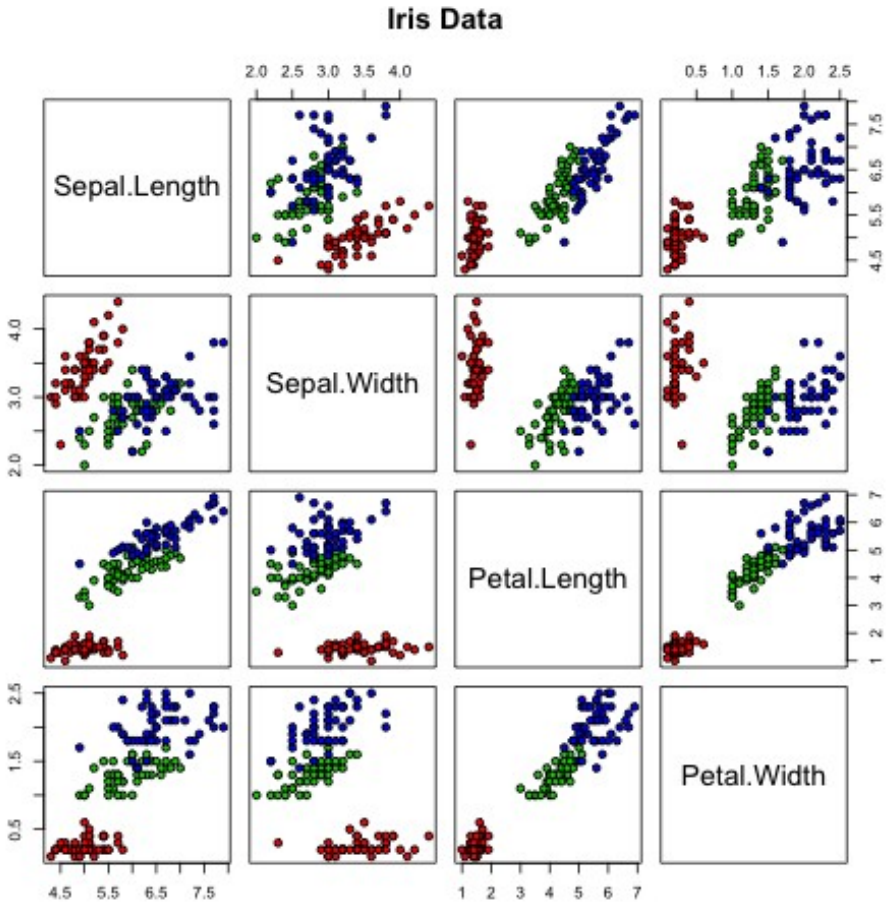






The iris dataset


```
> pairs(iris[1:4], main = “Iris Data", pch = 21, bg = c("red", "green3",
"blue")[unclass(iris$Species)])
```

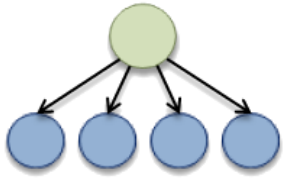


setosa
versicolor
virginica

Naïve Bayes

$$P(Y \mid X) \propto P(X \mid Y)\, P(Y)$$ Target function

$$P(X \mid Y) = \prod_{i=1}^{k} P(X_i \mid Y)$$ Naïve Bayes Assumption



Naïve Bayes

```
> testidx <- which(1:length(iris[,1])%%5 == 0)
 [1]   5  10  15  20  25  30  35  40  45  50  55  60  65  70  75  80  85  90  95
[20] 100 105 110 115 120 125 130 135 140 145 150
> iristrain <- iris[-testidx,] # 120 samples
> iristest <- iris[testidx,] # 30 samples
```

```
> library(klaR)
> nbmodel <- NaiveBayes(Species~., data=iristrain)
> prediction <- predict(nbmodel, iristest[,-5])
> table(prediction$class, iristest[,5])

prediction    setosa versicolor virginica
  setosa          10          0         0
  versicolor       0         10         2
  virginica        0          0         8
```

```
> nbmodel <- NaiveBayes(Species~., data=iristrain)
> nbmodel

Naive Bayes Classifier for Discrete Predictors

Call:
naiveBayes.default(x = X, y = Y, laplace = laplace)

A-priori probabilities:
Y
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Conditional probabilities:
            Sepal.Length
Y                [,1]      [,2]
  setosa     4.9975 0.3675892
  versicolor 5.9900 0.5295378
  virginica  6.6100 0.6647922

            Sepal.Width
Y                [,1]      [,2]
  setosa     3.4175 0.3960623
  versicolor 2.7775 0.3415556
  virginica  2.9700 0.3081791
….
```

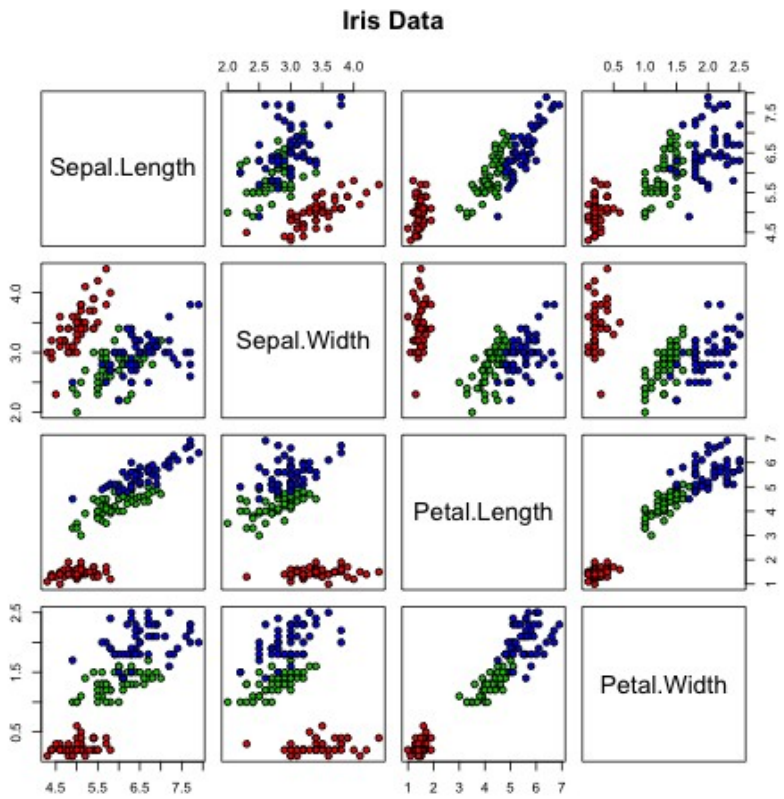
$$P(Y \mid X) \propto P(X \mid Y)\, P(Y)$$

$$P(X \mid Y) = \prod_{i=1}^{k} P(X_i \mid Y)$$

Naïve Bayes

```
> table(prediction, iristest[,5])
prediction    setosa versicolor virginica
  setosa          10          0         0
  versicolor       0         10         2
  virginica        0          0         8
```

setosa
versicolor
virginica




```
> pairs(iris[1:4], main = “Iris
Data", pch = 21, bg = c("red",
"green3", "blue")
[unclass(iris$Species)])
```

Decision Tree (CART)

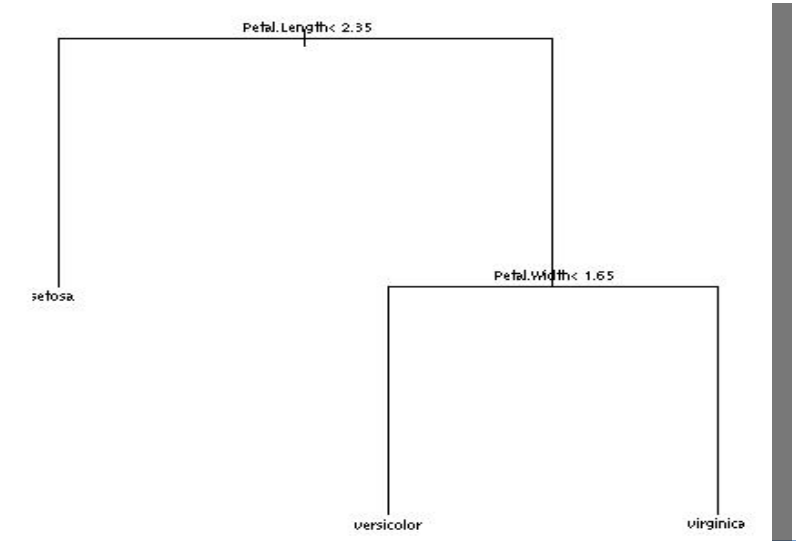
	Not SPAM	Not SPAM	SPAM	SPAM	SPAM
money	0.00	0.26	0.23	0.63	0.99
parts	0	0	0	0	0
address	0.10	0.00	0.05	0.63	0.99
charDollar	0.022	0.000	0.011	0.496	0.596
charSemicolon	0.022	0.00	0.00	0.00	0.00
	$\vec{X1}$	$\vec{X2}$	$\vec{X3}$	$\vec{X4}$	$\vec{X5}$

CART tree (Breiman 1983)

Split on the feature and threshold that minimizes class impurity

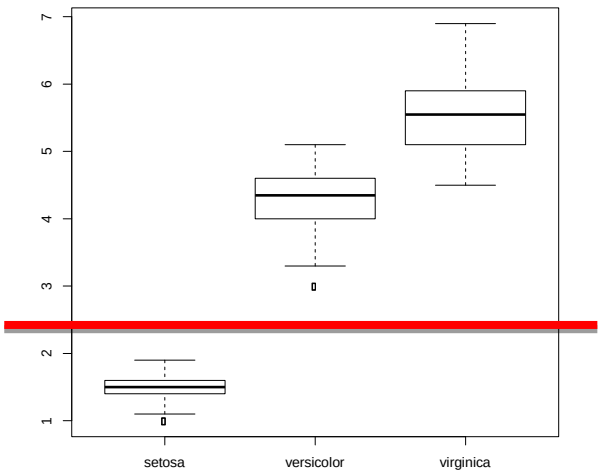
http://cran.r-project.org/web/packages/rpart/rpart.pdf

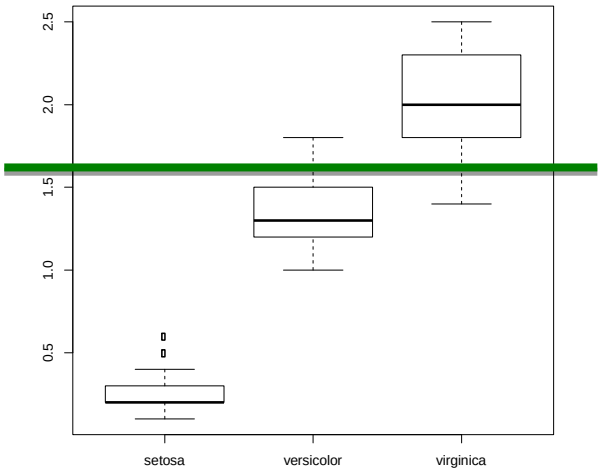
Decision Tree

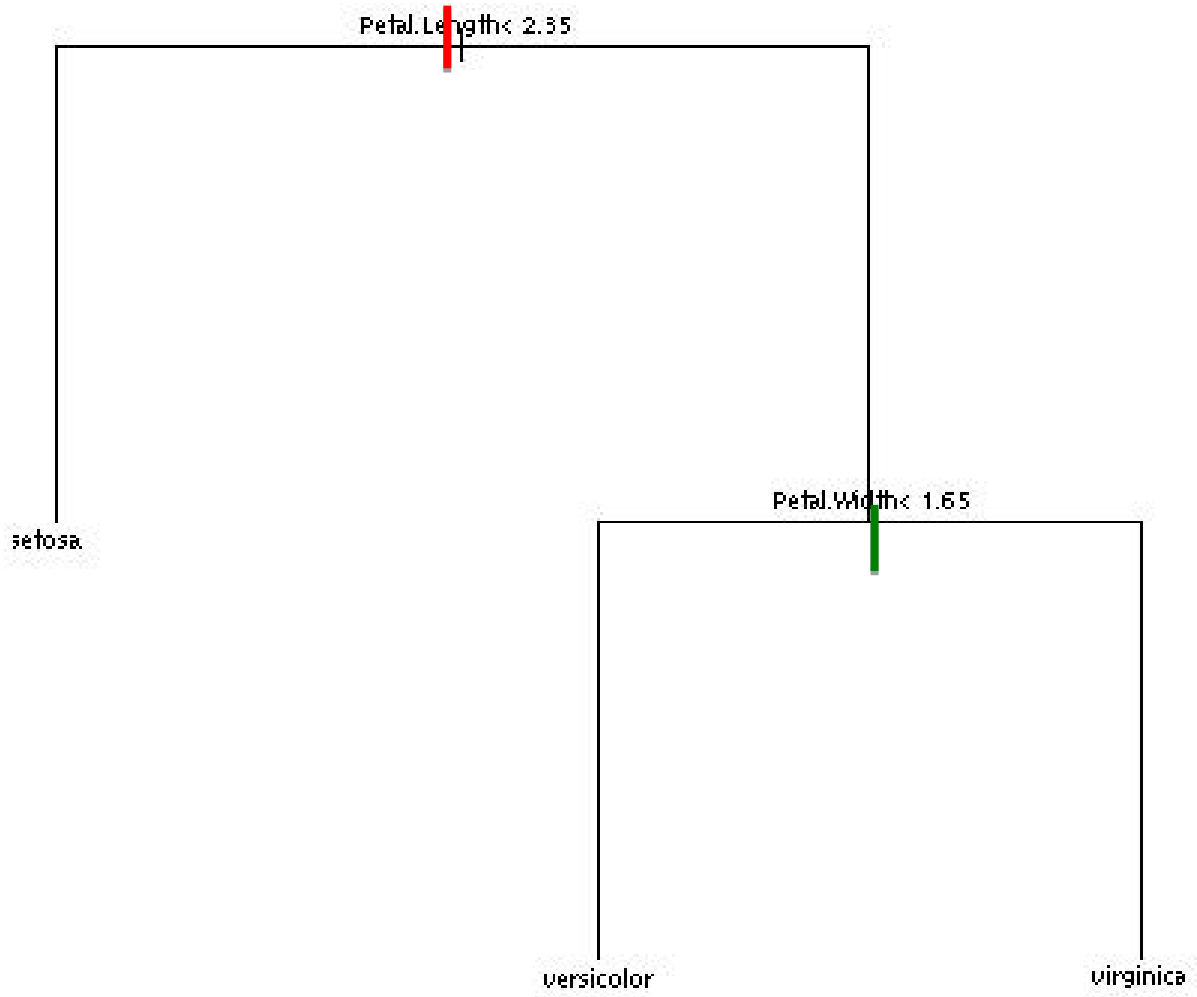



```
> library(rpart)
> treemodel <- rpart(Species~., data=iristrain)
> plot(treemodel)
> text(treemodel, use.n=T)

> prediction <- predict(treemodel, newdata=iristest,
type='class’)
> table(prediction, iristest$Species)

prediction   setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         3
  virginica       0          0         7
```

Decision Tree







Support Vector Machines (SVM)

```
> library(e1071)

> model <- svm(Species~., data=iristrain)

> prediction <- predict(model, iristest)

> table(iristest$Species, prediction)
```

```
> summary(model)

Call:
svm(formula = Species ~ ., data = iristrain)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.25

Number of Support Vectors:  47

 ( 8 21 18 )


Number of Classes:  3

color virginica
```

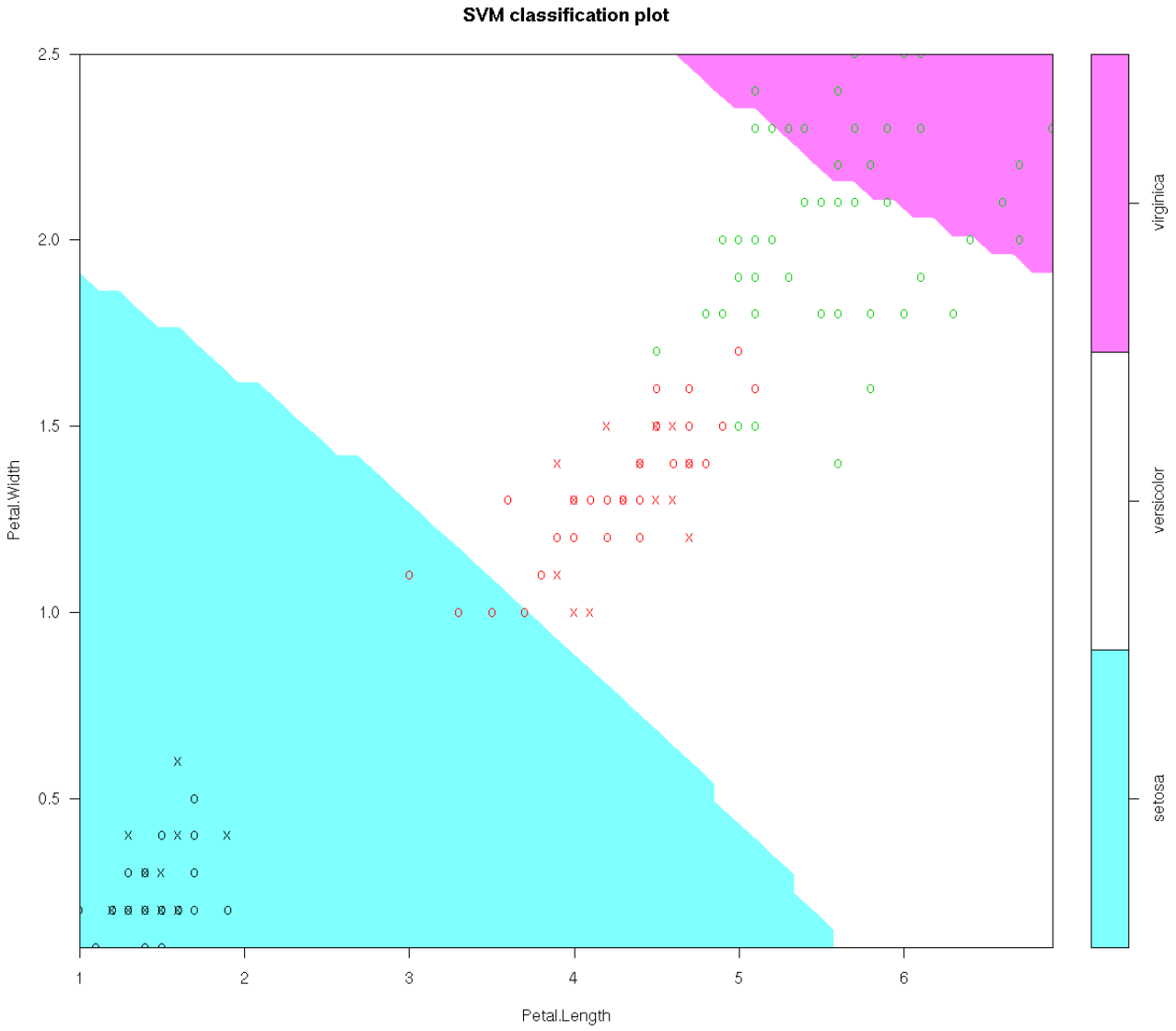
```
# kernels for svm method in e1071 package
linear: u'*v
polynomial: (gamma*u'*v + coef0)^degree
radial basis: exp(-gamma*|u-v|^2)  # default
sigmoid: tanh(gamma*u'*v + coef0)
```

Support Vector Machines (SVM)

```
> library(e1071)
> model <- svm(Species~., data=iristrain)
> prediction <- predict(model, iristest)
> table(iristest$Species, prediction)
```

```
> table(iristest$Species, prediction)
            prediction
             setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         0
  virginica       0          1         9
```

```
> plot(model, iris, Petal.Width ~ Petal.Length,
+      slice = list(Sepal.Width = 3, Sepal.Length = 4))
```



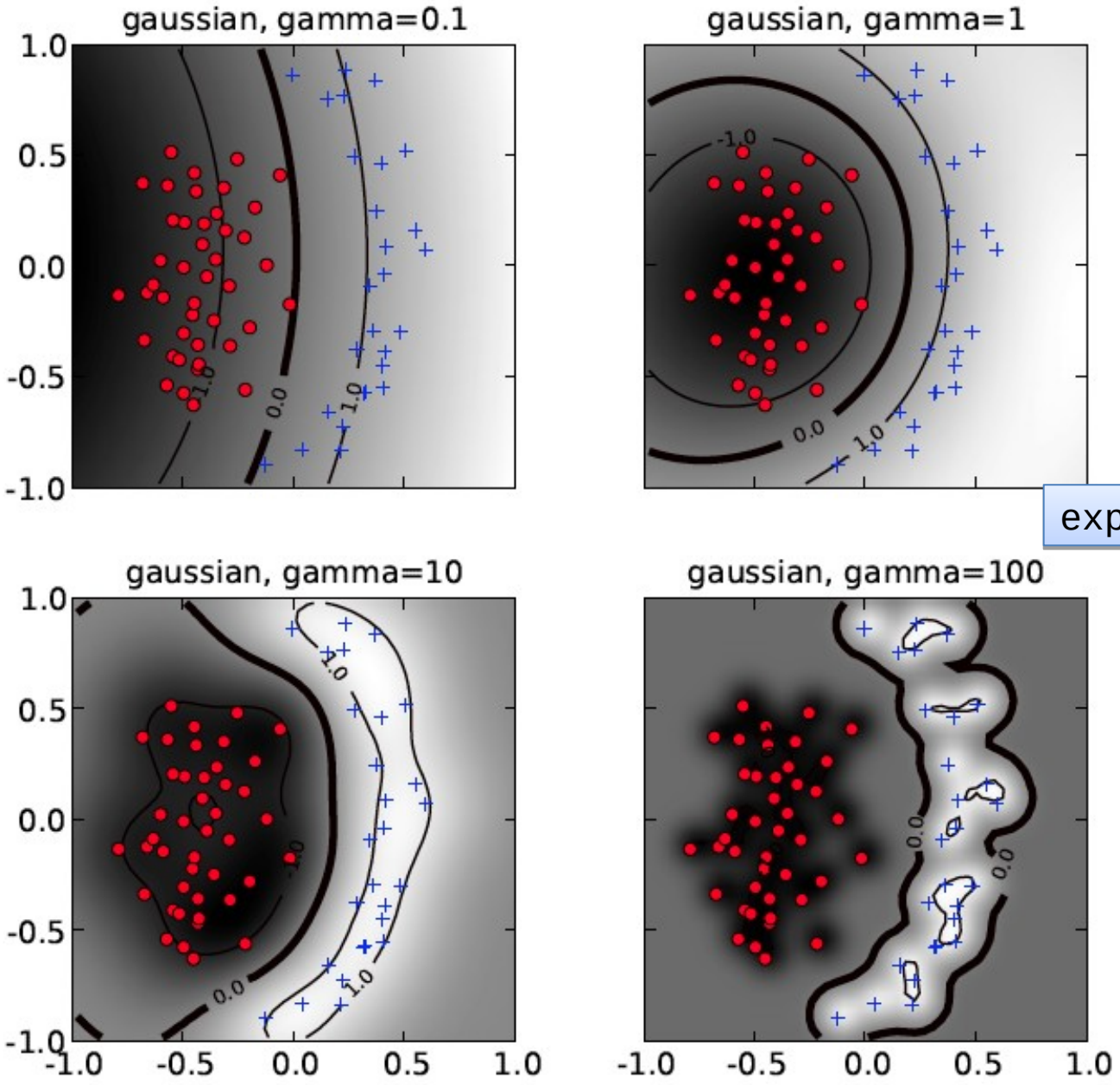
?plot.svm

SVM Tuning

```
> tune <- tune.svm(Species~., data=iristrain,
gamma=10^(-5:0), cost=10^(0:5))
```

```
> summary(tune)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:
 gamma  cost
 0.001 10000

- best performance: 0.025

- Detailed performance results:
   gamma  cost      error dispersion
1  1e-05 1e+00 0.71666667 0.22291558
2  1e-04 1e+00 0.71666667 0.22291558
3  1e-03 1e+00 0.61666667 0.21588177
4  1e-02 1e+00 0.13333333 0.08050765
5  1e-01 1e+00 0.03333333 0.04303315
```

exp(-gamma*|u-v|^2)



`exp(-gamma*|u-v|^2)`

http://pyml.sourceforge.net/doc/howto.pdf

SVM Tuning

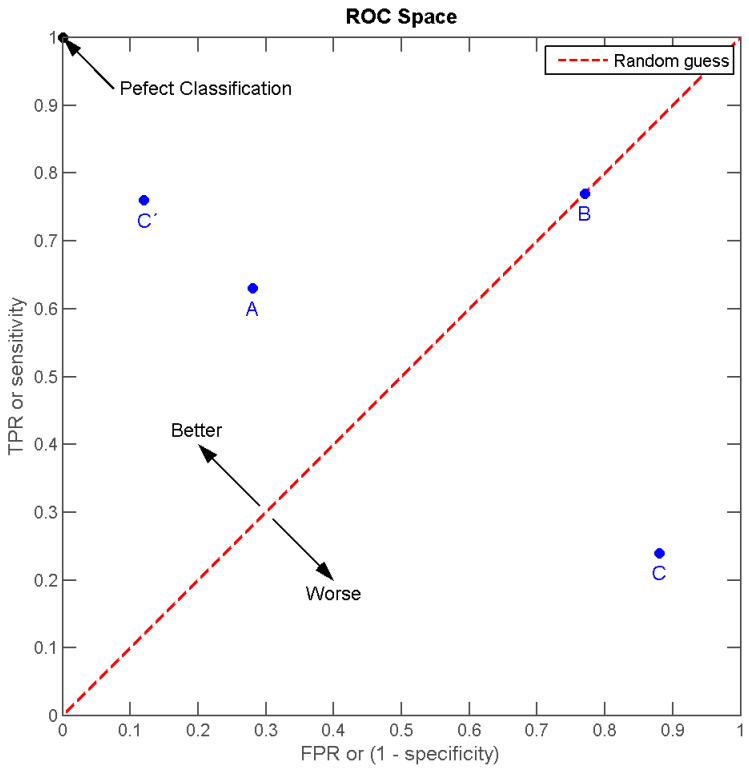
```
> model <- svm(Species~., data=iristrain,  probability=T,
gamma=0.001, cost=10000)

> prediction <- predict(model, iristest)

> table(iristest$Species, prediction)

 prediction
             setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         0
  virginica       0          3         7
```

No better performance than the example without tuning! What happened?
Overfiting? Underfitting?
Overall, this tuned SVM may still perform better than the untuned version.

Receiver Operating Characteristic

- Originated from signal detection theory
- Illustrates the performance of a binary classifier as discrimination threshold is varied

		actual value		
		p	***n***	**total**
p'		True Positive	False Positive	P'
prediction outcome				
n'		False Negative	True Negative	N'
total		P	N	



http://www.unc.edu/courses/2010fall/ecol/563/001/docs/lectures/lecture22.htm

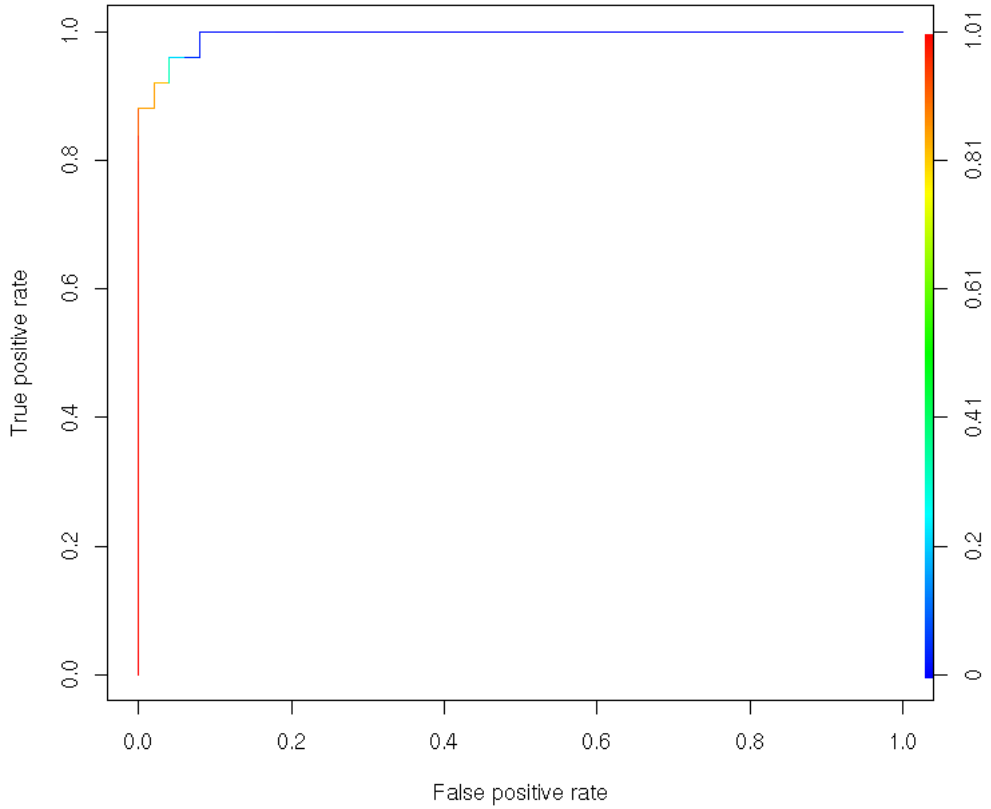
ROC curve for Naïve Bayes

```
> library(klaR)
> nbmodel <- NaiveBayes(Species~., data=iristrain)
> prediction <- predict(nbmodel, iristest[,-5])
> table(prediction$class, iristest[,5])

prediction    setosa versicolor virginica
  setosa          10          0         0
  versicolor       0         10         2
  virginica        0          0         8
```

ROC curve for Naïve Bayes

```
> library(ROCR)
> nbmodel <- NaiveBayes(Species~., data=iristrain)
> nbprediction <- predict(nbmodel, iristest[,-5], type= ‘raw’)
> score <- nbprediction$posterior[, c("virginica")]
> actual_class <- iristest$Species == 'virginica'
> pred <- prediction(score, actual_class)
> nbperf <- performance(pred, "tpr", "fpr")
> nbauc <- performance(pred, "auc")
> nbauc <- unlist(slot(nbauc, "y.values"))
> plot(nbperf, colorize=TRUE)
> legend(0.6,0.3,c(c(paste('AUC is', nbauc)),"\n”),
border="white",cex=1.0, box.col = "white")
```

ROC curve for Naïve Bayes

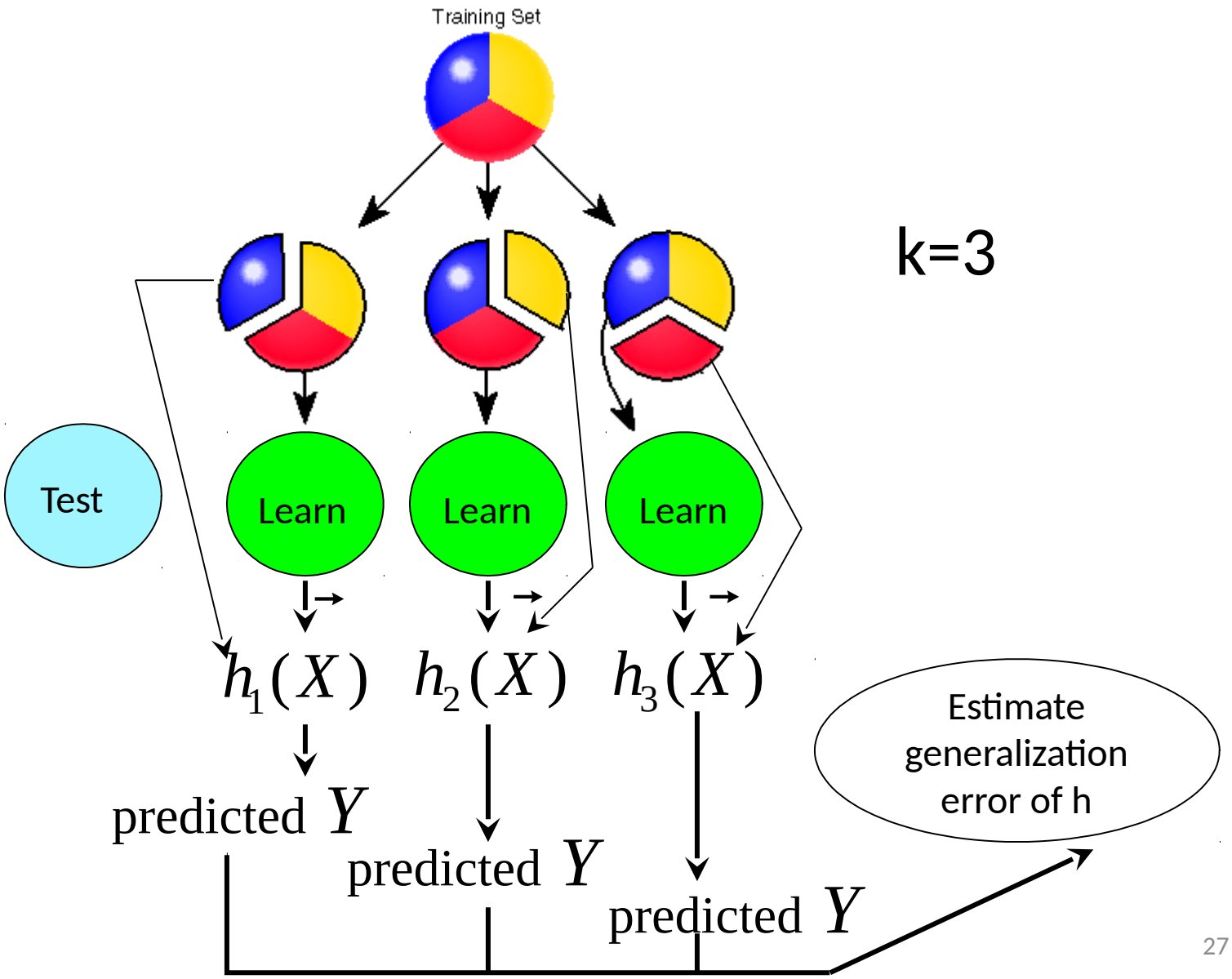


Classification of virginica

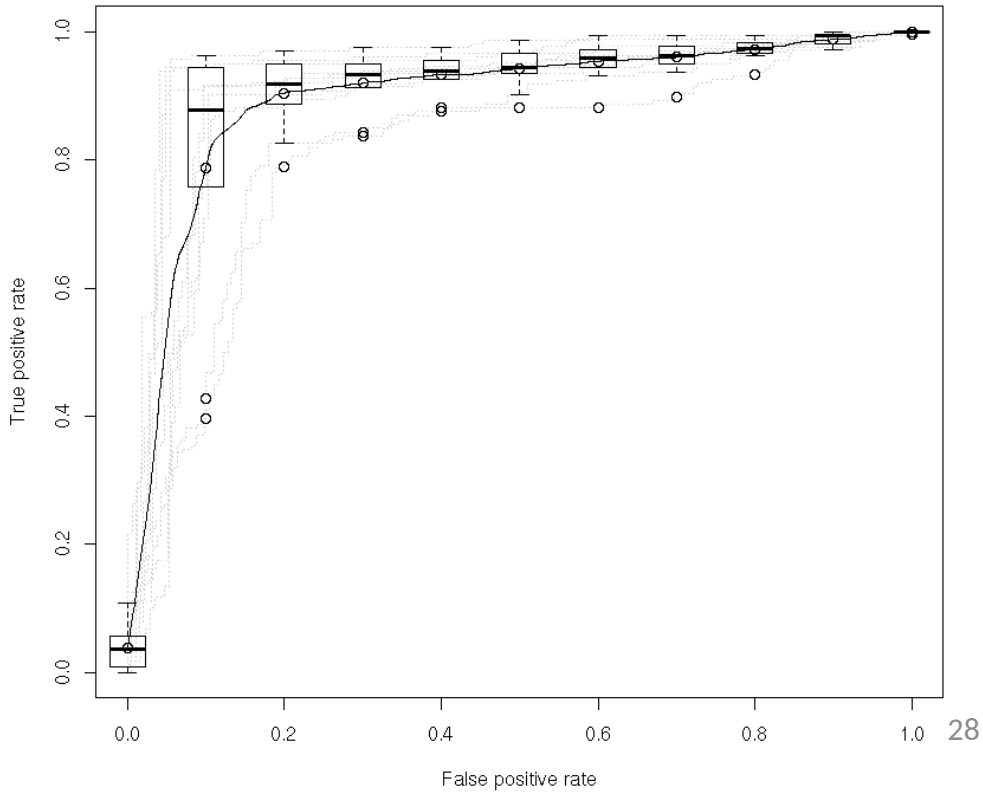
size(trainSet) ~= size(testSet)

```
testidx <- which(1:length(iris[,1])%%2 == 0)
```

k-fold Cross-Validation

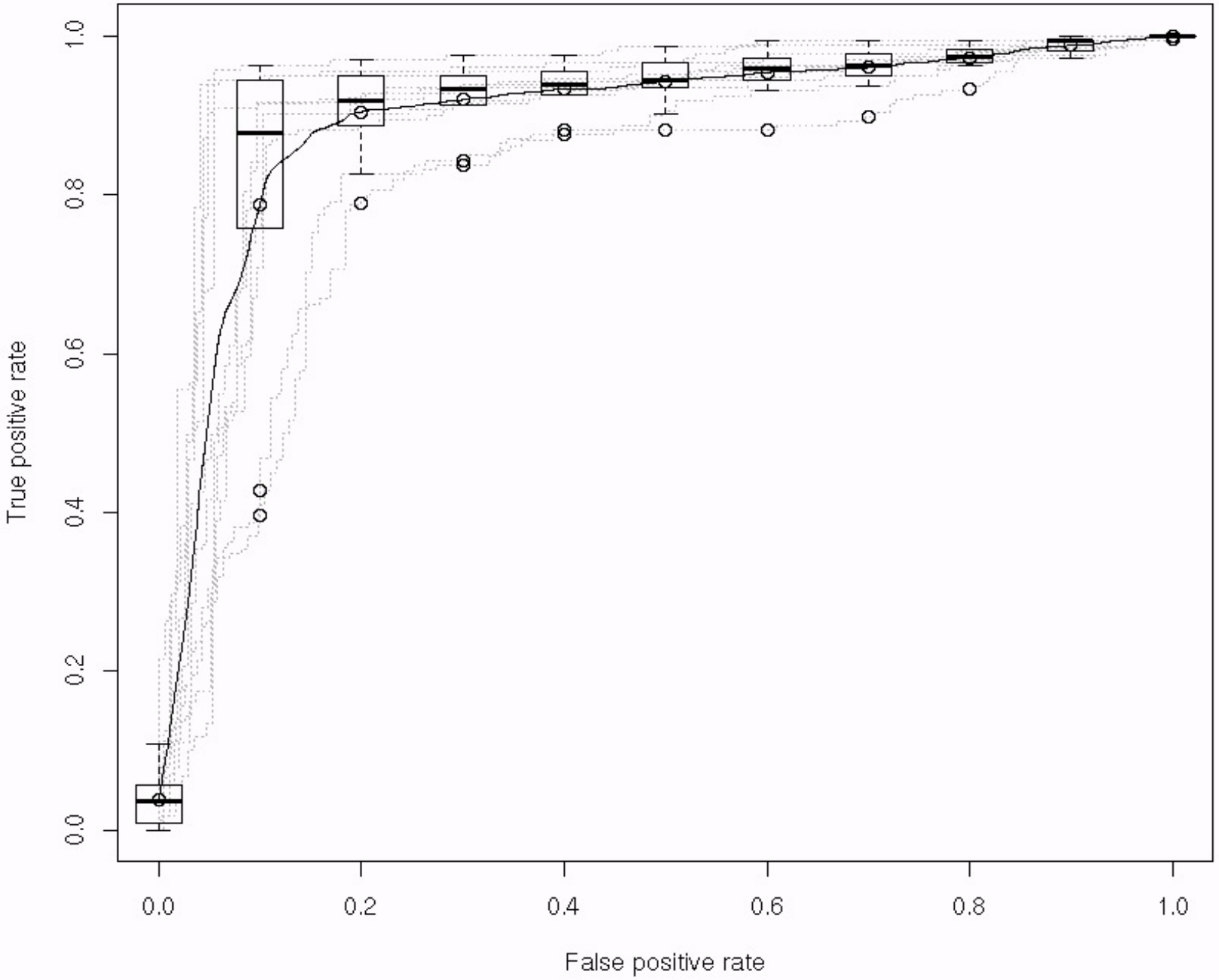


```
# plot ROC curves for several cross-validation runs (dotted
# in grey), overlaid by the vertical average curve and boxplots
# showing the vertical spread around the average.

data(ROCR.xval)
pred <- prediction(ROCR.xval$predictions, ROCR.xval$labels)
perf <- performance(pred,"tpr","fpr")

plot(perf,col="grey82",lty=3)
plot(perf, lwd=3,avg="vertical",spread.estimate="boxplot",add=T)
```



?ROCR.val

```
ROCR.xval                  package:ROCR                  R Documentation

Data set: Artificial cross-validation data for use with ROCR

Description:

     A mock data set containing 10 sets of predictions and
     corresponding labels as would be obtained from 10-fold
     cross-validation.

Usage:

     data(ROCR.xval)

Format:

     A two element list. The first element, 'ROCR.xval$predictions', is
     itself a 10 element list. Each of these 10 elements is a vector of
     numerical predictions for each cross-validation run. Likewise, the
     second list entry, 'ROCR.xval$labels' is a 10 element list in
     which each element is a vector of true class labels corresponding
     to the predictions.
```


True positive rate
False positive rate
0.0
0.2
0.4
0.6
0.8
1.0

Identify your expertise level

- Beginner
- Intermediate or Advanced

Beginner Exercise

- Get the spam database into R (install package kernlab)
- Randomly split into a training/test set of 80% to 20%
- Train and test SVM with 2 kernels of your choice
- Plot ROC curve for both SVM predictions
  - Save plot to a file named Figure1.pdf
  - Report AUC for each on legend
- 10-fold cross-validation on one of the SVM predictions
  - split data into 10 equally sized part

```
folds <- function(x,n) split(x, sort(rep(1:n,len=length(x))))
```

  - like in ROCR example, plot ROC curves for each cross-validation fold and corresponding predictions with average ROC curve and box plot (Figure2.pdf)

Intermediate/Advanced Exercise (page 1)

- Get the spam database into R (install package kernlab)
- Randomly split into a training/test set of 80% to 20%
- Train and test
  - Decision Tree
  - Naive Bayes
  - SVM with 2 kernels of your choice
- Plot ROC curve for all predictions
  - Save plot to a file named Figure1.pdf
  - Report AUC for each on legend
- 10-fold cross-validation on the Decision Tree and SVM models
  - split data into 10 equally sized part

```
folds <- function(x,n) split(x, sort(rep(1:n,len=length(x))))
```

  - like in ROCR example, plot ROC curves for each cross-validation fold and corresponding predictions with average ROC curve and box plot (Figure2.pdf)

Intermediate/Advanced Exercise (page 2)

- Can you improve the performance of any of the classifiers using feature selection?
  - Install the caret R package
    install.package("caret")
    library("caret")
  - Build a pairwise correlation matrix over all features in the spam database
  - For each pair of features with positive or negative correlation > 0.75 randomly select a feature to drop
  - Run all models (DT, NB, SVMx2) with the reduced feature set and calculate and plot performance as before (train/test, 10-fold CV) (Figure3.pdf, Figure4.pdf)
  - Repeat this procedure with a feature selection method of your choice (Figure5.pdf, Figure6.pdf)