Classification using R

1. Example: Fish Data

Let us examine the fish dataset. A description of the variables is given next along with a picture that explains some of the variables.

```
Weight      Weight of the fish (in grams)
Length1     Length from the nose to the beginning of the tail (in cm)
Length2     Length from the nose to the notch of the tail (in cm)
Length3     Length from the nose to the end of the tail (in cm)
Height%     Maximal height as % of Length3
Width%      Maximal width as % of Length3
```

The frequencies of the 7 classes can be found in the next table.

SPECIES	Frequency	Percent	Cumulative Frequency	Cumulative Percent
bream	33	22.3	33	22.3
parki	10	6.8	43	29.1
perch	54	36.5	97	65.5
pike	16	10.8	113	76.4
roach	18	12.2	131	88.5
smelt	12	8.1	143	96.6
white	5	3.4	148	100.0

The next table gives the correlation matrix R for the 6 variables in the dataset.

	WEIGHT	L1	L2	L3	HEIGHT	WIDTH
WEIGHT	1.00	0.92	0.92	0.93	0.19	0.08
L1	0.92	1.00	0.99	0.99	0.02	-0.01
L2	0.92	0.99	1.00	0.99	0.04	-0.01
L3	0.93	0.99	0.99	1.00	0.12	-0.01
HEIGHT	0.19	0.02	0.04	0.12	1.00	0.44
WIDTH	0.08	-0.01	-0.01	-0.01	0.44	1.00

It can be seen that the three length variables are very highly correlated. This finding implies that the variables L2 and L3 contribute very little additional information. Whether this would have any consequences for classification remains to be seen. The parallel coordinate plot along with some 2-dim and 3-dim views of the data are given next.

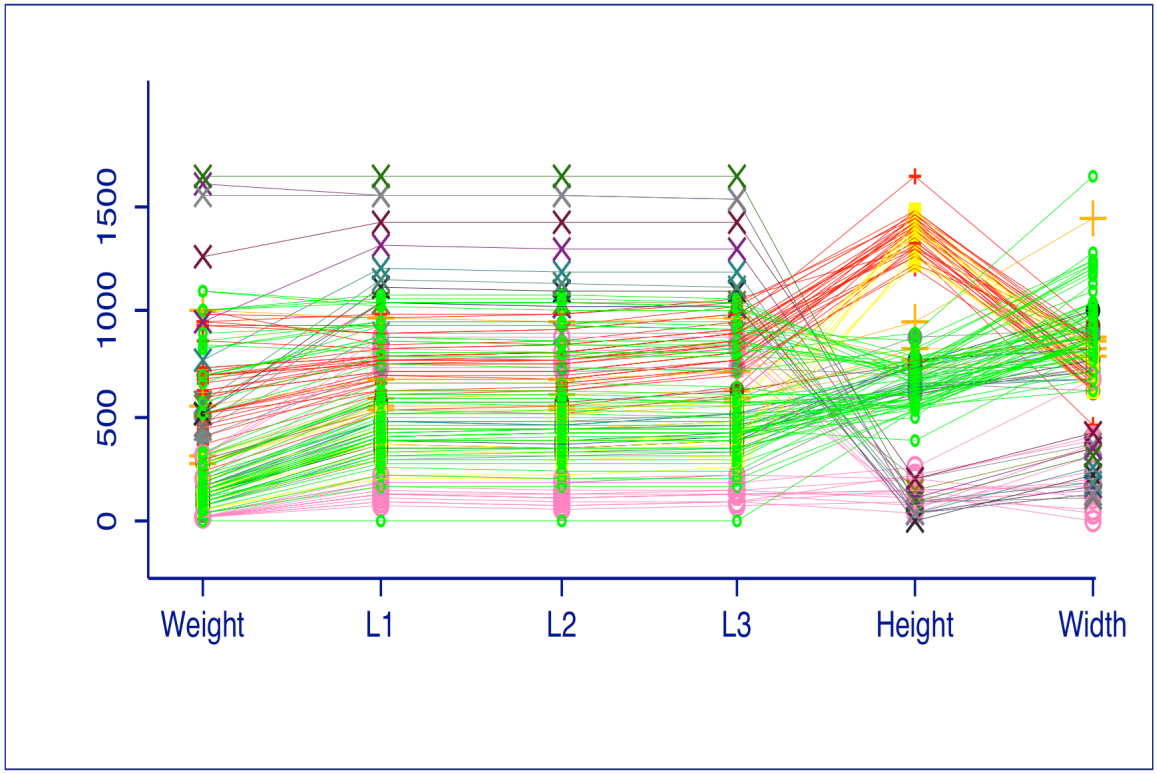

FIGURE 1. Parallel coordinates plot for the fish data. Bream: red +, White: orange big+, Roach: blue solid circle, Parki: yellow solid square, Smelt: pink open circle, Pike: cyan x, Perch: green small open circle

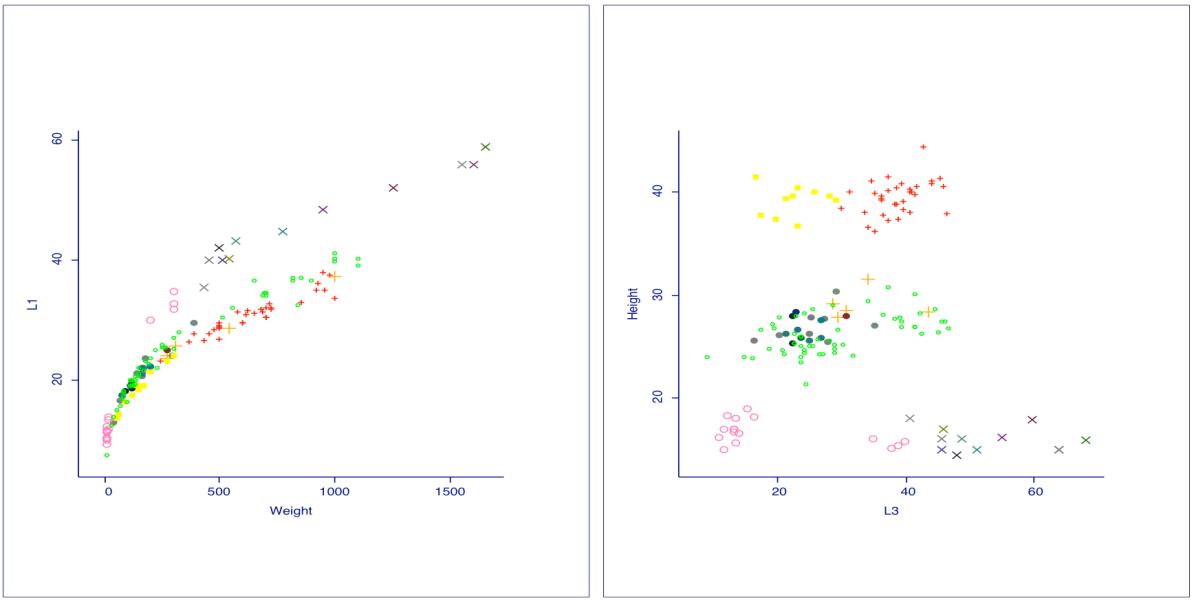

FIGURE 2. Scatterplots

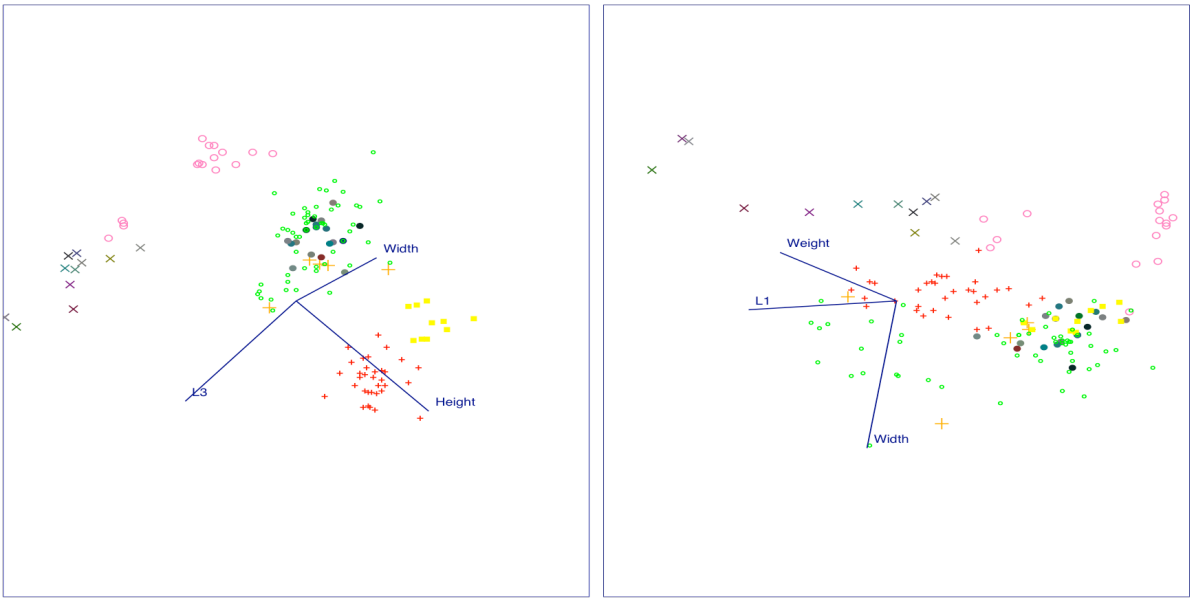

FIGURE 3. Scatterplots

In some of the plots we see a fairly nice separation of the classes. But given the very high correlations between the length variables we decided to compute the following new variables:

```
L21=L2-L1
L32=L3-L2
L31=L3-L1
```

The correlation matrix becomes

	WEIGHT	L1	L2	L3	HEIGHT	WIDTH	L21	L32	L31
WEIGHT	1.00								
L1	0.92	1.00							
L2	0.92	0.99	1.00						
L3	0.93	0.99	0.99	1.00					
HEIGHT	0.19	0.02	0.04	0.12	1.00				
WIDTH	0.08	-0.01	-0.01	-0.01	0.44	1.00			
L21	0.88	0.91	0.92	0.93	0.28	0.14	1.00		
L32	0.60	0.54	0.56	0.64	0.63	-0.06	0.66	1.00	
L31	0.75	0.72	0.74	0.80	0.55	0.01	0.84	0.96	1.00

It can be seen immediately that the correlations between the 3 length variables and the new variables vary substantially. The parallel coordinates plot for all 9 variables is given next and we can see that especially variable L32 seems promising when it comes to separating the 7 classes.

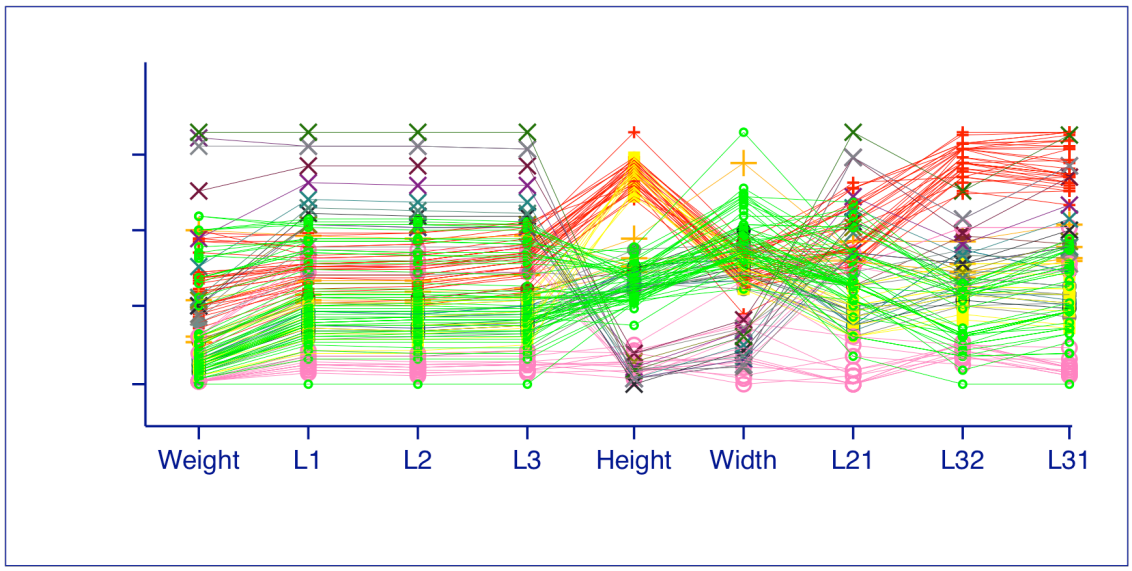

FIGURE 4. Parallel coordinates plot for the fish data (9 variables). Bream: red +, White: orange big+, Roach: blue solid circle, Parki: yellow solid square, Smelt: pink open circle, Pike: cyan x, Perch: green small open circle

(1) Classification Tree

We start by using the method of classification trees. The reason for choosing this method first is because it has a built-in mechanism for variable selection (through pruning). The following commands find the tree for the *original* set of variables. Notice that with the parameters minsplit=10 and minbucket=3 you control the growth of your tree. In this example I require that at least 10 objects must exist in a node for a split to be attempted (minsplit=10), and that in any *terminal* node at least 3 objects must be present. The default values are 20 and 7, which for this data set result in a very small tree that does not contain the class White.

** **Note that when you copy the following R commands, you need to retype all the quote symbols (“ and ”) in the R console.**

```
> fish<-read.table(“fish.dat”,h=T)
> library(rpart)
> fish.control<-rpart.control(minisplit=10,minbucket=3,xval=0)
> fish.treeorig<-rpart(Species~
  Weight+L1+L2+L3+Height+Width,data=fish,method=”class”,control=fish.control)
```

Let’s now plot the tree:

```
> plot(fish.treeorig)
> text(fish.treeorig)
```

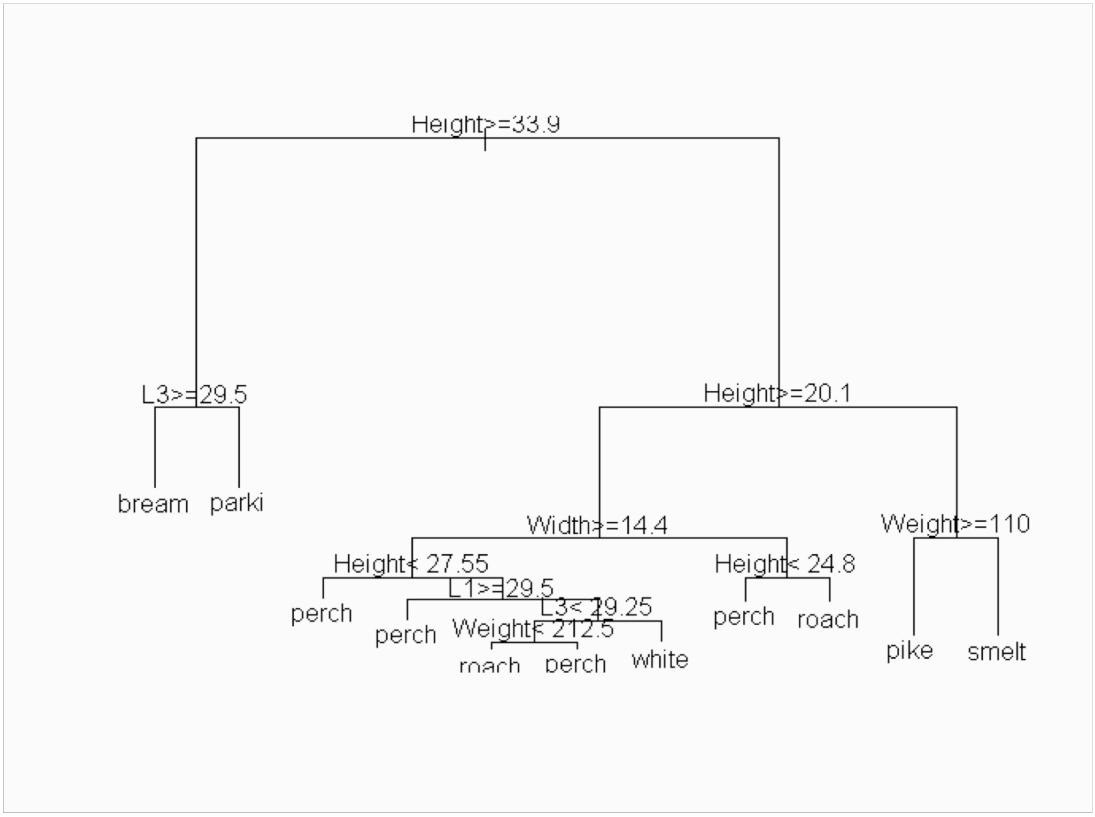

FIGURE 5. Classification tree using the original variables

Also check out the complexity parameter (CP):

```
> printcp(fish.treeorig)

Classification tree:
rpart(formula = Species ~ Weight + L1 + L2 + L3 + Height + Width,
      data = fish, method = "class", control = fish.control)

Variables actually used in tree construction:
[1] Height L1     L3     Weight Width

Root node error: 94/148 = 0.63514
          CP nsplit rel error
1 0.351064      0   1.00000
2 0.170213      1   0.64894
3 0.127660      2   0.47872
4 0.106383      3   0.35106
5 0.053191      4   0.24468
6 0.031915      5   0.19149
7 0.010638      6   0.15957
8 0.010000     10   0.11702
```

We know that this model used all the variables and that it misclassifies $94 \times 0.117 \approx 10$ objects. Hence its apparent misclassification rate is $10/148 \approx 7\%$. A more detailed view of the splits and of the other candidate variables for splitting at any node is given next:

```
> summary(fish.treeorig)

Call:
rpart(formula = Species ~ Weight + L1 + L2 + L3 + Height + Width,
      data = fish, method = "class", control = fish.control)

           CP nsplit  rel error
1 0.35106383      0 1.0000000
2 0.17021277      1 0.6489362
3 0.12765957      2 0.4787234
4 0.10638298      3 0.3510638
5 0.05319149      4 0.2446809
6 0.03191489      5 0.1914894
```

```
7 0.01063830          6    0.1595745
8 0.01000000         10    0.1170213

Node number 1: 148 observations,        complexity param=0.3510638
  predicted class=perch    expected loss=0.6351351
    class counts:       33       10       54       16       18       12        5
    probabilities: 0.223 0.068 0.365 0.108 0.122 0.081 0.034
  left son=2 (43 obs) right son=3 (105 obs)
  Primary splits:
      Height    < 33.9     to the right, improve=29.75863, (0 missing)
      Width     < 11.85    to the right, improve=17.98385, (0 missing)
      L3        < 29.7     to the right, improve=13.80398, (0 missing)
      L2        < 28.85    to the right, improve=12.96300, (0 missing)
      L1        < 26.1     to the right, improve=12.56245, (0 missing)

Node number 2: 43 observations,        complexity param=0.106383
  predicted class=bream    expected loss=0.2325581
    class counts:       33       10        0        0        0        0        0
    probabilities: 0.767 0.233 0.000 0.000 0.000 0.000 0.000
  left son=4 (33 obs) right son=5 (10 obs)
  Primary splits:
      L3        < 29.5     to the right, improve=15.348840, (0 missing)
      L2        < 26.15    to the right, improve=13.530660, (0 missing)
      L1        < 23.1     to the right, improve=13.407660, (0 missing)
      Weight    < 331.5    to the right, improve=12.015500, (0 missing)
      Width     < 14.85    to the right, improve= 1.063123, (0 missing)
  Surrogate splits:
      L1        < 23.1     to the right, agree=0.977, adj=0.9, (0 split)
      L2        < 25.2     to the right, agree=0.977, adj=0.9, (0 split)
      Weight    < 221      to the right, agree=0.953, adj=0.8, (0 split)

Node number 3: 105 observations,        complexity param=0.1702128
  predicted class=perch    expected loss=0.4857143
    class counts:        0        0       54       16       18       12        5
    probabilities: 0.000 0.000 0.514 0.152 0.171 0.114 0.048
  left son=6 (77 obs) right son=7 (28 obs)
  Primary splits:
      Height    < 20.1     to the right, improve=21.78355, (0 missing)
```

Width < 12.45 to the right, improve=20.93000, (0 missing)
Weight < 25.95 to the right, improve=13.35778, (0 missing)
L3 < 15.6 to the right, improve=10.68888, (0 missing)
L1 < 12.3 to the right, improve=10.63876, (0 missing)

Surrogate splits:

Width < 12.45 to the right, agree=0.990, adj=0.964, (0 split)
Weight < 25.95 to the right, agree=0.838, adj=0.393, (0 split)
L1 < 12.3 to the right, agree=0.819, adj=0.321, (0 split)
L2 < 13.35 to the right, agree=0.819, adj=0.321, (0 split)
L3 < 14.25 to the right, agree=0.819, adj=0.321, (0 split)

.
.
.

Since the resulting tree (Figure 5) has too many nodes, we prune it with the command

```
> fish.prunetree<-prune.rpart(fish.treeorig,cp=0.02)
> plot(fish.prunetree)
> text(fish.prunetree)
```

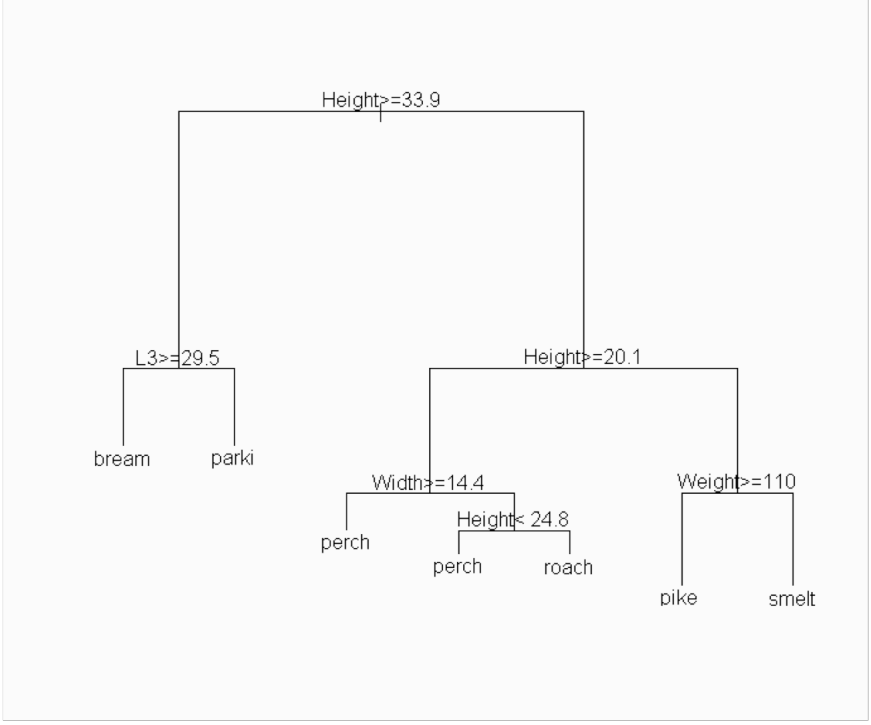

FIGURE 6. Pruned classification tree using original variables

where the parameter cp specifies the desired complexity and the resulting tree is given in Figure 6. Notice that the pruned tree totally misses the class “White”. Let’s examine what

happens if we use the new variables we defined for this data set, namely L21, L32, L31.

```
> L21<-fish$L2-fish$L1
> L32<-fish$L3-fish$L2
> L31<-fish$L3-fish$L1
> newfish<-cbind(fish,L21,L32,L31)
> newfish.treenew<-rpart(Species~., data=newfish,method="class",
  parms=list(split='information'),control=fish.control)
> printcp(newfish.treenew)
```

Classification tree:

rpart(formula = Species ~ ., data = newfish, method = "class",
parms = list(split = "information"), control = fish.control)

Variables actually used in tree construction:

[1] Height L21 L3 L32 Weight

Root node error: 94/148 = 0.63514

	CP	nsplit	rel error
1	0.351064	0	1.000000
2	0.170213	1	0.648936
3	0.127660	2	0.478723
4	0.106383	3	0.351064
5	0.095745	4	0.244681
6	0.053191	5	0.148936
7	0.047872	6	0.095745
8	0.010000	8	0.000000

We see that this tree has an apparent misclassification rate of 0% and uses only the variables Height, Weight, L21, L32 and L3. The resulting tree is given in Figure 7.

```
> plot(newfish.treenew)
> text(newfish.treenew)
```

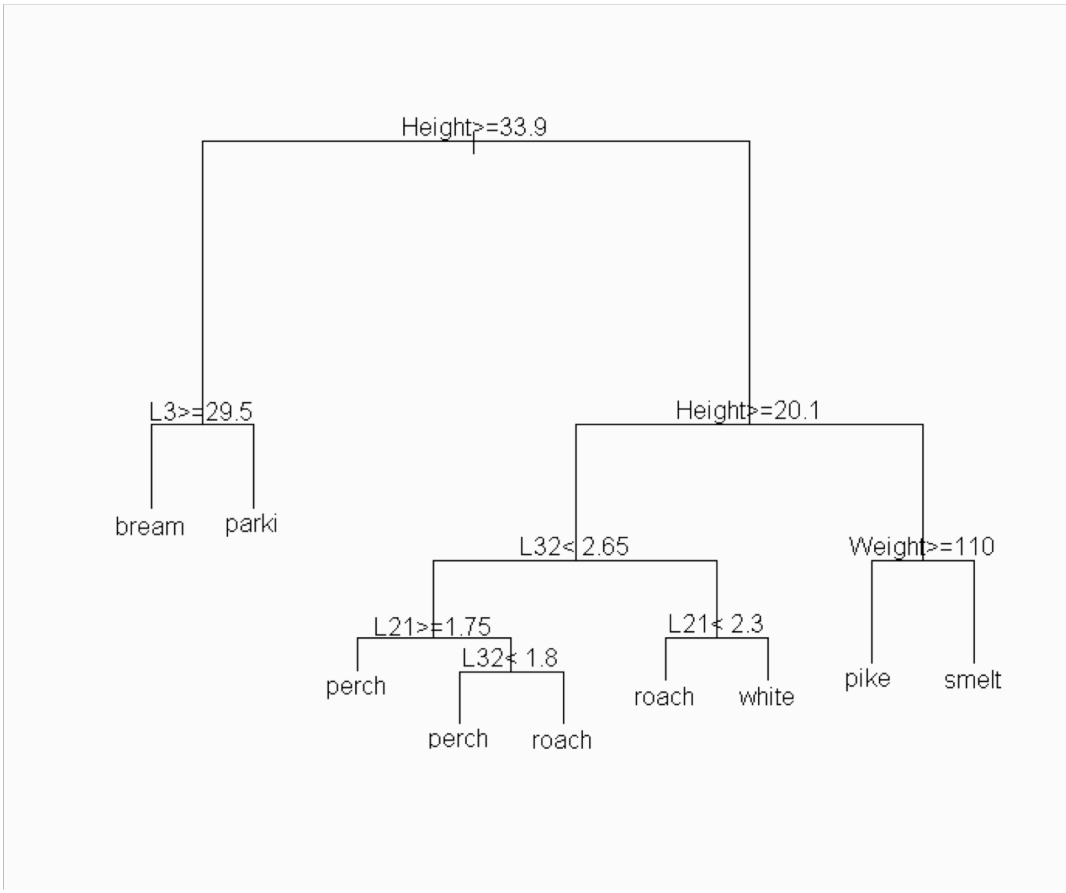

FIGURE 7. Classification tree using all the variables (including L21,L32,L31)

It is noted that the resulting tree has 9 terminal nodes. Somebody may decide to see whether pruning may reduce the size of the tree and retain such an excellent misclassification rate. But as you can find out smaller size trees miss the class White, which obviously increases the misclassification rate.

So far we have fitted our tree using the entire training set and thus were able to calculate the apparent misclassification rate. We repeat this exercise by using cross-validation in order to access the true misclassification rate. We can use cross-validation by changing the value of "xval" when defining the control parameters. In this instance I decided to cross-validate excluding one object at a time (leave-one-out).

```
> fish.control<-rpart.control(minbucket=3,minsplit=10,xval=148)
> newfish.treenewcv<- rpart(Species~., data=newfish,method="class",
  parms=list(split='information'),control=fish.control)
> printcp(newfish.treenewcv)
```

Classification tree:

```
rpart(formula = Species ~ ., data = newfish, method = "class",
      parms = list(split = "information"), control = fish.control)

Variables actually used in tree construction:
[1] Height  L21     L3      L32     Weight

Root node error: 94/148 = 0.63514

          CP nsplit  rel error  xerror     xstd
1 0.351064      0 1.000000 1.000000 0.062302
2 0.170213      1 0.648936 0.648936 0.063704
3 0.127660      2 0.478723 0.478723 0.059534
4 0.106383      3 0.351064 0.351064 0.053870
5 0.095745      4 0.244681 0.361702 0.054442
6 0.053191      5 0.148936 0.170213 0.040187
7 0.047872      6 0.095745 0.180851 0.041267
8 0.010000      8 0.000000 0.031915 0.018238
```

Either choosing the minimum "xerror" or using the 1-SE rule, we decide to keep the tree with "nsplit=8" (or size=9), which is shown in Figure 7. The plot of the relative error as a function of the complexity parameter and the size (nsplit+1) of the tree is shown in Figure 8.

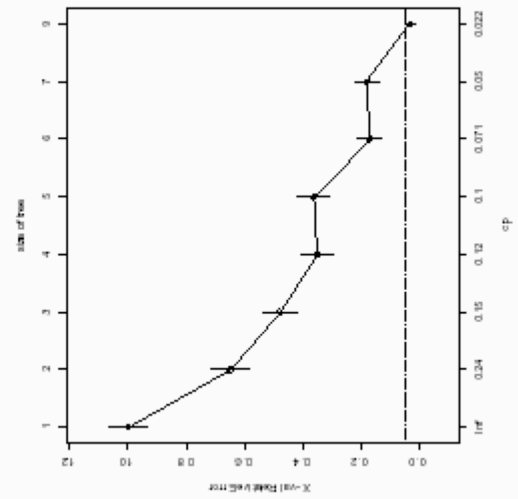

FIGURE 8. Relative error vs Complexity parameter & Size plot for the model containing all the variables

Note:

One may also perform a grid search for the parameters (minisplit, minbucket) so that the minimum value of "xerror" can be obtained based on CP.

Note: here "rel error" is related to the apparent error rate for the training data, while "xerror" corresponds to the estimated true error rate by CV. The true error rate for the above resulting tree is (94*0.0319)/148 ~ 2%.

Let us now examine how using this tree we can classify the new observations whose class identity is unknown. The following commands help us in our task.

```
> newfish.test<-read.table("fish_test.dat",h=T)
> L31<-newfish.test$L3- newfish.test$L1
> L32<-newfish.test$L3- newfish.test$L2
> L21<-newfish.test$L2- newfish.test$L1
> newfish.test<-cbind(newfish.test,L21,L32,L31)
> newfish.tpred<-predict(newfish.treenewcv,newfish.test)
> newfish.tpred
```

	bream	parki	perch	pike	roach	smelt	white
1	1	0	0	0	0	0	0
2	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0
4	0	0	1	0	0	0	0
5	0	0	0	1	0	0	0
6	0	0	0	0	0	1	0
7	0	0	0	0	0	1	0
8	0	1	0	0	0	0	0
9	0	0	0	0	1	0	0
10	0	0	0	0	1	0	0
11	0	0	0	0	0	0	1

So, the new observations are classified as

bream, bream, perch, perch, pike, smelt, smelt, parki, roach, roach, white.

(2) Linear Discriminant Analysis

We continue by applying LDA. Since we discovered that the 3 additional variables help us to better classify the fish, we keep them in any subsequent analysis.

```
> library(MASS)
> newfish.lda<-lda(Species~.,data=newfish)
```

Warning message:

variables are collinear in: lda.default(x, grouping, ...)

This program complains that the variables are collinear and we should do something about this. Hence we fit

```
> newfish.lda<-lda(Species~Weight+L1+Height+Width+L21+L32,data=newfish)
> newfish.lda
Call:
lda.formula(Species ~ Weight + L1 + Height + Width + L21 + L32,
    data = newfish)

Prior probabilities of groups:
     bream      parki      perch       pike      roach      smelt      white
0.22297297 0.06756757 0.36486486 0.10810811 0.12162162 0.08108108 0.03378378

Group means:
              Weight       L1   Height    Width      L21      L32
Bream     636.1818 30.60606 39.52727 14.10000 2.8060606 5.272727
parki     155.8000 18.62000 39.20000 14.18000 1.6100000 2.430000
perch     360.9333 25.31852 26.17778 15.78519 2.1259259 1.650000
pike      742.0625 42.88125 15.85625 10.48125 3.0375000 3.281250
roach     159.1111 20.66667 26.88333 14.57222 1.6388889 2.716667
smelt      11.5000 11.34167 16.99167 10.21667 0.6916667 1.091667
white     477.2000 27.82000 29.10000 15.76000 2.4800000 2.960000

Coefficients of linear discriminants:
                LD1          LD2          LD3          LD4          LD5
Weight  0.000911022 -0.002710071  0.007553399  0.001688806  0.006182751
L1      0.132200166  0.036926540 -0.259794107 -0.235599786 -0.330471903
Height -0.618519868 -0.332732865 -0.053863042 -0.330737436 -0.029226039
Width   0.464670922 -0.341184928 -0.353062958  0.842951264 -0.201141743
L21    -0.114071841  0.712452136 -2.278059990  0.277900320  2.700516892
L32    -2.311243186  2.141452146  0.539501848  1.803654269 -0.461925634
                LD6
Weight -0.003600115
L1     -0.119589009
Height -0.019796935
Width  -0.159484049
L21     2.813216431
L32    -0.080912628
```

Proportion of trace:

LD1	LD2	LD3	LD4	LD5	LD6
0.7998	0.1327	0.0473	0.0167	0.0035	0.0000

It can be seen that the first two LDs capture 93% of the between groups variance. The coefficients of the variables on the first 2 LDs suggest that we may decide to keep only Height, Width, L21 and L32. We next calculate the apparent error rate:

```
> newfish.ldapred<-predict(newfish.lda,newfish[,-1])
> table(newfish$Species,newfish.ldapred$class)
```

	bream	parki	perch	pike	roach	smelt	white
bream	33	0	0	0	0	0	0
parki	0	10	0	0	0	0	0
perch	0	0	54	0	0	0	0
pike	0	0	0	16	0	0	0
roach	0	0	0	0	18	0	0
smelt	0	0	0	0	0	12	0
white	0	0	0	0	1	0	4

The resulting misclassification rate is 0.6%. You can also estimate the true error rate using CV:

```
> newfish.ldacv<-lda(Species~Weight+L1+Height+Width+L21+L32,data=newfish,CV=T)
> table(newfish$Species,newfish.ldacv$class)
```

	bream	parki	perch	pike	roach	smelt	white
bream	33	0	0	0	0	0	0
parki	0	10	0	0	0	0	0
perch	0	0	54	0	0	0	0
pike	0	0	0	16	0	0	0
roach	0	0	0	0	18	0	0
smelt	0	0	0	0	0	12	0
white	0	0	0	0	1	0	4

The true error rate remains to be 0.6%.

A plot of the various groups on the first 2 LDs can be obtained by:

```
> eqscplot(newfish.ldapred$x,type="n",xlab="1st LD",ylab="2nd LD")
> fish.species<-c(rep("B",33),rep("W",5),rep("R",18),rep("Pa",10),rep("S",12),rep("Pi",16),
  rep("Pe",54))
> fish.colors<-c(rep(1,33),rep(2,5),rep(3,18),rep(4,10),rep(5,12),rep(6,16),rep(7,54))
> text(newfish.ldapred$x[,1:2],fish.species,col=fish.colors)
```

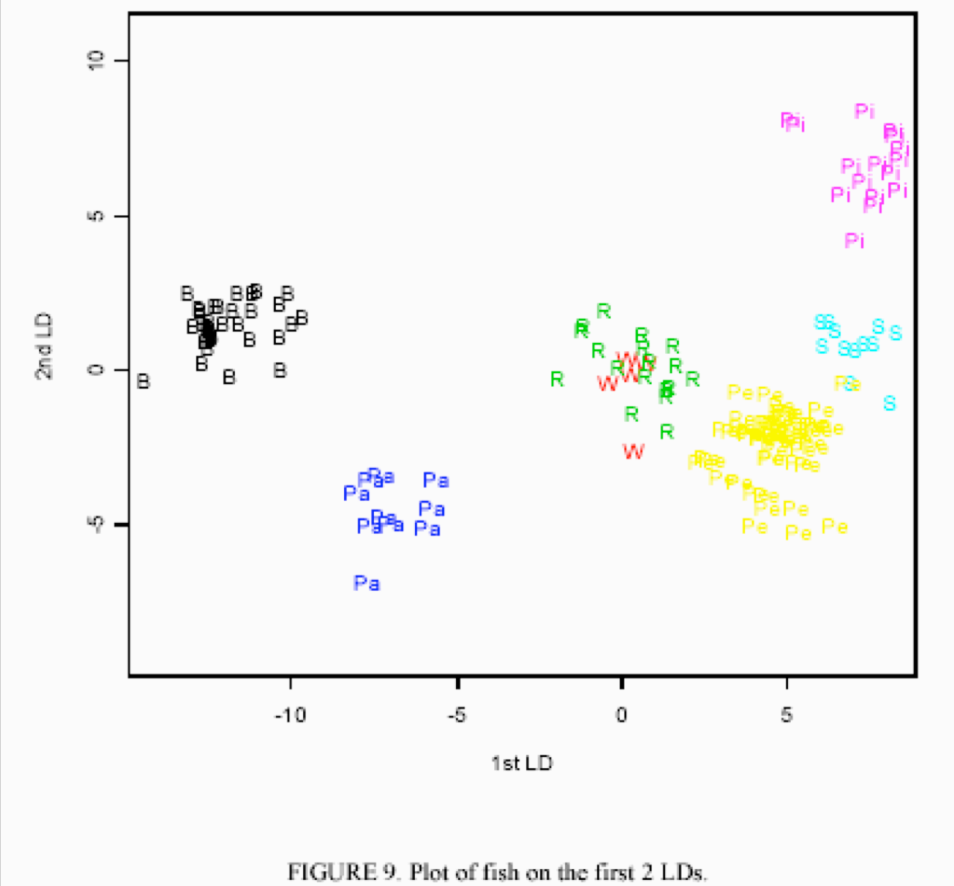

FIGURE 9. Plot of fish on the first 2 LDs.

To predict the class identities of the new data points we use:

```
> newfish.ldatest<-predict(newfish.lda,newfish.test)
> newfish.ldatest$class
```

[1] bream bream perch perch pike smelt smelt parki roach roach white

We see that the results agree with those obtained from the classification tree.

(3) Quadratic Discriminant Analysis

Let us examine how to apply QDA to this dataset.

Let us pretend that the multivariate normal assumption is valid (you can check for this by using the R package "MVN", as shown in the lab of CCA)

```
> newfish.qda<-qda(Species~.,data=newfish)
Error in qda.default(x, grouping, ...) : some group is too small for qda
```

The program complains that one of the groups is too small. Hence, we exclude all the data points from the group White (that only has 5 observations). Obviously, by removing one group we have changed the nature of the problem, but, nevertheless, let us continue.

```
> newfish.q<-read.table("newfish.qdat",h=T)
> newfish.qda<-qda(Species~.,data=newfish.q)
Error in qda.default(x, grouping, ...) : Rank deficiency in group bream
```

The program is again complaining about multicollinearity; so, we drop some variables. We decide to keep the same variables as in LDA, but we are free to choose other subset of variables.

```
> newfish.qda<-qda(Species~Weight+L1+Height+Width+L21+L32,data=newfish.q)
> newfish.qdapred<-predict(newfish.qda,newfish.q)

> table(newfish.q$Species,newfish.qdapred$class)
```

	bream	parki	perch	pike	roach	smelt
bream	33	0	0	0	0	0
parki	0	10	0	0	0	0
perch	0	0	54	0	0	0
pike	0	0	0	16	0	0
roach	0	0	0	0	18	0
smelt	0	0	0	0	0	12

We obtain 0% apparent error rate for the 6 group problem. We next predict the class identities of the new observations.

```
> predict(newfish.qda,newfish.test)$class
```

[1] bream bream perch perch pike smelt smelt parki roach roach perch

The last observation is classified as "perch", since the White class is missing. However, there is complete agreement between QDA and the previous two methods (Tree and LDA) for the first 10 observations.

How do the results change if we decide to cross-validate this model ?

```
> newfish.qda<-qda(Species~Weight+L1+Height+Width+L21+L32,data=newfish.q,CV=T)
> table(newfish.q$Species,newfish.qda$class)
```

	bream	parki	perch	pike	roach	smelt
bream	33	0	0	0	0	0
parki	0	10	0	0	0	0
perch	0	0	54	0	0	0
pike	0	0	0	16	0	0
roach	0	0	1	0	17	0
smelt	0	0	1	0	0	11

We see that the true error rate is estimated to be 1.4% for this problem, which as expected is higher than the apparent one.

(4) Nearest Neighbor Methods.

In order to use nearest neighbor method in R we need to use a new library. We next fit a model using all the variables.

```
> library(class)
> newfish.knn<-knn(newfish[,2:10],newfish[,2:10],newfish[,"Species"],k=3,prob=T)
> table(newfish$Species,newfish.knn)
```

	bream	parki	perch	pike	roach	smelt	white
bream	29	0	3	0	1	0	0
parki	0	6	2	0	2	0	0
perch	4	0	47	0	2	1	0
pike	1	0	2	12	1	0	0
roach	1	0	8	0	8	0	1
smelt	0	0	0	0	0	12	0
white	0	0	3	0	0	0	2

We see that the apparent error rate for k = 3 is about 21%. For k = 2, we have:

```
> newfish.knn<-knn(newfish[,2:10],newfish[,2:10],newfish[,"Species"],k=2,prob=T)
> table(newfish$Species,newfish.knn)
```

	bream	parki	perch	pike	roach	smelt	white
bream	30	0	1	0	1	0	1
parki	1	7	0	0	2	0	0
perch	1	0	46	0	4	1	2
pike	2	0	3	11	0	0	0
roach	1	0	5	0	12	0	0
smelt	0	0	0	0	0	12	0
white	0	0	0	0	0	0	5

We see that the apparent error rate for k = 2 is about 17%. Obviously, we are not particularly happy with the performance of k = 2 or k = 3. However, if we check out for k = 1, we get perfect classification:

```
> newfish.knn<-knn(newfish[,2:10],newfish[,2:10],newfish[,"Species"],k=1,prob=T)
> table(newfish$Species,newfish.knn)
```

	bream	parki	perch	pike	roach	smelt	white
bream	33	0	0	0	0	0	0
parki	0	10	0	0	0	0	0
perch	0	0	54	0	0	0	0
pike	0	0	0	16	0	0	0
roach	0	0	0	0	18	0	0
smelt	0	0	0	0	0	12	0
white	0	0	0	0	0	0	5

To be comparable with the previous approaches, one might select the same variables: Species, Weight, L1, Height, L21, L32 and repeat this analysis.

If we choose k =1, the true error rate can be estimated using CV (leave-one-out):

```
> newfish1<-newfish[,c(1,2,3,6,8,9)]
> newfish.knncv<-knn.cv(newfish1[,2:6],newfish1[,"Species"],k=1,prob=T)
> table(newfish1$Species,newfish.knncv)
```

	bream	parki	perch	pike	roach	smelt	white
bream	26	0	4	0	2	0	1
parki	1	4	0	0	4	0	1
perch	3	0	37	0	11	1	2
pike	2	0	4	9	0	0	1
roach	2	0	10	0	5	0	1
smelt	0	0	0	0	0	12	0
white	0	0	3	0	0	0	2

This conducts to a rather large true error rate 53/148 = 35.8%.

Note that for k = 2 and 3, the true error rates are 42.6% and 39.9% respectively.

This implies that the NN methods did not do very well using only these 6 variables.

However, if we have decided to choose this NN-1 model for prediction (of course you can try using other variables), the resulting classification for the test data is then:

```
> newfish1.test<-newfish.test[,c(1,2,5,7,8)]
> newfish.knntest<-knn(newfish1[,2:6],newfish1.test,newfish1[,"Species"],k=1,prob=T)
> newfish.knntest
```

```
 [1] bream bream perch white perch smelt smelt parki perch perch perch
```

(5) Logistic Discrimination

Again, let us pretend that all assumptions (including "multivariate normal" and "common covariance matrix" over all classes) are valid (need to check that in practice).
In order to use this method in R, we need another package.

```
> library(nnet)
> newfish.logd<-multinom(Species~.,data=newfish,maxit=250)
> newfish.logd
```

```
Call:
 multinom(formula = Species ~ ., data = newfish, maxit = 250)
```

Coefficients:

	(Intercept)	Weight	L1	L2	L3	Height
parki	-29.45533	0.02917110	6.349592	17.8259066	-23.500970	9.645257
perch	-80.11405	0.16021628	3.267803	56.6489214	-53.765482	6.684178
pike	15.22567	-0.05874368	8.093673	0.9753102	-3.095179	-13.084687
roach	-277.16409	-0.51539077	54.195310	-43.6844446	4.362472	-2.952463
smelt	455.64639	0.18459382	29.363751	-20.5072503	-10.290211	-13.228223
white	-57.01255	0.19991067	-17.467221	31.7667558	-20.454096	-4.118171

	Width	L21	L32	L31
parki	3.247584	11.476314	-41.326877	-29.850563
perch	21.052273	53.381118	-110.414404	-57.033286
pike	21.652958	-7.118363	-4.070489	-11.188852
roach	40.080836	-97.879755	48.046916	-49.832838
smelt	18.368009	-49.871001	10.217040	-39.653961
white	26.549555	49.233977	-52.220851	-2.986874

Residual Deviance: 2.009659e-11

AIC: 84

We see that this method uses one of the classes as the baseline one (here is “bream”) and then fits a separate model for the remaining classes. The performance of this method can be accessed by:

```
> table(newfish$Species,predict(newfish.logd,newfish))
```

	bream	parki	perch	pike	roach	smelt	white
bream	33	0	0	0	0	0	0
parki	0	10	0	0	0	0	0
perch	0	0	54	0	0	0	0
pike	0	0	0	16	0	0	0
roach	0	0	0	0	18	0	0
smelt	0	0	0	0	0	12	0
white	0	0	0	0	0	0	5

➔ resulting in 0% misclassification rate.

Now we examine the true error rate by cross-validation. To do this, we use another package

called “glmnet”:

```
> install.packages('glmnet')
> library(glmnet)
> x <- as.matrix(newfish[,-1])
> y <- newfish$Species
> cvfit <- cv.glmnet(x, y, family='multinomial', type.measure='class', nfolds=148)
> predict.value <- predict(cvfit, x, s = "lambda.min", type = "class")
> table(predict.value,newfish$Species)

  predict.value bream parki perch pike roach smelt white
        bream     33     0     0    0     0     0     0
        parki      0    10     0    0     0     0     0
        perch      0     0    54    0     0     0     0
        pike       0     0     0   16     0     0     0
        roach      0     0     0    0    18     0     0
        smelt      0     0     0    0     0    12     0
        white      0     0     0    0     0     0     5
```

This yields a 0% of true error rate.

In order to predict the class identities of the new observations, we use:

```
> predict(newfish.logd,newfish.test)
 [1] bream bream perch perch pike smelt smelt parki roach roach white
```

It is noted that we obtain the same results as LDA and the classification trees.

Let us summarize next the different classifications of the eleven new objects. We see that the various methods disagree on some of the class identities (e.g. check the 11^{th} object). What can we do ? One way is to ***combine the classifiers*** by assigning the object to the majority class. For example, following this scheme we would assign the first object to class bream, while the last object to class white (since 3 of 5 suggest white). Therefore, the majority assignments are as shown in the last row “True Class”.

Method	1	2	3	4	5	6	7	8	9	10	11
LDA	bream	bream	perch	perch	pike	smelt	smelt	parki	roach	roach	white
QDA	bream	bream	perch	perch	pike	smelt	smelt	parki	roach	roach	perch
Tree	bream	bream	perch	perch	pike	smelt	smelt	parki	roach	roach	white
NN-1	bream	bream	perch	white	perch	smelt	smelt	parki	perch	perch	perch
Logistic	bream	bream	perch	perch	pike	smelt	smelt	parki	roach	roach	white
TrueClass	**bream**	**bream**	**perch**	**perch**	**pike**	**smelt**	**smelt**	**parki**	**roach**	**roach**	**white**

Some Concluding Remarks :

Classification has emerged as a very important field both in statistics and in engineering (machine learning, artificial intelligence community) due to technological advances in computation and storage (easy to collect, store and process enormous datasets). Whenever attempting to "train" a classifier (i.e. build a good classification/decision rule), a lot of attention should be paid to the following issues:

- Bias-variance tradeoffs for $\hat{p}(x \mid T)$.
- The high dimensional nature of our dataset (curse of dimensionality).
- Variable selection, outliers and influential observations.
- The computational complexity of the various methods.
- Accuracy versus parsimony.