R for Classification

Jennifer Broughton
Shimadzu Research Laboratory
Manchester, UK

jennifer.broughton@srlab.co.uk

2nd May 2013

SHIMADZU

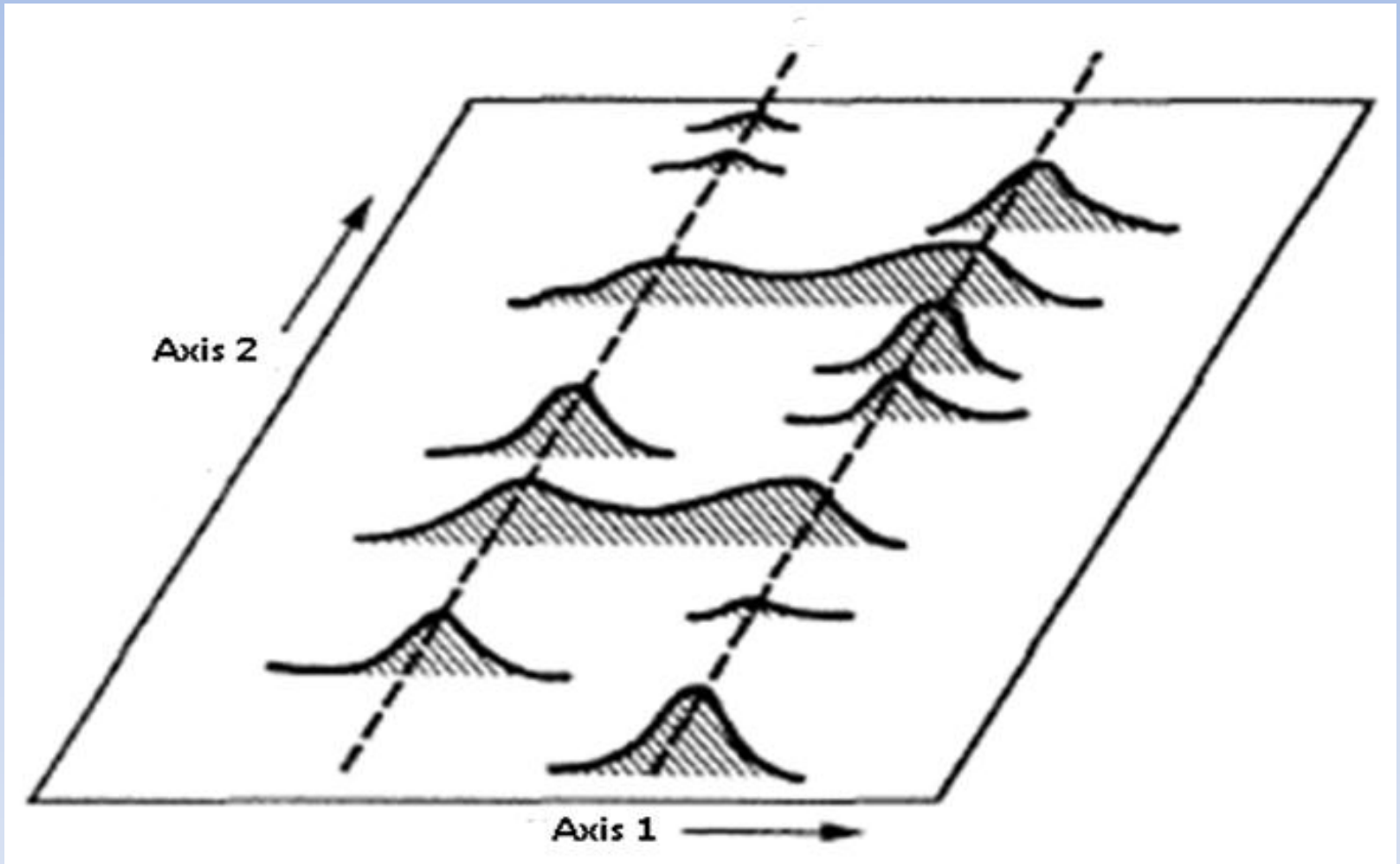
Classification?

Automatic Identification
of Type (Class) of Object
from Measured Variables (Features)

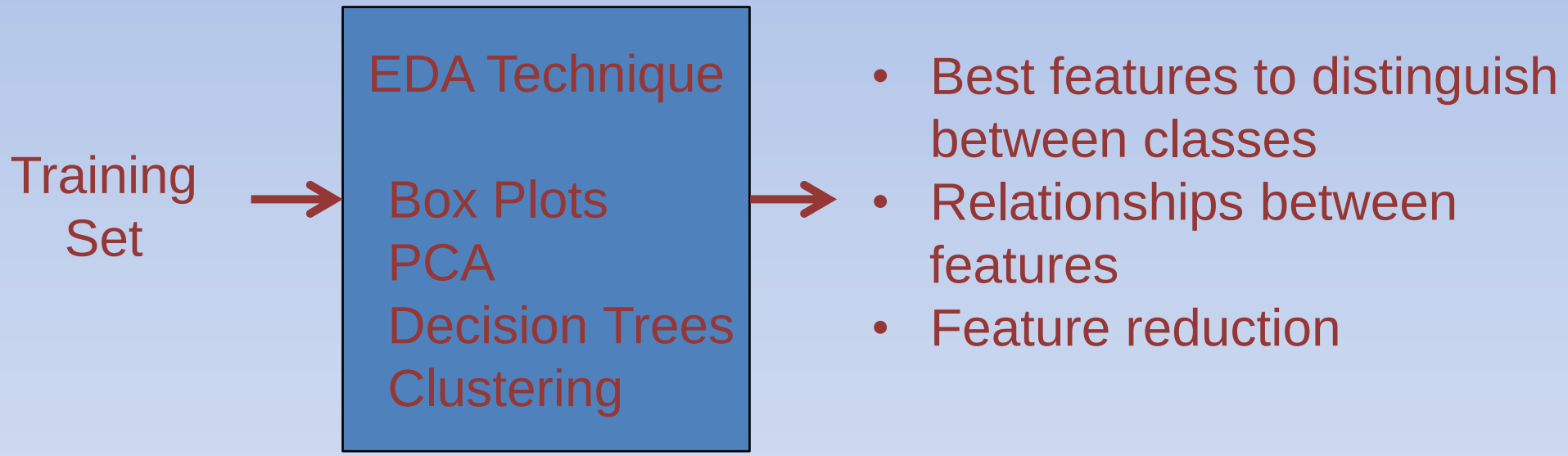
Object Type	Feature1	Feature2	Feature3	…….	Feature n
Label 1	val[1,1]	val[1,2]	val[1,3]	…….	val[1,n]
Label 2	val[2,1]	val[2,2]	val[2,3]	…….	val[2,n]
……	…….	…….	…….	…….	……
Label m	val[m,1]	val[m.2]	val[m,3]	…….	val[m,n]

SHIMADZU

Example Data



Data Preparation & Investigation



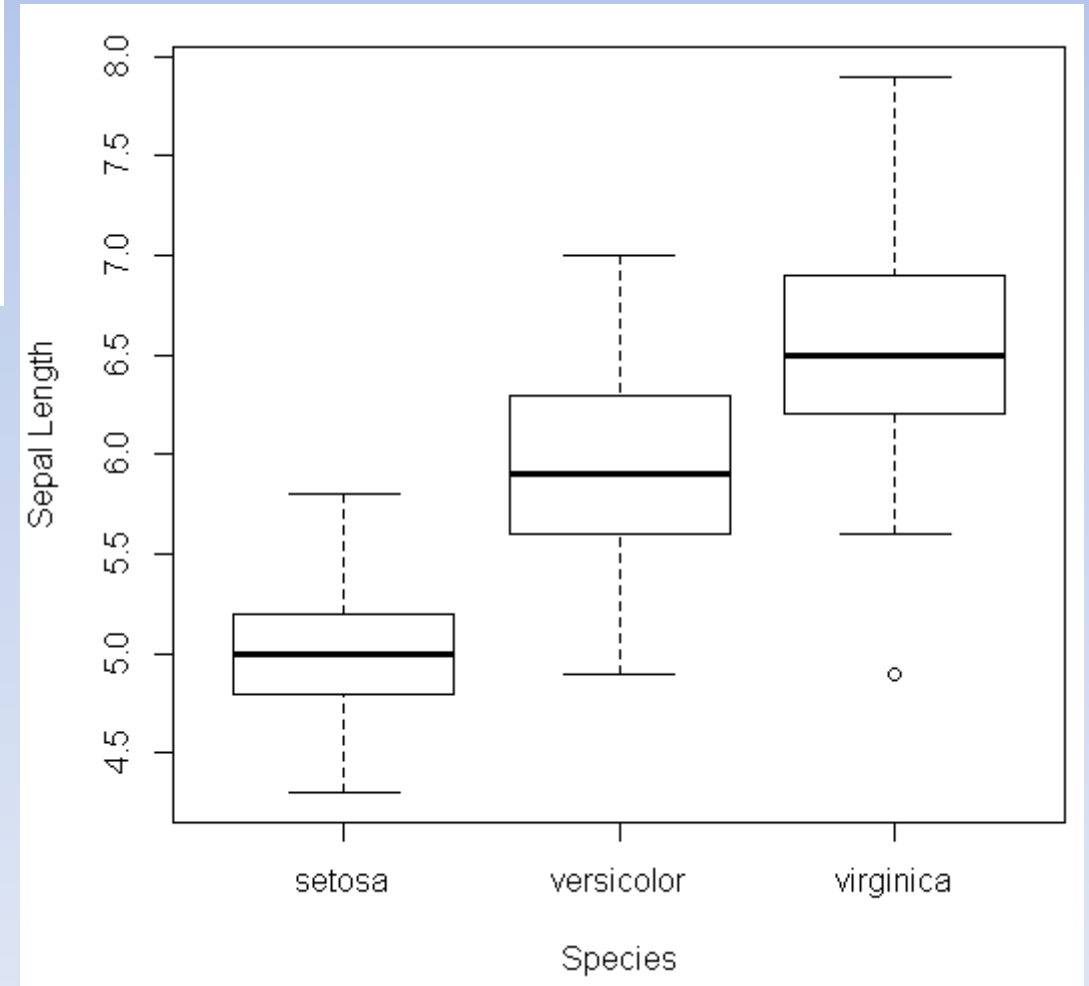
SHIMADZU

Box Plots

```
> 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
7            4.6         3.4          1.4         0.3     setosa
8            5.0         3.4          1.5         0.2     setosa
9            4.4         2.9          1.4         0.2     setosa
10           4.9         3.1          1.5         0.1     setosa
11           5.4         3.7          1.5         0.2     setosa
12           4.8         3.4          1.6         0.2     setosa
13           4.8         3.0          1.4         0.1     setosa
14           4.3         3.0          1.1         0.1     setosa
15           5.8         4.0          1.2         0.2     setosa
```

```
boxplot(iris$Sepal.Length ~ iris$Species, data=iris,
        xlab="Species", ylab="Sepal Length")
```

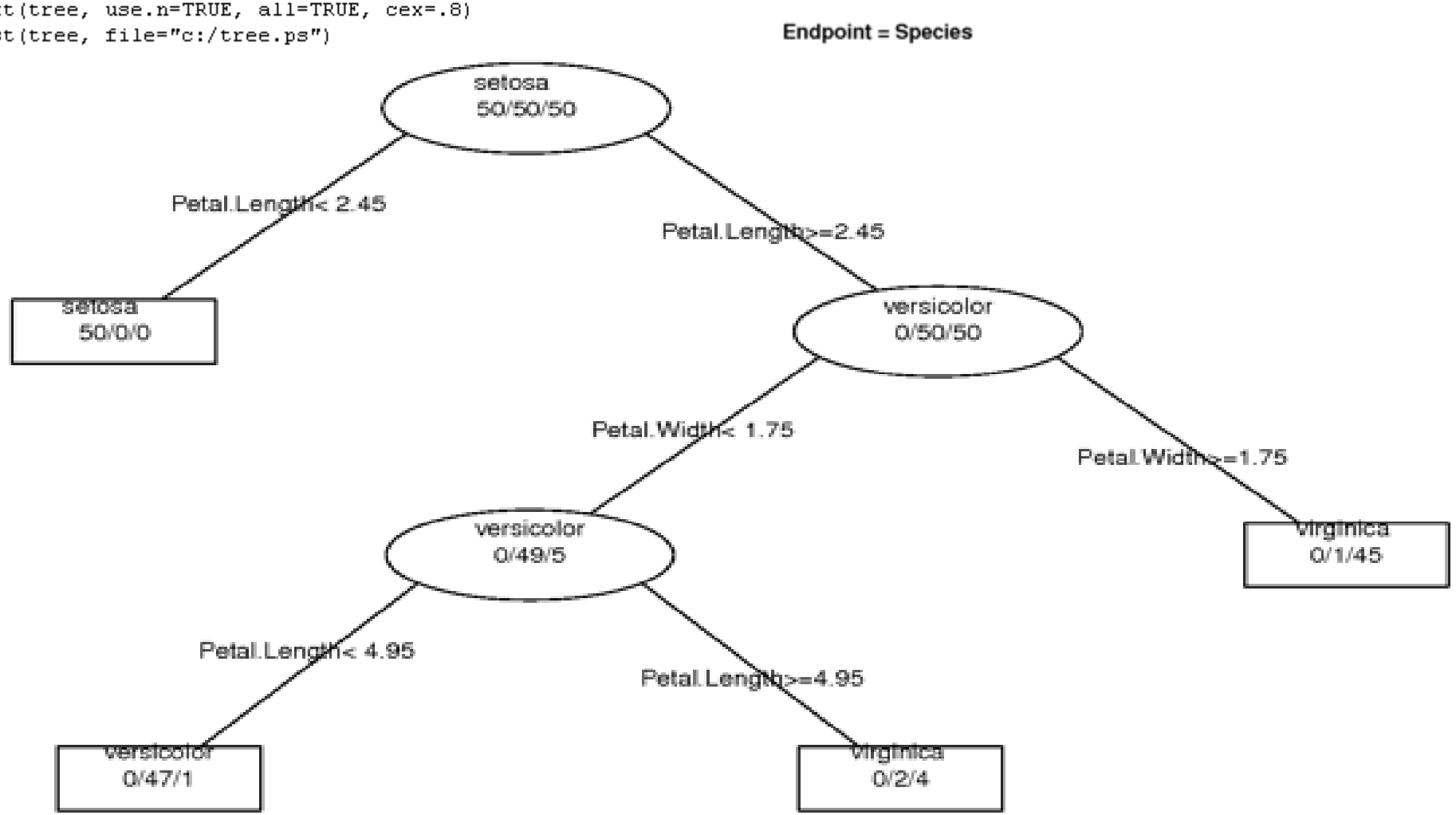
PCA & Multivariate Analysis:
ade4
FactoMineR



SHIMADZU

Example Classifier

```
require(rpart)
tree <- rpart(Species ~ ., method = "class", data=iris, control=rpart.control(minbucket=1))
plot(tree, uniform=TRUE)
text(tree, use.n=TRUE, all=TRUE, cex=.8)
post(tree, file="c:/tree.ps")
```

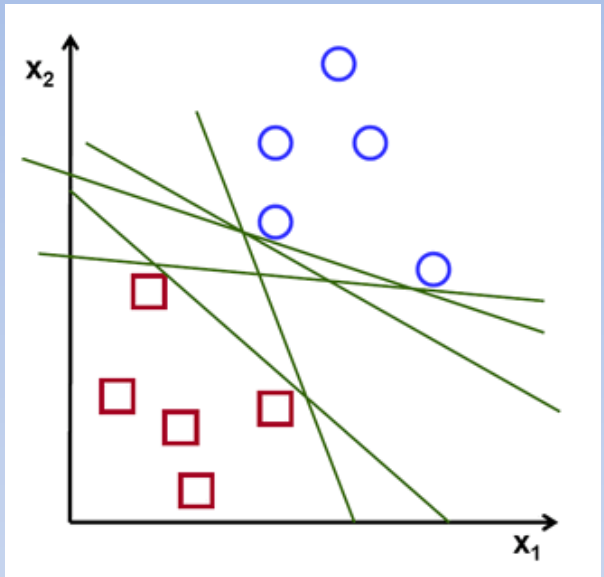


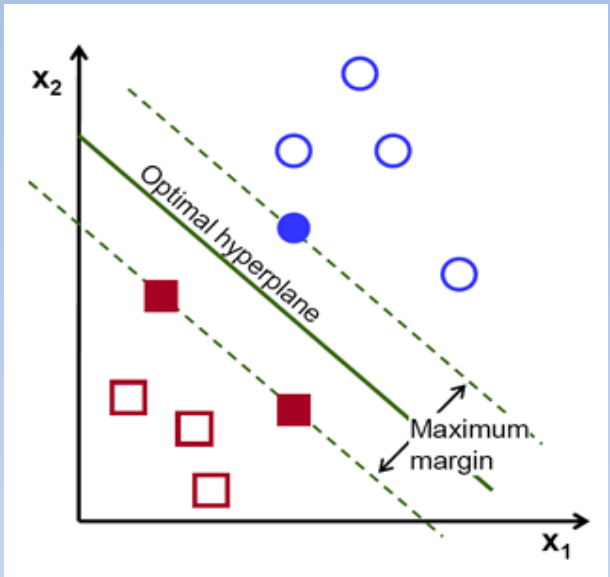
Classification Algorithms in R

Algorithm	R Package	Notes
Conditional Inference Tree	party	Selects variables for splitting according to their association with the response variable.
Recursive Partitioning Tree	rpart	Similar to ci tree but users an information measure to assess which variable should be split, e.g. the Gini Coefficient which estimates purity of the nodes.
Support Vector Machine	e1071	Interface to the libsvm implementation by Chih-Chung Chang and Chih-Jen Lin.
Neural Network	nnet	Feed forward neural networks with a single hidden layer.
Bootstrap Aggregation	ipred	Collection of samples chosen at random with replacement from the training set. Ensemble of rpart controls.
Random Forest	randomForest	Ported from original code in Fortran by Breiman and Cutler. Ensemble of rpart controls.
Adaptive Boosting	adabag	Implements Freund and Schapire's Adaboost algorithm. Ensemble of rpart controls.

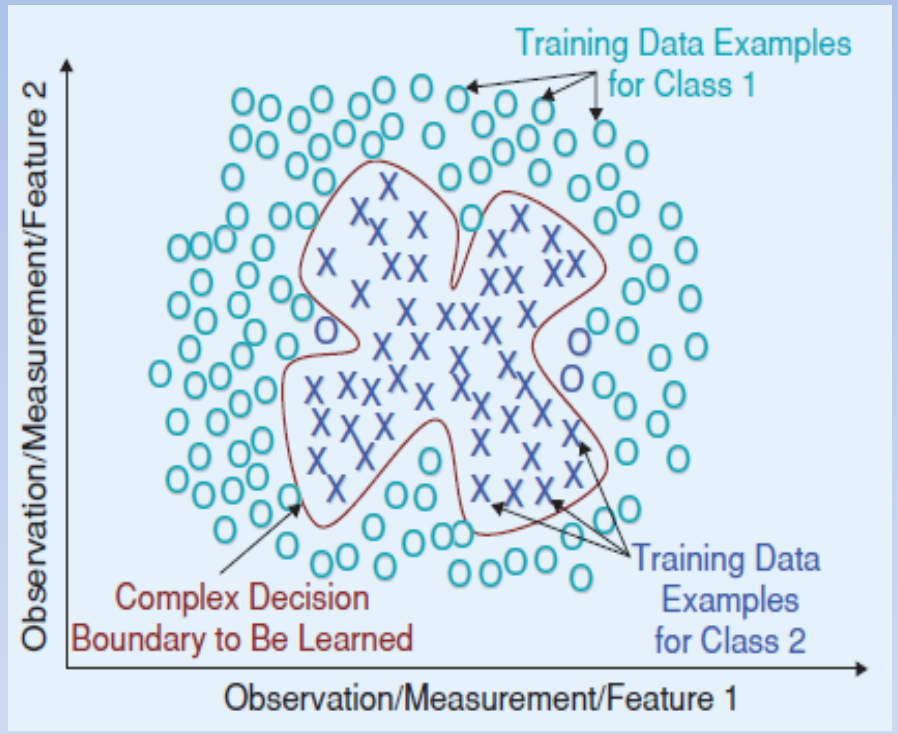
Rattle: R Analytical Tool to Learn Easily (*Rattle: A Data Mining GUI for R, Graham J Williams, The R Journal, 1(2):45-55*)

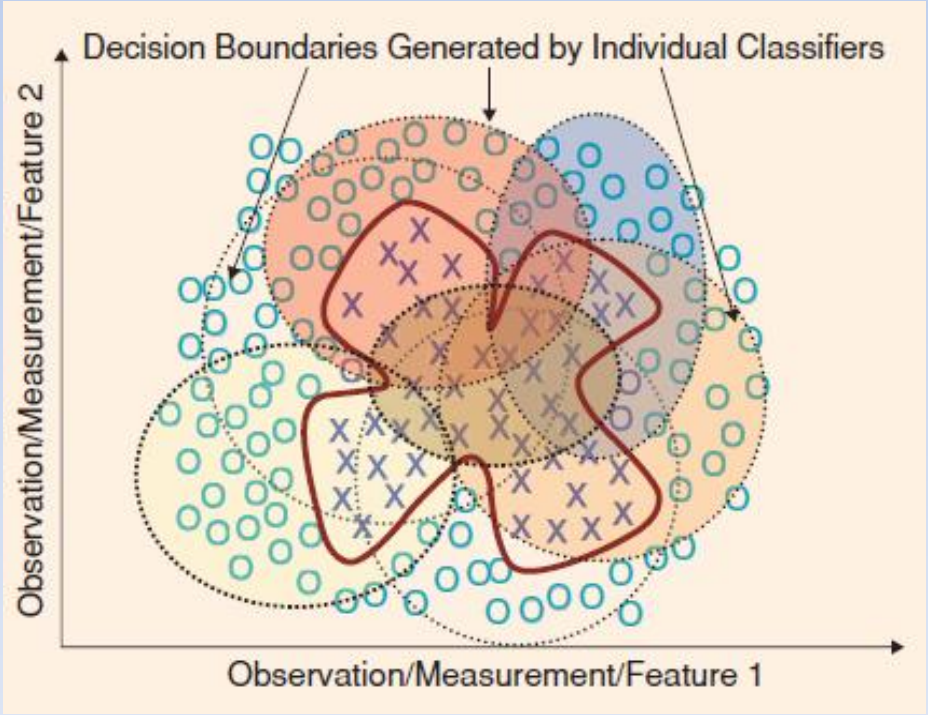
SVM





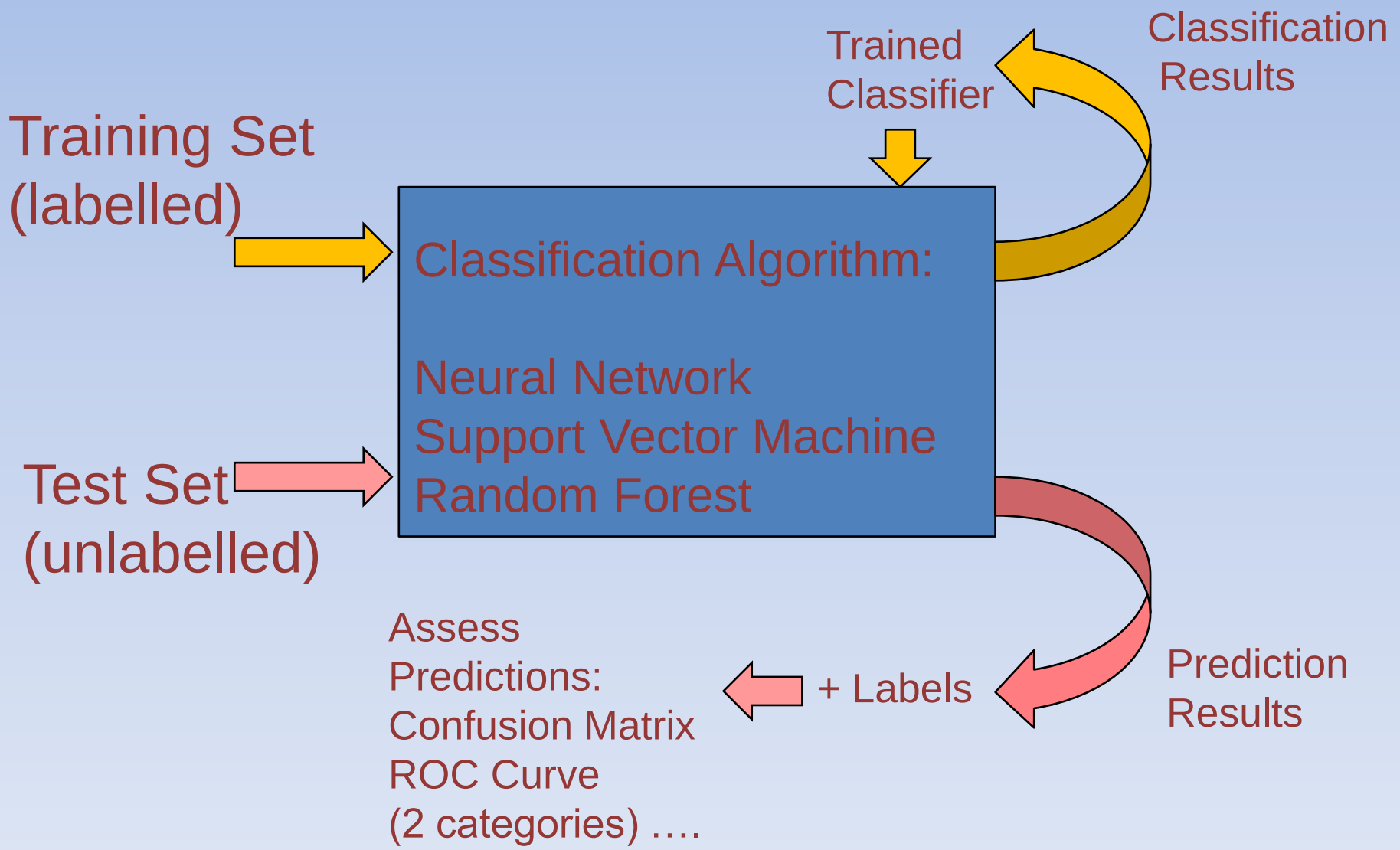
Ensemble Algorithm





SHIMADZU

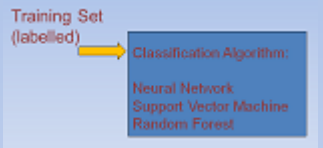
Training and Testing



SHIMADZU

Using Classifiers in R

Select Training Data

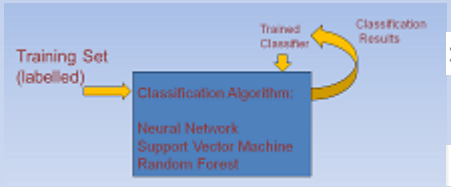



```
## set required columns
iCatCol <- 1
featureCols <- c(3:37)
```

```
## sample data
ind <- sample(2, nrow(dataSet), replace = TRUE, prob=c(0.75, 0.25))
```

Build Classifier

classifier ← algorithm(formula, data, options)



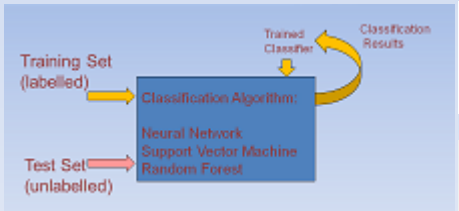

```
rf <- randomForest(dataSet[ind == 1, iCatCol] ~ ., data = dataSet[ind == 1, featureCols])
```

```
adaboost <- boosting(Category ~ ., data =dataSet[ind == 1, c(featureCols, iCatCol)]
```

(boosting and nnet)

Run Classifier

classifier.pred ← predict(classifier, newdata, options)




```
nnet.pred <- predict(nnt, newdata=peakSet[ind == 2, c(featureCols, iCatCol)], type="class")
rp.pred <- predict(rp, newdata=dataSet[ind == 2, featureCols], type="class")
rf.pred <- predict(rf, newdata=dataSet[ind == 2, featureCols])
```

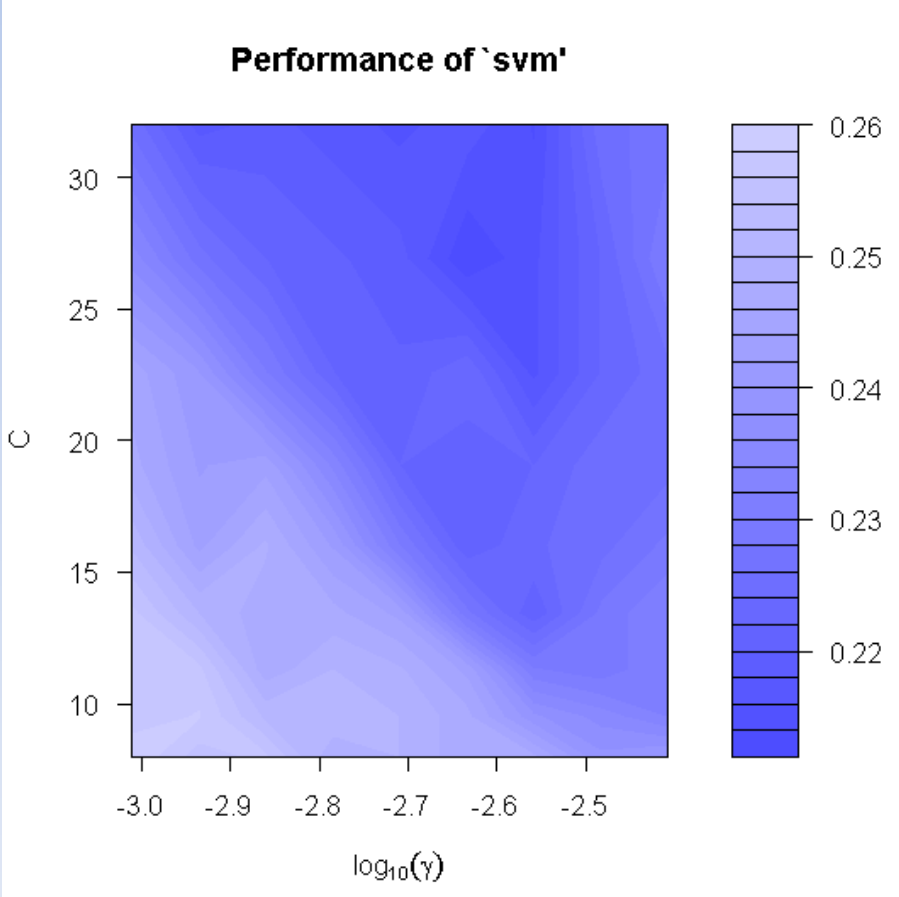
```
rp.prob <- predict(rp, newdata=dataSet[ind == 2, featureCols], type="prob")
nnet.prob <- predict(nnet, newdata=dataSet[ind == 2, featureCols], type="raw")
```

SHIMADZU

SVM & Neural Net Tuning

```
svm.tune <- tune.svm(Category~., data = dataSet[ind==1, c(featureCols, iCatCol)],
                     gamma = 2^seq(-10, -8, 0.25), cost = 2^seq(3, 5, 0.25),
                     tunecontrol = tune.control(sampling = "fix"))

summary(svm.tune)
bestGamma <- svm.tune$best.parameters[[1]]
bestC <- svm.tune$best.parameters[[2]]
plot(svm.tune, transform.x = log10, xlab = expression(log[10](gamma)), ylab = "C")
```

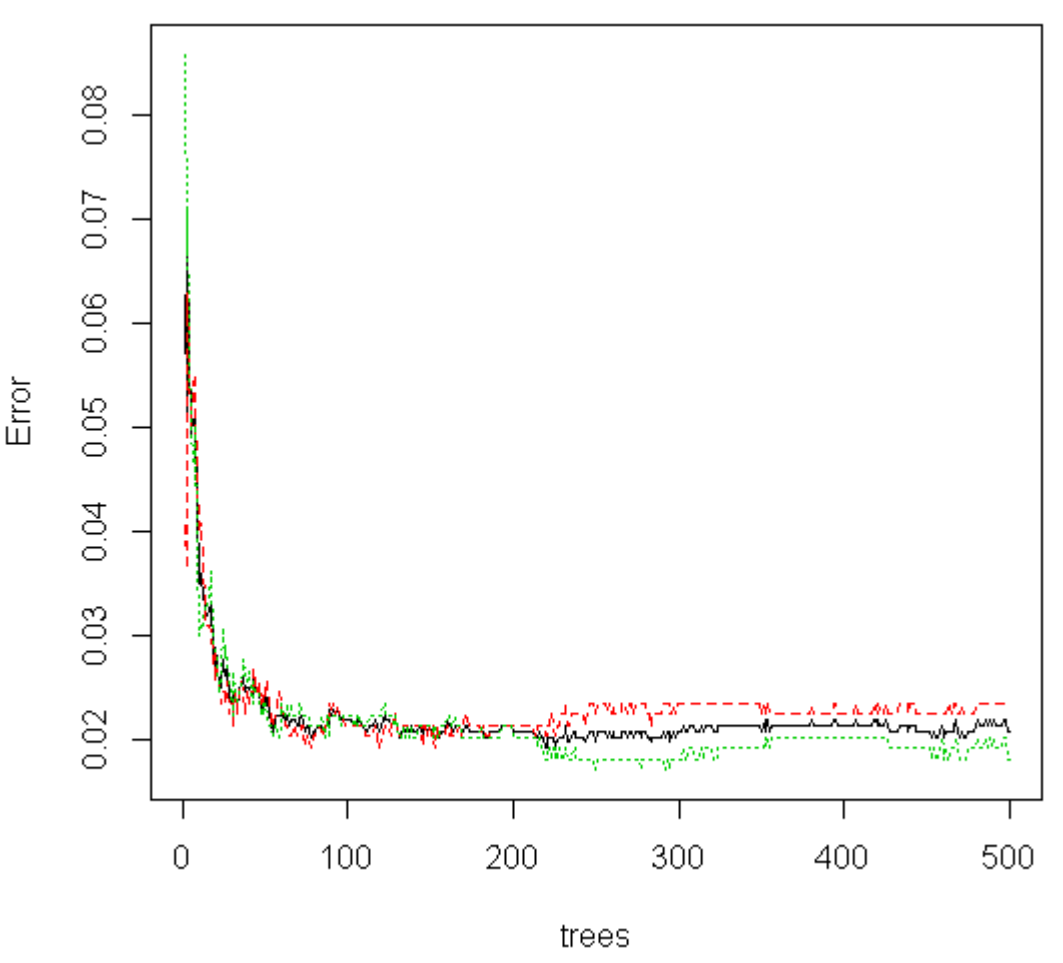


SHIMADZU

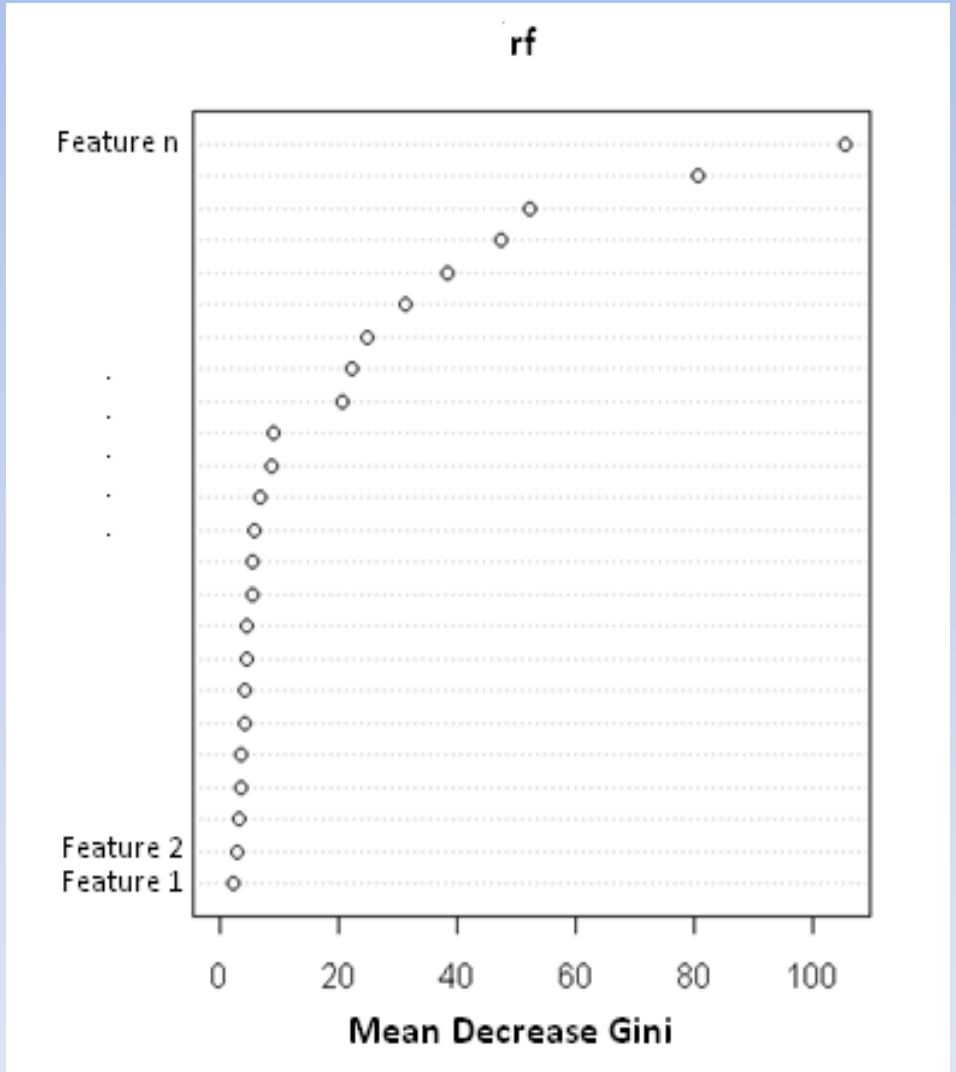
Classifier Feedback

print(classifier)
plot(classifier)

```
print(rf)
plot(rf)
```



`varImpPlot(rf)` high Gini Coefficient = high dispersion



SHIMADZU

Classifier Prediction Results

predict(type = “class”)

```
        27         80        101        139
    setosa versicolor  virginica  virginica
       134        102         30         77
 virginica  virginica     setosa versicolor
        72         33         70         93
```

predict(type = “prob”)

```
    setosa versicolor virginica
27   1.000      0.000     0.000
80   0.000      0.998     0.002
101  0.000      0.000     1.000
139  0.000      0.154     0.846
132  0.000      0.000     1.000
17   1.000      0.000     0.000
79   0.000      0.994     0.006
114  0.000      0.022     0.978
143  0.000      0.014     0.986
40   1.000      0.000     0.000
```

```
rf.pred <- predict(rf, newdata=iris[ind==2, 1:4], type="class")
rf.table <- table(pred=rf.pred, true= iris[ind==2, 5])
```

```
            true
pred         setosa versicolor virginica
  setosa          9          0         0
  versicolor      0          8         0
  virginica       0          0        16
```

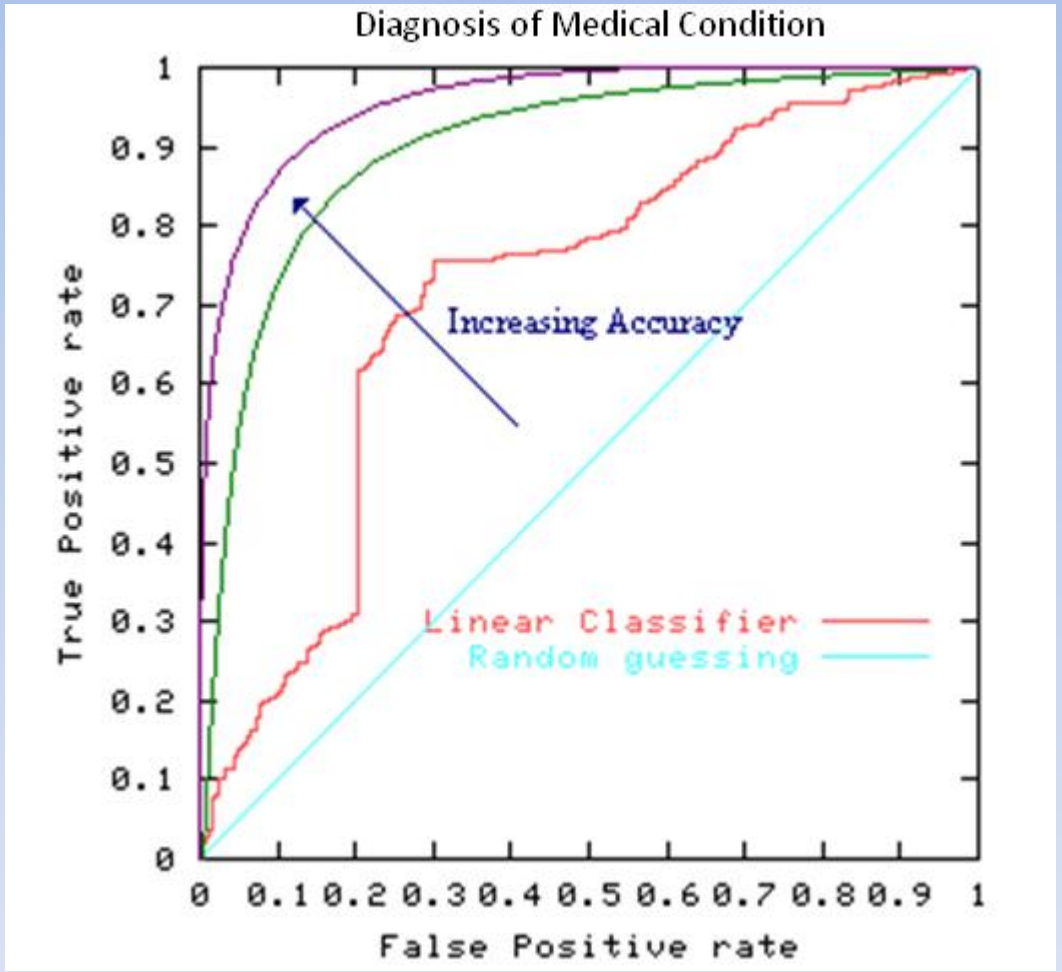
confusion matrix

Binary Classification Results

Class Detected? \ Class Present?	Y	N
Y	✓ True Positive	✗ False Positive
N	✗ False Negative	✓ True Negative

$$TPR = \frac{TP}{TP + FN} = Sensitivity$$

$$FPR = \frac{FP}{TN + FP} = 1 - Specificity$$



SHIMADZU

ROC Curves in R

ROCR package

```
require(ROCR)
rf.prob <- 1- unlist(predict(rf, newdata=dataSet[ind==2,featureCols],
                     type="prob"), use.names=F)[seq(1,nrow(dataSet[ind == 2,]))]

rf.prob.rocr <- prediction(rf.prob, dataSet[ind == 2, iCatCol])
rf.perf.rocr <- performance(rf.prob.rocr, "tpr", "fpr")
rf.auc <- performance(rf.prob.rocr, "auc")@y.values[(1)]

plot(rf.perf.rocr, main="ROC Curve", col=2, cex.lab = 0.75, cex.axis = 0.75)
legend("bottomright",legend=c(paste("Random Forest (auc=", format(x.rf.auc, digits=6)
                              col=c(2), cex= 0.75, lty=1)

rf.perf.lift <- performance(rf.prob.rocr,"lift","rpp")
plot(rf.perf.lift, main="lift curve", colorize=T)
```

```
rf.prob <- predict(rf, newdata=dataSet[ind==2,featureCols], type="prob")
```

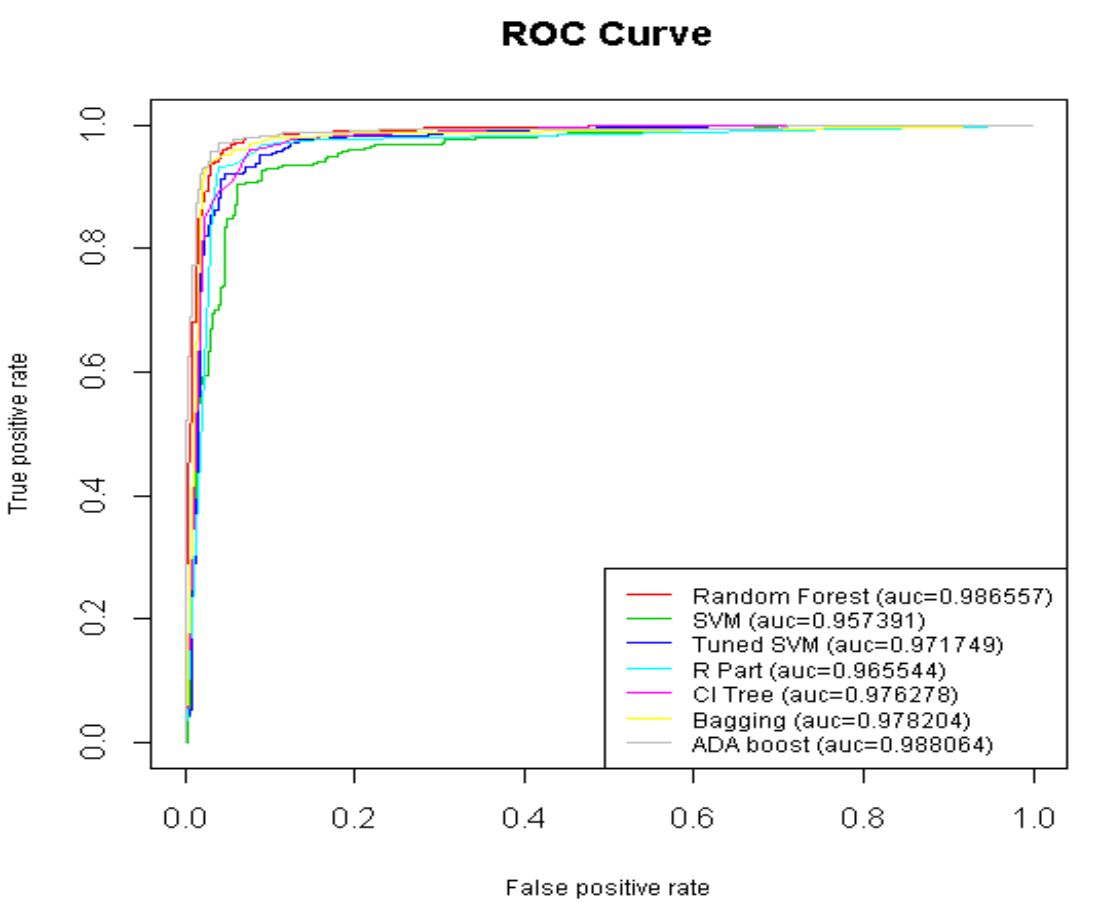
```
> rf.prob
  [1] 0.092 0.000 0.220 0.000 1.000 1.000 0.176 0.952
 [13] 0.000 0.040 0.354 0.856 0.020 1.000 0.002 0.018
 [25] 1.000 1.000 0.036 0.010 0.960 0.000 0.998 0.000
 [37] 0.002 0.040 0.002 0.032 0.020 0.022 0.006 0.008
 [49] 0.166 0.034 1.000 0.992 0.930 0.988 0.000 0.002
 [61] 0.618 0.918 0.058 0.000 0.000 1.000 0.008 0.938
```

```
rp.prob <- predict(rp, newdata=dataSet[ind==2,featureCols], type="prob")
```

```
> rp.prob
                FP           TP
5645 0.998815166 0.001184834
4891 0.998815166 0.001184834
3857 0.998815166 0.001184834
4728 0.998815166 0.001184834
336  0.008583691 0.991416309
715  0.008583691 0.991416309
```

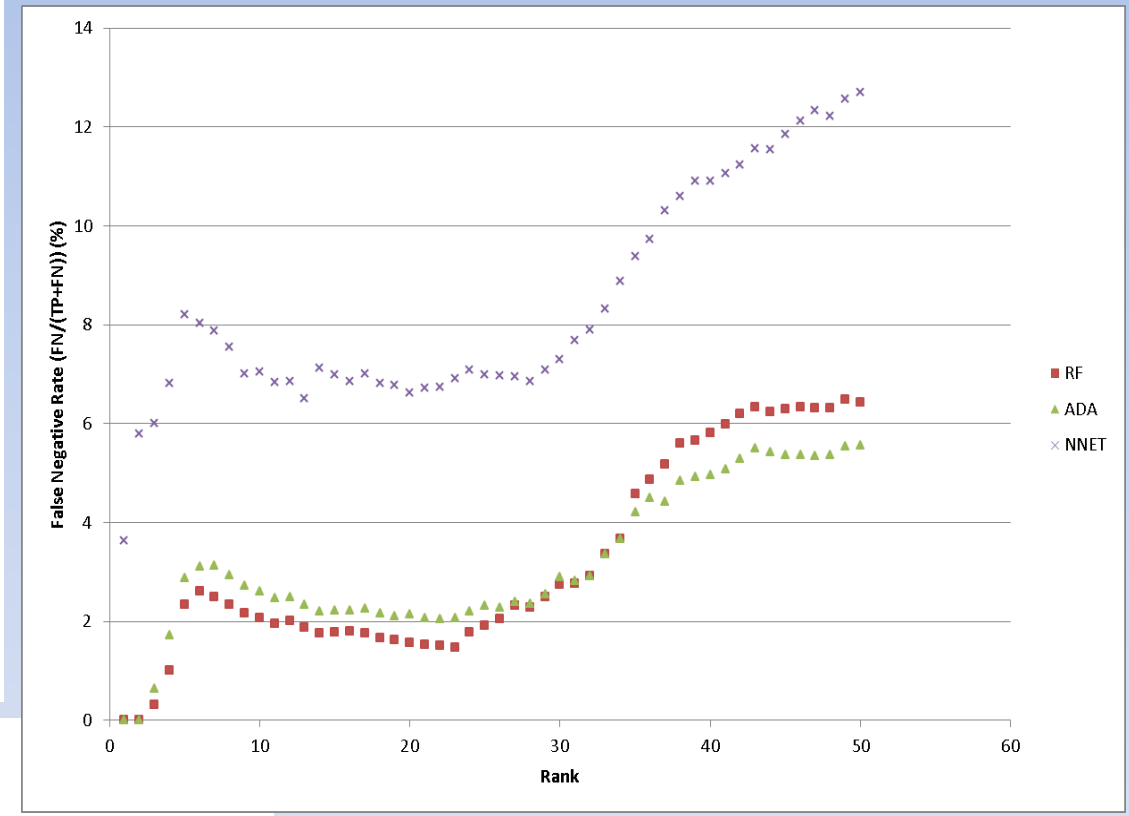
SHIMADZU

Example Results



Predicted	Actual	
	Type 1	Type 2
Type 1	1195	44
Type 2	78	1246

Predicted	Actual		
	Type 1	Type 2	Type 3
Type 1	1125	7	7
Type 2	11	190	11
Type 3	10	11	929




```
rank <- dataSet$Rank[ind == 2]
## convert to a data frame
ranks <- as.data.frame(table(rank))
algs <- c("rf", "ada", "nnet")
correct <- cbind(rf.correct, adaboost.correct, nnet.correct)
dfRanks <- array(0, c(nrow(ranks), 6, 3),
dimnames = list(1:nrow(ranks), c("rank", "freq", "TP", "TPR", "FN", "FNR"), algs))
dfRanks[,"rank",] <- as.numeric(levels(ranks$rank))[ranks$rank]
dfRanks[,"freq",] <- ranks$Freq
dfRanks[j,"TP",algs[i]] <- length((rank[correct[,i]==TRUE])[(rank[correct[,i]==TRUE])
            <= dfRanks[j,"rank",algs[i]]])
```

```
> ranks
  rank Freq
1    1   15
2    2   11
3    3    6
4    4   11
5  100  565
```