Results.pdf

Shane Zabel

```
----------------------------
Results for kyphosis dataset

Accuracy for 80/20 training/test split
0.7058824
Accuracy for 90/10 training/test split
0.8888889
----------------------------
Results for solder dataset

Accuracy for 80/20 training/test split
0.6944444
Accuracy for 90/10 training/test split
0.7777778
----------------------------
```

Example OUTPUT of Kyphosis.R

```
> # Kyphosis.R
> # Analysis Using RPart Package
> #
> # Author: Shane Zabel
> ###############################################################################
>
> #Uncomment line below if rpart is not installed
> #install.packages("rpart", dependencies=TRUE)
>
> #Install rpart
> library(rpart)
>
> #Load kyphosis data
> data(kyphosis)
>
> #Create a model using the rpart function
> K  <- rpart(kyphosis$Kyphosis ~ ., data = kyphosis, method = 'class', minsplit=2,
minbucket=1)
>
> #Create a nice decision tree plot
> par(mar = rep(0.1, 4))
> plot(K,margin=0.05)
> text(K,use.n=TRUE,cex=0.8)
> #summary(K)
>
> #List the important attributes
> K$variable.importance
      Age     Start    Number
14.770095 13.780608  3.208137
>
> #Find the best pruning paramater - CP with lowest xerror
> printcp(K)

Classification tree:
rpart(formula = kyphosis$Kyphosis ~ ., data = kyphosis, method = "class",
    minsplit = 2, minbucket = 1)

Variables actually used in tree construction:
```

```
[1] Age    Number Start

Root node error: 17/81 = 0.20988

n= 81

        CP nsplit rel error xerror    xstd
1 0.176471      0   1.00000 1.0000 0.21559
2 0.117647      1   0.82353 1.3529 0.23872
3 0.078431      2   0.70588 1.2353 0.23200
4 0.058824      5   0.47059 1.2353 0.23200
5 0.029412     10   0.17647 1.1765 0.22829
6 0.010000     16   0.00000 1.2941 0.23548
>
> #Create a pruned tree
> K1 <- prune(K,K$cptable[which.min(K$cptable[,"xerror"]),"CP"])
>
> #Create a nice decision tree plot
> #par(mar = rep(0.1, 4))
> #plot(K1,margin=0.05)
> #text(K1,use.n=TRUE,cex=0.8)
> #summary(K1)
>
> #Divide the dataset into two parts. 80% for training data and 20% for test data
> train80<-sample(nrow(kyphosis),size=64)
> trainingData80<-kyphosis[train80,]
> testData20<-kyphosis[-train80,]
>
> #Build a pruned model using just the taining data
> K80  <- rpart(trainingData80$Kyphosis ~ ., data = trainingData80, method =
'class',minsplit=2,minbucket=1)
> printcp(K80)

Classification tree:
rpart(formula = trainingData80$Kyphosis ~ ., data = trainingData80,
    method = "class", minsplit = 2, minbucket = 1)

Variables actually used in tree construction:
[1] Age    Number Start

Root node error: 12/64 = 0.1875

n= 64

        CP nsplit rel error xerror    xstd
1 0.250000      0  1.000000 1.0000 0.26021
2 0.166667      1  0.750000 1.3333 0.28868
3 0.083333      2  0.583333 1.3333 0.28868
4 0.027778      8  0.083333 1.0833 0.26822
5 0.010000     11  0.000000 1.0833 0.26822
> K80_1 <- prune(K80,K80$cptable[which.min(K80$cptable[,"xerror"]),"CP"])
>
> #Create a nice decision tree plot
> #par(mar = rep(0.1, 4))
> #plot(K80_1,margin=0.05)
> #text(K80_1,use.n=TRUE,cex=0.8)
> #summary(K80_1)
>
> #Find the prediction on the test data
> out <- predict(K80_1,newdata=testData20,type="vector")
> #Print out results
> out
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
> testData20$Kyphosis
 [1] absent  absent  absent  absent  absent  absent  absent  present present
[10] absent  present absent  absent  absent  absent  present present
Levels: absent present
> 
> #Calculate the accuracy
> dataNum=array(NA,c(1,dim(testData20)[1]))
> for(i in 1:dim(testData20)[1]){
+     if(testData20$Kyphosis[i]=="absent"){
+           dataNum[i] <- 1
+     }else{
+           dataNum[i] <- 2
+     }
+ }
> result<-dataNum-out
> counter<-0
> for(i in 1:dim(testData20)[1]){
+     counter<-counter+abs(result[i])
+ }
> accuracy80<-1-counter/dim(testData20)[1]
> #Print out the accuracy
> accuracy80
[1] 0.7058824
> 
> #Divide the dataset into two parts. 90% for training data and 10% for test data
> train90<-sample(nrow(kyphosis),size=72)
> trainingData90<-kyphosis[train90,]
> testData10<-kyphosis[-train90,]
> 
> #Build a pruned model using just the taining data
> K90  <- rpart(trainingData90$Kyphosis ~ ., data = trainingData90, method =
'class',minsplit=2,minbucket=1)
> printcp(K90)

Classification tree:
rpart(formula = trainingData90$Kyphosis ~ ., data = trainingData90, 
    method = "class", minsplit = 2, minbucket = 1)

Variables actually used in tree construction:
[1] Age    Number Start 

Root node error: 16/72 = 0.22222

n= 72 

        CP nsplit rel error xerror    xstd
1 0.125000      0    1.0000 1.0000 0.22048
2 0.093750      2    0.7500 1.4375 0.24727
3 0.062500      4    0.5625 1.4375 0.24727
4 0.046875      6    0.4375 1.3125 0.24105
5 0.031250     10    0.2500 1.3125 0.24105
6 0.020833     14    0.1250 1.3125 0.24105
7 0.010000     19    0.0000 1.3125 0.24105
> K90_1 <- prune(K90,K90$cptable[which.min(K90$cptable[,"xerror"]),"CP"])
> 
```

```
> #Create a nice decision tree plot
> #par(mar = rep(0.1, 4))
> #plot(K90_1,margin=0.05)
> #text(K90_1,use.n=TRUE,cex=0.8)
> #summary(K90_1)
>
> #Find the prediction on the test data
> out2 <- predict(K90_1,newdata=testData10,type="vector")
> #Print out results
> out2
[1] 1 1 1 1 1 1 1 1 1
> testData10$Kyphosis
[1] absent  absent  absent  present absent  absent  absent  absent  absent
Levels: absent present
>
> #Calculate the accuracy
> dataNum2=array(NA,c(1,dim(testData10)[1]))
> for(i in 1:dim(testData10)[1]){
+     if(testData10$Kyphosis[i]=="absent"){
+           dataNum2[i] <- 1
+     }else{
+           dataNum2[i] <- 2
+     }
+ }
> result2<-dataNum2-out2
> counter2<-0
> for(i in 1:dim(testData10)[1]){
+     counter2<-counter2+abs(result2[i])
+ }
> accuracy90<-1-counter2/dim(testData10)[1]
> #Print out accuracy
> accuracy90
[1] 0.8888889
```

Example OUTPUT of Solder.R

```
> # Solder.R
> # Analysis Using RPart Package
> #
> # Author: Shane Zabel
> ###############################################################################
>
> #Uncomment line below if rpart is not installed
> #install.packages("rpart", dependencies=TRUE)
>
> #Install rpart
> library(rpart)
>
> #Load kyphosis data
> data(solder)
>
> #Build a model using the rpart function
> K  <- rpart(solder$Solder ~ ., data = solder, method = 'class', minsplit=2,
minbucket=1)
>
> #Create a nice decision tree plot
> par(mar = rep(0.1, 4))
> plot(K,margin=0.05)
> text(K,use.n=TRUE,cex=0.8)
> #summary(K)
>
> #List the important attributes
> K$variable.importance
    skips   Opening      Mask   PadType     Panel
76.726522 36.712848 32.892830  8.638895  1.041505
>
> #Find the best pruning paramater - CP with lowest xerror
> printcp(K)

Classification tree:
rpart(formula = solder$Solder ~ ., data = solder, method = "class",
    minsplit = 2, minbucket = 1)

Variables actually used in tree construction:
[1] Mask    Opening skips

Root node error: 360/720 = 0.5

n= 720

        CP nsplit rel error  xerror     xstd
1 0.338889      0   1.00000 1.07778 0.037155
2 0.030556      1   0.66111 0.66111 0.035063
3 0.025926      4   0.56944 0.63611 0.034713
4 0.010000      7   0.49167 0.54167 0.033123
>
> #Create a pruned tree
> K1 <- prune(K,K$cptable[which.min(K$cptable[,"xerror"]),"CP"])
>
> #Create a nice decision tree plot
> #par(mar = rep(0.1, 4))
> #plot(K1,margin=0.05)
> #text(K1,use.n=TRUE,cex=0.8)
> #summary(K1)
>
> #Divide the dataset into two parts. 80% for training data and 20% for test data
> train80<-sample(nrow(solder),size=576)
> trainingData80<-solder[train80,]
> testData20<-solder[-train80,]
>
> #Build a pruned model using just the taining data
> K80  <- rpart(trainingData80$Solder ~ ., data = trainingData80, method =
'class',minsplit=2,minbucket=1)
> printcp(K80)

Classification tree:
rpart(formula = trainingData80$Solder ~ ., data = trainingData80,
    method = "class", minsplit = 2, minbucket = 1)

Variables actually used in tree construction:
[1] Mask    Opening PadType skips

Root node error: 283/576 = 0.49132

n= 576
```

```
        CP nsplit rel error  xerror     xstd
1 0.349823      0   1.00000 1.03887 0.042394
2 0.031802      1   0.65018 0.65018 0.039542
3 0.017668      4   0.54417 0.59011 0.038479
4 0.014134      8   0.47350 0.60424 0.038746
5 0.012367     11   0.43110 0.58304 0.038341
6 0.010000     13   0.40636 0.57244 0.038129
> K80_1 <- prune(K80,K80$cptable[which.min(K80$cptable[,"xerror"]),"CP"])
>
> #Create a nice decision tree plot
> #par(mar = rep(0.1, 4))
> #plot(K80_1,margin=0.05)
> #text(K80_1,use.n=TRUE,cex=0.8)
> #summary(K80_1)
>
> #Find the prediction on the test data
> out <- predict(K80_1,newdata=testData20,type="vector")
> out
  1   2   4  12  14  15  21  22  38  42  59  60  62  63  65  78  82  83  94 104
  1   1   1   1   1   1   1   1   2   1   1   2   1   1   1   1   1   1   2   1
107 109 112 120 125 135 143 147 152 165 168 172 216 222 224 225 226 230 238 249
  1   1   2   2   1   2   1   1   2   2   2   1   1   1   2   1   1   2   1   1
253 258 266 268 269 271 278 279 281 294 296 297 310 316 323 325 326 328 329 331
  1   1   1   1   1   2   2   1   1   1   1   1   2   2   1   1   1   1   2   2
334 335 336 338 343 347 350 351 352 361 362 372 374 380 382 385 396 397 399 406
  2   2   2   2   2   2   2   2   2   1   1   2   2   2   2   1   1   1   1   2
408 410 414 416 426 430 431 435 437 438 443 447 450 451 454 456 466 472 481 494
  2   2   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   2
495 501 507 514 529 535 538 544 549 568 569 576 577 583 584 586 587 592 596 598
  2   2   1   2   2   1   2   1   1   1   1   1   2   2   1   2   2   2   1   1
601 602 605 607 609 612 626 628 630 635 639 640 641 643 644 646 653 667 675 679
  1   1   2   2   2   1   1   2   1   2   2   2   2   2   2   2   2   2   2   2
682 686 706 712
  2   2   2   1
> testData20$Solder
  [1] Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick
 [13] Thick Thick Thick Thick Thick Thick Thin  Thin  Thin  Thin  Thin  Thin
 [25] Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thick Thick Thick Thick
 [37] Thick Thick Thick Thick Thick Thick Thick Thick Thick Thin  Thin  Thin
 [49] Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin
 [61] Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thick Thick Thick
 [73] Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick
 [85] Thick Thick Thick Thick Thick Thick Thick Thick Thick Thin  Thin  Thin
 [97] Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thick
[109] Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick
[121] Thick Thick Thick Thick Thick Thick Thick Thick Thick Thin  Thin  Thin
[133] Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin
Levels: Thick Thin
>
> #Calculate the accuracy
> dataNum=array(NA,c(1,dim(testData20)[1]))
> for(i in 1:dim(testData20)[1]){
+     if(testData20$Solder[i]=="Thick"){
+           dataNum[i] <- 1
+     }else{
+           dataNum[i] <- 2
+     }
+ }
```

```
> result<-dataNum-out
> counter<-0
> for(i in 1:dim(testData20)[1]){
+     counter<-counter+abs(result[i])
+ }
> accuracy80<-1-counter/dim(testData20)[1]
> accuracy80
[1] 0.6944444
>
> #Divide the dataset into two parts. 90% for training data and 10% for test data
> train90<-sample(nrow(solder),size=648)
> trainingData90<-solder[train90,]
> testData10<-solder[-train90,]
>
> #Build a pruned model using just the taining data
> K90  <- rpart(trainingData90$Solder ~ ., data = trainingData90, method =
'class',minsplit=2,minbucket=1)
> printcp(K90)

Classification tree:
rpart(formula = trainingData90$Solder ~ ., data = trainingData90,
    method = "class", minsplit = 2, minbucket = 1)

Variables actually used in tree construction:
[1] Mask    Opening PadType skips

Root node error: 319/648 = 0.49228

n= 648

        CP nsplit rel error  xerror     xstd
1 0.338558      0   1.00000 1.04075 0.039887
2 0.028213      1   0.66144 0.66144 0.037394
3 0.024033      4   0.57680 0.63009 0.036913
4 0.010449      7   0.50470 0.53605 0.035171
5 0.010000     10   0.47335 0.56113 0.035681
> K90_1 <- prune(K90,K90$cptable[which.min(K90$cptable[,"xerror"]),"CP"])
>
> #Create a nice decision tree plot
> #par(mar = rep(0.1, 4))
> #plot(K90_1,margin=0.05)
> #text(K90_1,use.n=TRUE,cex=0.8)
> #summary(K90_1)
>
> #Find the prediction on the test data
> out2 <- predict(K90_1,newdata=testData10,type="vector")
> out2
  3   5   7   8  12  27  32  33  51  53  63  66  75  93  99 110 111 112 118 120
  1   1   1   1   1   1   1   1   1   1   1   1   1   2   2   2   2   2   1   2
121 125 140 166 171 175 198 212 213 239 240 256 258 263 274 281 303 306 315 331
  1   1   1   2   2   2   1   2   1   1   2   1   1   2   2   1   2   2   2   2
338 342 344 347 355 384 404 429 447 457 461 462 477 478 493 500 529 557 558 581
  2   2   2   2   1   2   2   2   1   2   2   2   2   1   1   2   2   1   1   1
589 602 630 632 646 648 651 684 696 707 714 715
  2   1   1   2   2   2   2   2   2   2   2   1
> testData10$Solder
 [1] Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick Thick
[13] Thick Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin
[25] Thin  Thin  Thick Thick Thick Thick Thick Thick Thick Thick Thin  Thin
```

```
[37] Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thick Thick Thick
[49] Thick Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thick Thick Thick
[61] Thick Thick Thick Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin  Thin
Levels: Thick Thin
>
> #Calculate the accuracy
> dataNum2=array(NA,c(1,dim(testData10)[1]))
> for(i in 1:dim(testData10)[1]){
+     if(testData10$Solder[i]=="Thick"){
+           dataNum2[i] <- 1
+     }else{
+           dataNum2[i] <- 2
+     }
+ }
> result2<-dataNum2-out2
> counter2<-0
> for(i in 1:dim(testData10)[1]){
+     counter2<-counter2+abs(result2[i])
+ }
> accuracy90<-1-counter2/dim(testData10)[1]
> accuracy90
[1] 0.7777778
```