Capstone Project Logistic Regression

Tom Thorpe

August 15, 2018

Tree Coverage Logistic Regression Summary

The results of the logistic regression for each of the tree types and initial and insignificant features removed was saved to a csv file. The logistic summary is read and some data rounded to the nearest percent.

```
LogisticSummary=read.csv("ForestCoverLogisticStats.csv")

LogisticSummary$Description<-as.character(LogisticSummary$Description)
LogisticSummary$Selected<-as.character(LogisticSummary$Selected)

# Add a row for the weighted average calculations
LogisticSummary[nrow(LogisticSummary)+1,]<- list("Weighted Average",NA,NA,NA,
                                            0,0,0,0,0,0,0,0,0,0,0,0,"")
```

```
lastRow=nrow(LogisticSummary)
testCount=LogisticSummary$Count[[1]]
LogisticSummary$Count[[lastRow]] = testCount

# Calculate weighted average for accuracy, sensitivity and specificity
for (i in 1:(lastRow-1)) {

  # "X" in selected indicates the Logistic model for the current tree type
  # The model was selected manually by updating the CSV file.
  if(LogisticSummary$Selected[[i]]=="X") {
    curNum = LogisticSummary$TP[[i]]+LogisticSummary$FN[[i]]

    LogisticSummary$Accuracy[[lastRow]] = LogisticSummary$Accuracy[[lastRow]] +
      LogisticSummary$Accuracy[[i]]*(curNum/testCount)

    LogisticSummary$Sensitivity[[lastRow]] = LogisticSummary$Sensitivity[[lastRow]] +
      LogisticSummary$Sensitivity[[i]]*(curNum/testCount)

    LogisticSummary$Specificity[[lastRow]] = LogisticSummary$Specificity[[lastRow]] +
      LogisticSummary$Specificity[[i]]*(curNum/testCount)
  }
}
```

```
# Create a copy of the summary and drop columns not being displayed
LogSummaryTable=LogisticSummary

LogSummaryTable$TrueLabel=NULL
LogSummaryTable$FalseLabel=NULL
LogSummaryTable$BaselineLabel=NULL
LogSummaryTable$AIC=NULL
#LogSummaryTable$TP=NULL
LogSummaryTable$TN=NULL
LogSummaryTable$FP=NULL
LogSummaryTable$FN=NULL
```

```r
LogSummaryTable$Count=NULL

# Abbreviate columns names so each row is on one line when being displayed
colnames(LogSummaryTable)<- c("Description", "BaseAcc", "Acc", "Sens", "Spec", "AUC",
                              "Num", "Thresh","Select")

LogSummaryTable$Num = LogSummaryTable$Num + LogisticSummary$FN # calculate number of trees
LogSummaryTable$Num[[lastRow]] = testCount # Set the number in the weighted average row to number of te

# Round data to nearest percent
LogSummaryTable$BaseAcc = paste(as.integer(LogSummaryTable$BaseAcc*100),"%",sep="")
LogSummaryTable$Acc = paste(as.integer(LogSummaryTable$Acc*100),"%",sep="")
LogSummaryTable$Sens = paste(as.integer(LogSummaryTable$Sens*100),"%",sep="")
LogSummaryTable$Spec = paste(as.integer(LogSummaryTable$Spec*100),"%",sep="")
LogSummaryTable$AUC = paste(as.integer(LogSummaryTable$AUC),"%",sep="")

LogSummaryTable$BaseAcc[[lastRow]]=""
LogSummaryTable$AUC[[lastRow]]=""
#LogSummaryTable$Thresh[[lastRow]]=""

# Display the summary table
LogSummaryTable
```

```
##              Description BaseAcc Acc Sens Spec AUC    Num Thresh Select
## 1          Aspen All Agg     98% 68%  86%  68% 82%   2848  0.015
## 2          Aspen All Ind     98% 69%  88%  69% 83%   2848  0.014
## 3          Aspen Sig Agg     98% 57%  85%  56% 79%   2848  0.012
## 4          Aspen Sig Ind     98% 68%  93%  68% 87%   2848  0.011      X
## 5  Cotton/Willow All Agg     99% 96%  97%  96% 99%    824  0.006
## 6  Cotton/Willow All Ind     99% 96%  97%  96% 99%    824  0.006
## 7  Cotton/Willow Sig Agg     99% 95%  94%  95% 98%    824  0.008      X
## 8  Cotton/Willow Sig Ind     99% 95%  93%  95% 98%    824  0.008
## 9    Douglas Fir All Agg     97% 89%  95%  88% 96%   5210  0.035
## 10   Douglas Fir All Ind     97% 89%  95%  88% 96%   5210  0.035
## 11   Douglas Fir Sig Agg     97% 87%  97%  86% 95%   5210  0.033      X
## 12   Douglas Fir Sig Ind     97% 87%  94%  87% 95%   5210  0.032
## 13     Krummholz All Agg     96% 91%  93%  91% 98%   6153  0.035
## 14     Krummholz All Ind     96% 91%  93%  91% 98%   6153  0.034
## 15     Krummholz Sig Agg     96% 90%  95%  89% 97%   6153  0.029      X
## 16     Krummholz Sig Ind     96% 86%  96%  86% 96%   6153  0.030
## 17     Lodgepole All Agg     51% 75%  79%  71% 82%  84990  0.476
## 18     Lodgepole All Ind     51% 75%  78%  72% 82%  84990  0.481
## 19     Lodgepole Sig Agg     51% 75%  79%  72% 82%  84990  0.482      X
## 20     Lodgepole Sig Ind     51% 69%  91%  49% 80%  84990  0.345
## 21     Ponderosa All Agg     93% 93%  98%  93% 98%  10726  0.099
## 22     Ponderosa All Ind     93% 93%  98%  93% 98%  10726  0.099
## 23     Ponderosa Sig Agg     93% 92%  97%  91% 97%  10726  0.082
## 24     Ponderosa Sig Ind     93% 92%  97%  92% 98%  10726  0.068      X
## 25    Spruce/Fir All Agg     63% 73%  87%  65% 84%  63552  0.297
## 26    Spruce/Fir All Ind     63% 73%  87%  65% 84%  63552  0.297
## 27    Spruce/Fir Sig Agg     63% 73%  87%  66% 83%  63552  0.307      X
## 28    Spruce/Fir Sig Ind     63% 73%  87%  65% 84%  63552  0.298
## 29      Weighted Average         76%  84%  72%     174303  0.000
```

The Logistic Model selected used models that only kept significant variables since all the models using all the feature data had several coefficients that were in the millions or billions.

The aggregated vs individualized model chosen was based on the best sensitivity and specificity.

The individuated data only provided better results in two of the seven tree types.

ROC of Selected Models

Response Operating Characteristics are shown for the selected models.

The Aspen ROC is irregularly shaped and starts at 0.3

The Lodgepole and Spruce/Fir, which represent over 84% of the population have the worst response curves. This is going to limit the accuracy of the overall predictions since they will carry more weight in the performance results.

Logistic Regression Model Summary

All of the Logistic regression models were saved. Look at Logistic Model for Ponderosa

```r
load("Ponder_Ind_All_LogMod.Rdata")

summary(Ponder_Ind_All_LogMod)
```

```
##
## Call:
## glm(formula = PonderosaPine ~ Elev + Aspect + Slope + H2OHD +
##     H2OVD + RoadHD + FirePtHD + Shade9AM + Shade12PM + Shade3PM +
##     RWwild + NEwild + CMwild + CPwild + Montane_low + Montane +
##     SubAlpine + Alpine + Dry + Non_Dry + Alluvium + Glacial +
##     Sed_mix + Ign_Meta + Aquolis_cmplx + Argiborolis_Pachic +
##     Borohemists_cmplx + Bross + Bullwark + Bullwark_Cmplx + Catamount +
##     Catamount_cmplx + Cathedral + Como + Cryaquepts_cmplx + Cryaquepts_Typic +
##     Cryaquolls + Cryaquolls_cmplx + Cryaquolls_Typic + Cryaquolls_Typic_cmplx +
##     Cryoborolis_cmplx + Cryorthents + Cryorthents_cmplx + Cryumbrepts +
##     Cryumbrepts_cmplx + Gateview + Gothic + Granile + Haploborolis +
##     Legault + Legault_cmplx + Leighcan + Leighcan_cmplx + Leighcan_warm +
##     Moran + Ratake + Ratake_cmplx + Rogert + Supervisor_Limber_cmplx +
##     Troutville + Unspecified + Vanet + Wetmore + Bouldery_ext +
##     Rock_Land + Rock_Land_cmplx + Rock_Outcrop + Rock_Outcrop_cmplx +
##     Rubbly + Stony + Stony_extreme + Stony_very + Till_Substratum,
##     family = binomial, data = forestTrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8068  -0.0113   0.0000   0.0000   3.7911
##
## Coefficients: (17 not defined because of singularities)
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             1.393e+09  7.098e+11   0.002    0.998
## Elev                   -5.560e-03  1.016e-04 -54.732  < 2e-16 ***
## Aspect                  1.509e-03  1.385e-04  10.894  < 2e-16 ***
## Slope                  -5.128e-03  3.981e-03  -1.288    0.198
## H2OHD                   1.927e-03  9.707e-05  19.847  < 2e-16 ***
```

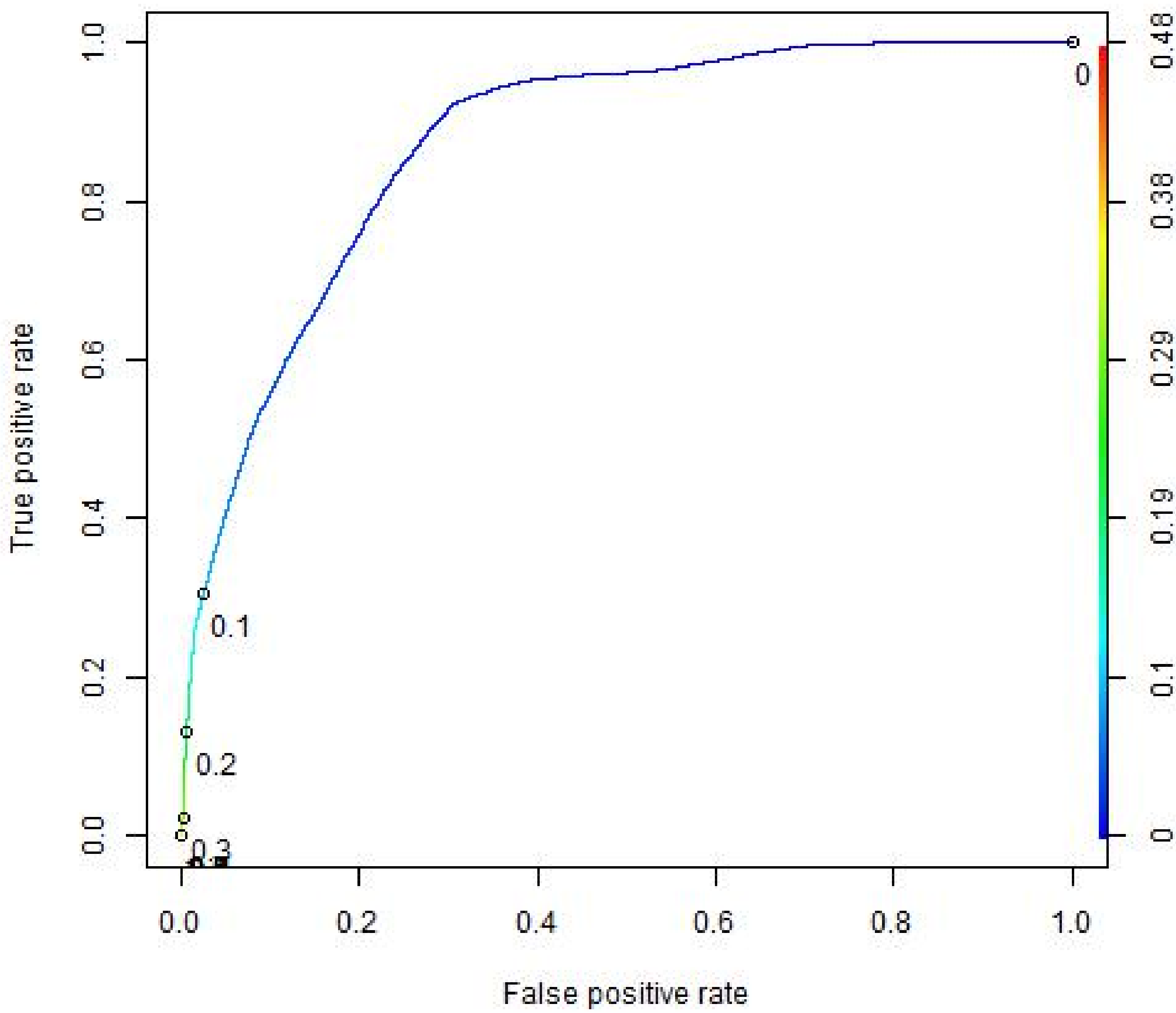

Figure 1: Aspen ROC for Significant Individuated Data

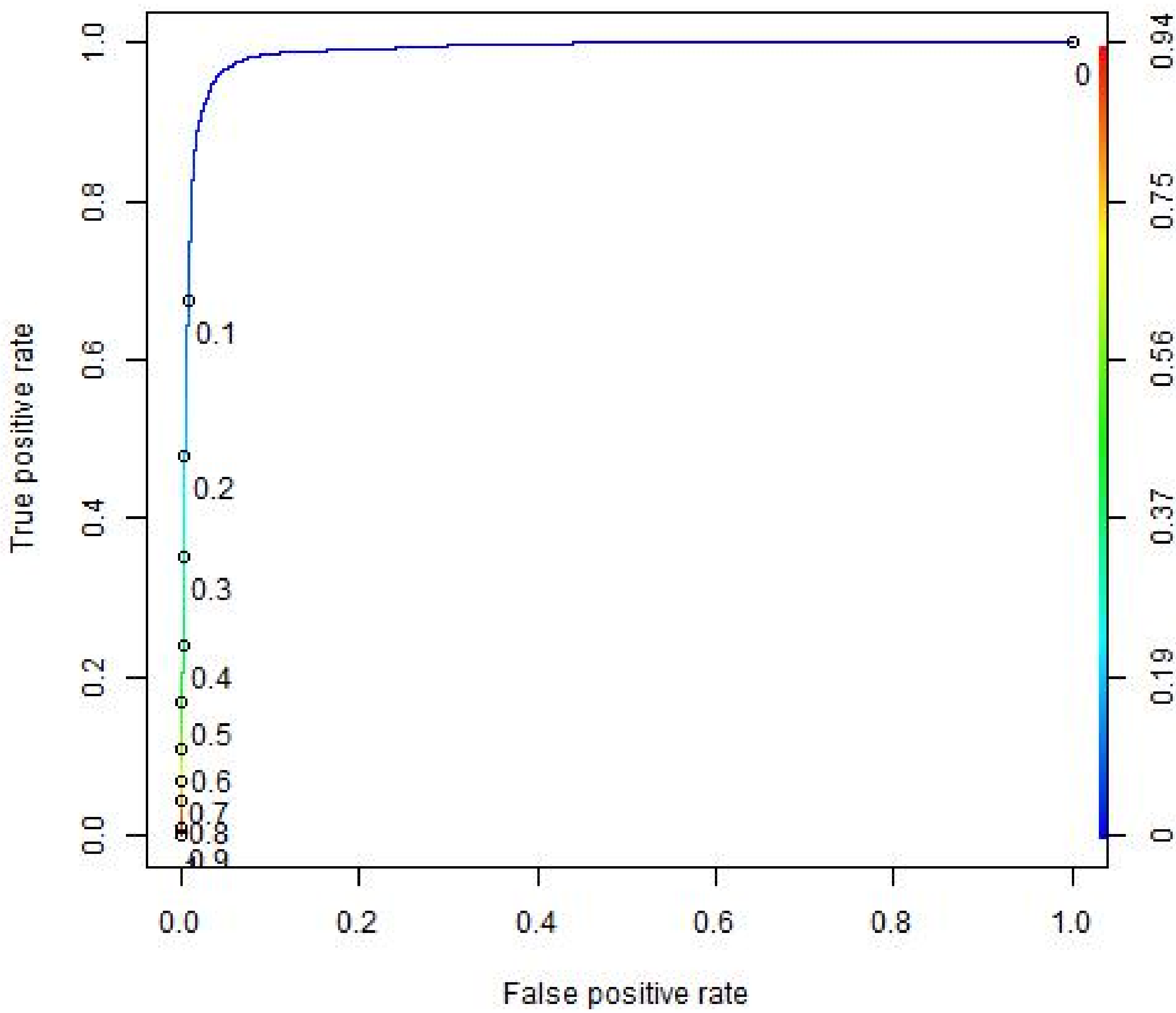

Figure 2: Cottonwood Willow ROC for Significant Aggregated Data

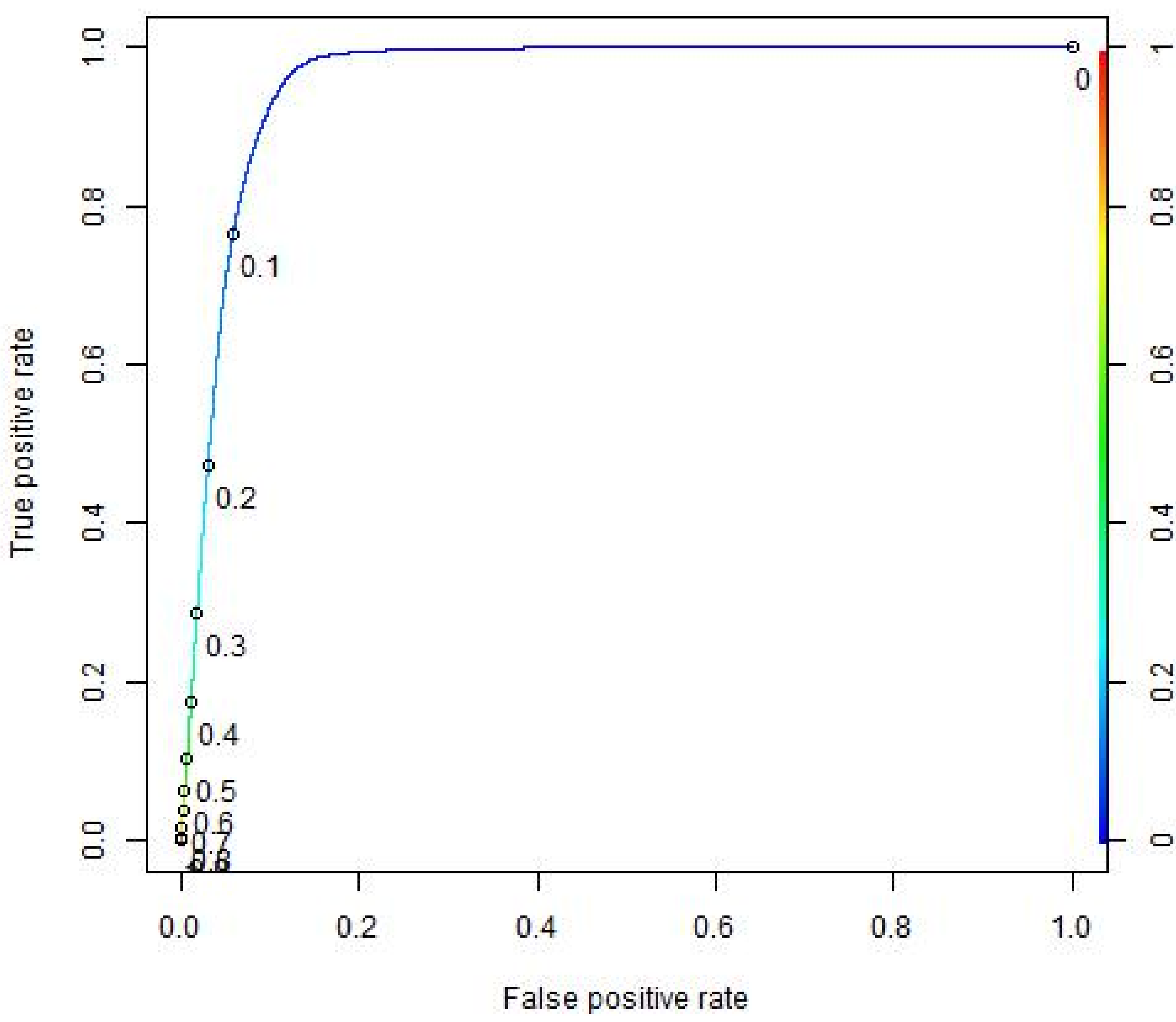

Figure 3: Douglas Fir ROC for Significant Aggregated Data

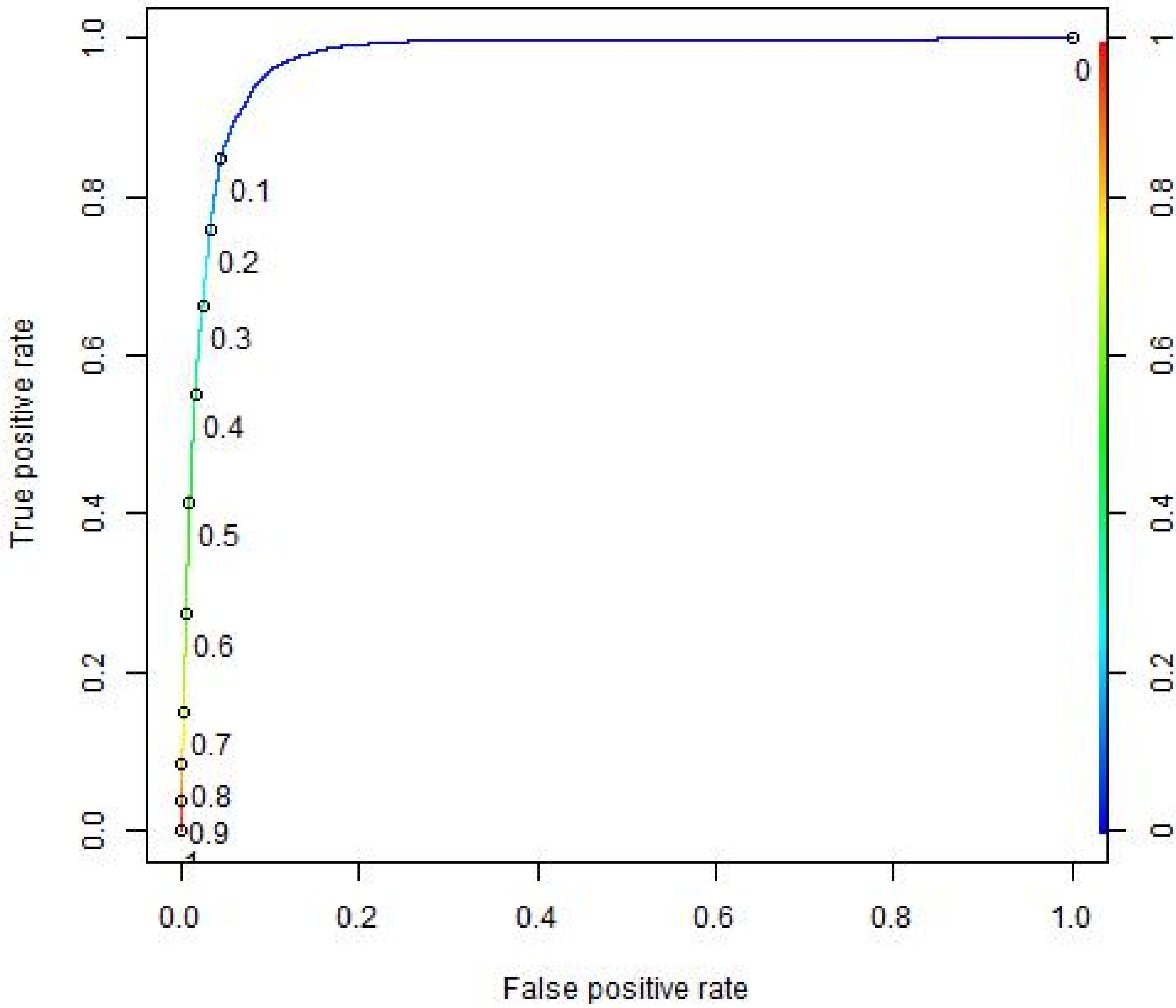

Figure 4: Krummholz ROC for Significant Aggregated Data

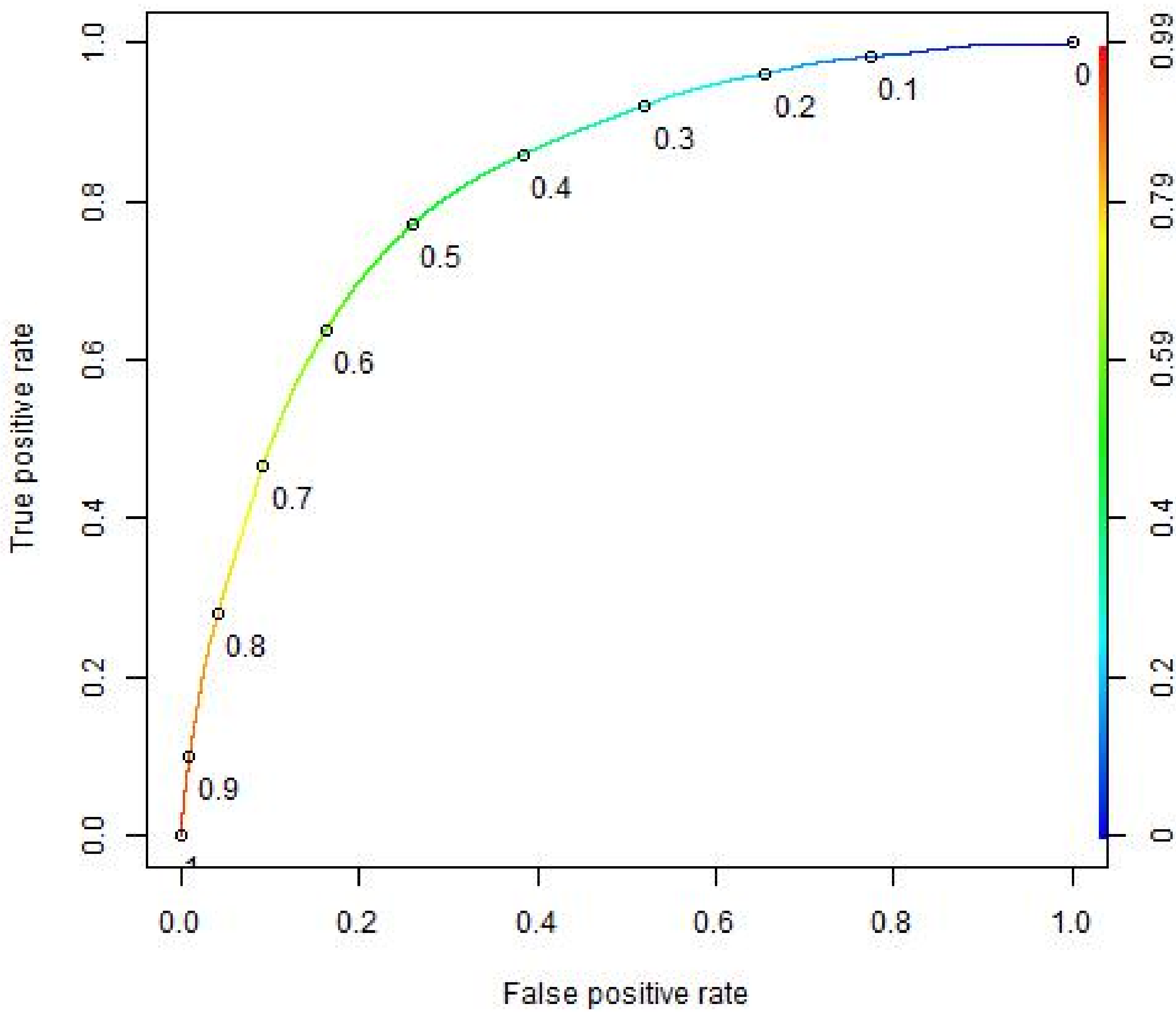

Figure 5: Lodgepole Pine ROC for Significant Aggregated Data

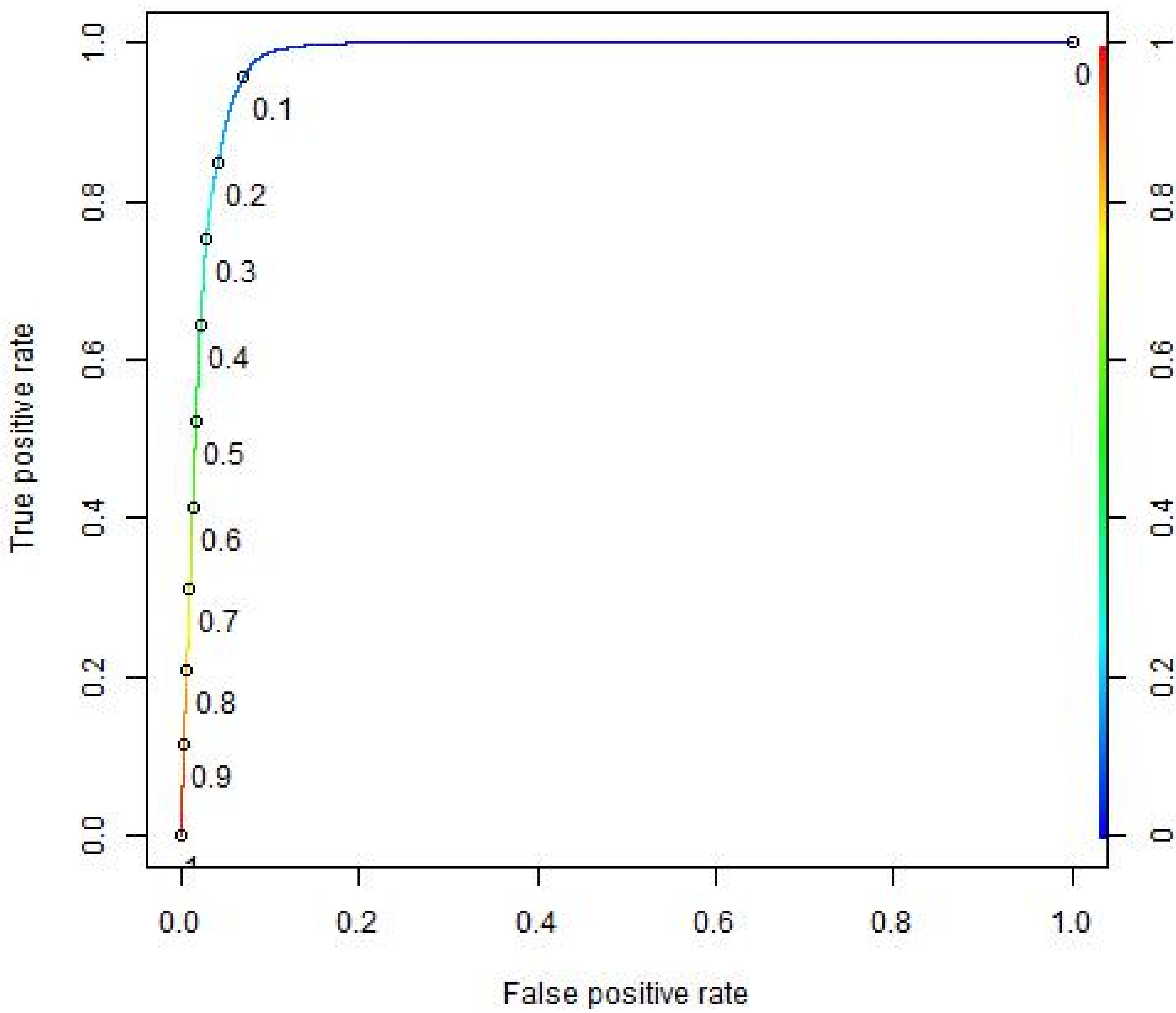

Figure 6: Ponderosa Pine ROC for Significant Individuated Data

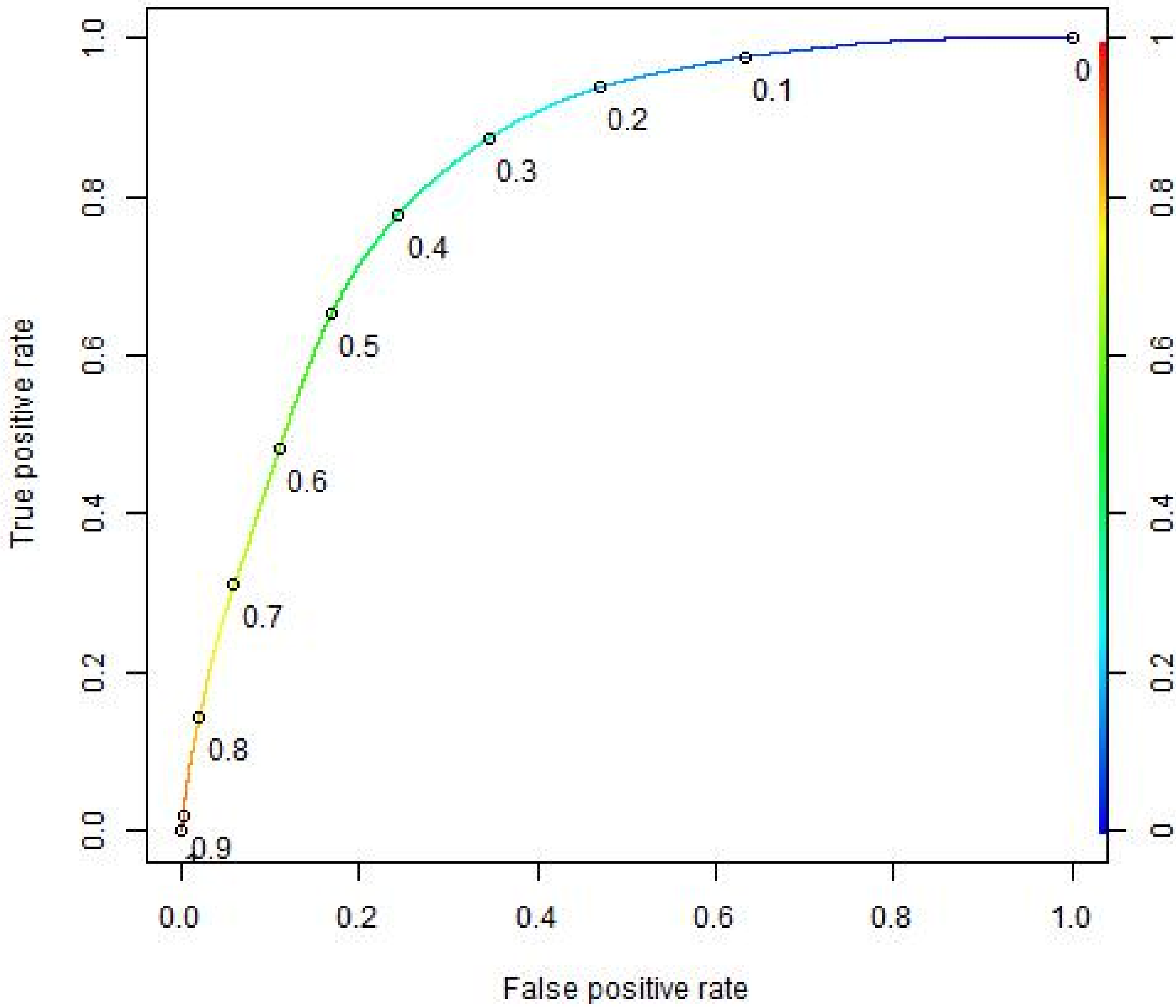

Figure 7: Spruce and Fir ROC for All Aggregated Data

```
## H2OVD                    1.878e-03  2.480e-04   7.574 3.63e-14 ***
## RoadHD                  -1.274e-04  1.798e-05  -7.084 1.40e-12 ***
## FirePtHD                -3.017e-04  2.082e-05 -14.492  < 2e-16 ***
## Shade9AM                -4.100e-02  3.687e-03 -11.120  < 2e-16 ***
## Shade12PM                4.647e-02  3.237e-03  14.356  < 2e-16 ***
## Shade3PM                -4.002e-02  3.152e-03 -12.697  < 2e-16 ***
## RWwild                  -1.453e+01  1.550e+02  -0.094    0.925
## NEwild                  -1.248e+01  2.671e+02  -0.047    0.963
## CMwild                   3.785e-01  3.878e-02   9.760  < 2e-16 ***
## CPwild                          NA         NA      NA       NA
## Montane_low             -2.508e+09  1.105e+12  -0.002    0.998
## Montane                  3.548e+09  5.683e+11   0.006    0.995
## SubAlpine               -1.393e+09  7.098e+11  -0.002    0.998
## Alpine                  -1.393e+09  7.098e+11  -0.002    0.998
## Dry                      4.009e+10  1.558e+13   0.003    0.998
## Non_Dry                  1.115e+09  5.444e+11   0.002    0.998
## Alluvium                -4.663e+09  8.035e+11  -0.006    0.995
## Glacial                 -7.224e+09  7.509e+12  -0.001    0.999
## Sed_mix                 -4.503e+10  1.627e+13  -0.003    0.998
## Ign_Meta                        NA         NA      NA       NA
## Aquolis_cmplx           -4.148e+10  1.609e+13  -0.003    0.998
## Argiborolis_Pachic              NA         NA      NA       NA
## Borohemists_cmplx       -3.224e+00  2.174e+03  -0.001    0.999
## Bross                   -7.726e+00  5.472e+03  -0.001    0.999
## Bullwark                -6.056e+09  1.278e+12  -0.005    0.996
## Bullwark_Cmplx          -6.056e+09  1.278e+12  -0.005    0.996
## Catamount                1.644e+01  3.038e+03   0.005    0.996
## Catamount_cmplx         -4.107e-01  4.941e+02  -0.001    0.999
## Cathedral                4.116e-01  9.010e-02   4.568 4.93e-06 ***
## Como                     1.012e+01  7.975e+02   0.013    0.990
## Cryaquepts_cmplx        -6.083e+00  2.159e+03  -0.003    0.998
## Cryaquepts_Typic        -2.561e+09  7.416e+12   0.000    1.000
## Cryaquolls              -1.792e+00  1.565e+03  -0.001    0.999
## Cryaquolls_cmplx        -1.944e+00  1.565e+03  -0.001    0.999
## Cryaquolls_Typic         4.663e+09  8.035e+11   0.006    0.995
## Cryaquolls_Typic_cmplx   7.224e+09  7.509e+12   0.001    0.999
## Cryoborolis_cmplx               NA         NA      NA       NA
## Cryorthents            -2.155e+00  3.403e+03  -0.001    0.999
## Cryorthents_cmplx       -5.399e+00  3.447e+03  -0.002    0.999
## Cryumbrepts                     NA         NA      NA       NA
## Cryumbrepts_cmplx               NA         NA      NA       NA
## Gateview                        NA         NA      NA       NA
## Gothic                   7.398e-02  7.113e+03   0.000    1.000
## Granile                 -5.007e+00  1.331e+03  -0.004    0.997
## Haploborolis             5.281e-01  8.636e-02   6.115 9.67e-10 ***
## Legault                 -6.056e+09  1.278e+12  -0.005    0.996
## Legault_cmplx                   NA         NA      NA       NA
## Leighcan                -4.556e+00  6.770e+02  -0.007    0.995
## Leighcan_cmplx          -4.920e+00  3.133e+03  -0.002    0.999
## Leighcan_warm           -6.657e-01  3.374e+03   0.000    1.000
## Moran                           NA         NA      NA       NA
## Ratake                   2.266e+00  8.352e-02  27.129  < 2e-16 ***
## Ratake_cmplx            -1.352e+00  3.059e+03   0.000    1.000
## Rogert                  -4.663e+09  8.035e+11  -0.006    0.995
```

```
## Supervisor_Limber_cmplx          NA          NA      NA       NA
## Troutville              1.168e+09  7.436e+12   0.000    1.000
## Unspecified            -4.148e+10  1.609e+13  -0.003    0.998
## Vanet                          NA          NA      NA       NA
## Wetmore                 1.546e+00  8.346e-02  18.527  < 2e-16 ***
## Bouldery_ext            7.224e+09  7.509e+12   0.001    0.999
## Rock_Land              -7.676e-01  3.491e+02  -0.002    0.998
## Rock_Land_cmplx        -3.776e+00  3.059e+03  -0.001    0.999
## Rock_Outcrop                   NA          NA      NA       NA
## Rock_Outcrop_cmplx     -3.557e+00  3.059e+03  -0.001    0.999
## Rubbly                         NA          NA      NA       NA
## Stony                          NA          NA      NA       NA
## Stony_extreme                  NA          NA      NA       NA
## Stony_very                     NA          NA      NA       NA
## Till_Substratum                NA          NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 188044  on 406708  degrees of freedom
## Residual deviance:  61743  on 406652  degrees of freedom
## AIC: 61857
##
## Number of Fisher Scoring iterations: 21
```

Load Data

Load the tree coverage data set and split into training and testing sets so they match the training and testing sets used for creating the logistic regression models.

Add columns to calculate response probabilities for each logistic regression model.

```
firstTime=TRUE
set.seed(127)
library(caTools) # needed for split function
library(dplyr) # needed for mutate function
```

```
##
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
if (firstTime) {
  infile="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcover_clean_full.csv"
 #infile="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestcoversmall_clean_full.csv"

  forestcover <- read.csv(infile,header=TRUE,sep=",")

  # Add columns for probabilities of each tree cover type
```

```
  forestcover <- mutate(forestcover, AspenProb=0.0)
  forestcover <- mutate(forestcover, CotWilProb=0.0)
  forestcover <- mutate(forestcover, DougFirProb=0.0)
  forestcover <- mutate(forestcover, KrummProb=0.0)
  forestcover <- mutate(forestcover, LodgeProb=0.0)
  forestcover <- mutate(forestcover, PonderProb=0.0)
  forestcover <- mutate(forestcover, SprFirProb=0.0)

  # Add column to store the tree type predicted by the model.
  # It will be compared to the CovName column to construct the confusion matrix.
  forestcover <- mutate(forestcover, EstTreeType="X")
  forestcover$EstTreeType <- as.character(forestcover$EstTreeType)
}
```

Calc Probabilities using preferred Models

```
if (firstTime) {
  # Calculate probabilities for each tree type based on appropriate logistic model
  load("Aspen_Ind_Sig_LogMod.Rdata")
  forestcover$AspenProb=predict(Aspen_Ind_Sig_LogMod, type="response",newdata=forestcover)

  load("CotWil_Agg_Sig_LogMod.Rdata")
  forestcover$CotWilProb=predict(CotWil_Agg_Sig_LogMod, type="response",newdata=forestcover)

  load("DougFir_Agg_Sig_LogMod.Rdata")
  forestcover$DougFirProb=predict(DougFir_Agg_Sig_LogMod, type="response",newdata=forestcover)

  load("Krumm_Agg_Sig_LogMod.Rdata")
  forestcover$KrummProb=predict(Krumm_Agg_Sig_LogMod, type="response",newdata=forestcover)

  load("Lodge_Agg_Sig_LogMod.Rdata")
  forestcover$LodgeProb=predict(Lodge_Agg_Sig_LogMod, type="response",newdata=forestcover)

  load("Ponder_Ind_Sig_LogMod.Rdata")
  forestcover$PonderProb=predict(Ponder_Ind_Sig_LogMod, type="response",newdata=forestcover)

  load("SprFir_Agg_Sig_LogMod.Rdata")
  forestcover$SprFirProb=predict(SprFir_Agg_Sig_LogMod, type="response",newdata=forestcover)
}
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

Create Training and Testing Data

```
if (firstTime) {
  # Create training and testing data
  split = sample.split(forestcover$CovType, 0.70) # we want 65% in the training set
  forestTrain = subset(forestcover, split == TRUE)
  forestTest  = subset(forestcover, split == FALSE)
```

```
  # Save the training file with probabilities for later use
  out1file="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestTrainProbs.csv"
  write.csv(forestTrain, file=out1file,row.names=FALSE)

  # Save the testing file with probabilities for later use
  out2file="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestTestProbs.csv"
  write.csv(forestTest, file=out2file,row.names=FALSE)
}
```

Load Training and Test Sets with Probabilities Already Calculated

```
if (!firstTime) {
  in1file="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestTrainProbs.csv"
  forestTrain <- read.csv(in1file,header=TRUE,sep=",")

  in2file="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestTestProbs.csv"
  forestTest <- read.csv(in2file,header=TRUE,sep=",")

  forestTrain$EstTreeType <- as.character(forestTrain$EstTreeType)
  forestTest$EstTreeType <- as.character(forestTest$EstTreeType)
}

#str(forestTest, list.len = ncol(forestTest))
```

Helper functions

Create helper functions to calculate tree types, model stats and search for optimum model thresholds.

Find Model Thresholds Helper Function

The thresholds that were found for the individual logistic regression runs are not the optimum when combining the models. A function to find the optimum thresholds on the training data is shown next.

Each threshold is varied from 0.0 to 1.0 in 0.1 increments finding the the threshold that maximizes the squared sums of sensitivity and specificity for all seven logistic models combined.. The threshold maximizing the sensitivity/specificity combination is further refined in 0.01 increments.

```
treeLabels=c("Aspen", "CotWill", "DougFir", "Krumm", "Lodge", "Ponder", "SpruceFir")
zeros=c(0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0)
zeroMat <- data.frame(Aspen=zeros, CotWil=zeros, DougFir=zeros, Krumm=zeros,
                           Lodge=zeros, Ponder=zeros, SpruceFir=zeros)
rownames(zeroMat)<-treeLabels
colnames(zeroMat)<-treeLabels
confusionMat=zeroMat
```

Calculate 7x7 Confusion Matrix

A 7x7 confusion matrix is used to aid calculating sensitivity and specificity of the data sets when all seven logistic models are applied to the data.

The results vary based on the order the logistic models are applied. Different orders are presented and analyzed.

A hybrid sensitivity and specificity is generated for the seven combined logistic regression models by creating a weighted average of the sensitivity and specificity of the seven tree types.

```r
calcConfusionMatrix<-function (
  df,         # dataset with Actual Coverage Type and Estimated Coverage Type set
  ccmDebug=0  # debug: 0=no printing, 1=print details
)
{
  treeNames=c("Aspen", "Cotton&Willow", "DouglasFir", "Krummholz",
              "Lodgepole", "Ponderosa", "Spruce&Fir")
  confusionMat=zeroMat

  # Create a confusion matrix
  for (drow in 1:7) {
    actLabel<-treeNames[drow]
    for (dcol in 1:7) {
      predLabel<-treeNames[dcol]

      # populate each cell of the confusion matrix comparing the actual coverage type
      # with the coverage type estimated by the model
      confusionMat[drow,dcol]=sum(df$CovName==actLabel & df$EstTreeType==predLabel)
    }
  }

  # Abbreviate the row and column names so the table is not split up by column
  confRows<-c("Aspen_Act", "Cot&Wil", "DougFir", "Krumm",
              "Lodge", "Ponder", "SprFir")
  confCols<-c("Aspen_Pre", "Cot&Wil", "DougFir", "Krumm",
              "Lodge", "Ponder", "Spr&Fir")
  rownames(confusionMat)<-confRows
  colnames(confusionMat)<-confCols

  if (ccmDebug) {
    print("Confusion Matrix (rows are actual, columns are predicted) =")
    print(confusionMat)
  }

  # create a 7x7 zero matrix to hold statistics
  statsMat=zeroMat
  rownames(statsMat)<-treeLabels
  colnames(statsMat)<-c("TP","FP","FN","TN","Acc","Sens","Spec")

  # initialize variables
  weightedSens=0.0
  weightedSpec=0.0
  accuracy=0.0

  # Calculate statistics from confusion matrix
  for(treeIndex in 1:7) { # calculate stats for each tree coverage type
    TP = confusionMat[treeIndex,treeIndex] # True Positive for class is on the diagonal
    accuracy=accuracy+TP # caclulate accuracy by first accumulating all True Positives
    totClass=sum(confusionMat[treeIndex,]) # total number of class is the row sum (all actual values fo
```

```r
    FN = sum(confusionMat[treeIndex,])-TP # False Neg = totClass - True Pos
    # which is sum of the cells in the Actual class row not predicting the class value
    FP = sum(confusionMat[,treeIndex])-TP # False Pos = col sum of predicted values - True Pos
    # which is the sum of the cells in the predicted class that are not the actual class value
    TN =0 # Initialize True Negative
    for (drow in 1:7) { # True negative is sum of all cells not in row or col of the class
      for (dcol in 1:7) {
        if (drow != treeIndex & dcol != treeIndex) TN=TN+confusionMat[drow,dcol]
      }
    }
    statsMat[treeIndex,1]=TP
    statsMat[treeIndex,2]=FP
    statsMat[treeIndex,3]=FN
    statsMat[treeIndex,4]=TN
    statsMat[treeIndex,5]=(TP + TN)/(TP+TN+FP+FN) # Set accuracy
    statsMat[treeIndex,6]=TP/(TP+FN) # Set Sensitivity for feature - positive predicted%
    statsMat[treeIndex,7]=TN/(TN+FP) # Set Specificity for feature - negative predicted%

    # accumulate weighted sensitivity and specificity for later overall model to calculation
    weightedSens = weightedSens + (totClass * statsMat[treeIndex,6])
    weightedSpec = weightedSpec + (totClass * statsMat[treeIndex,7])
  }

  # complete weighted calculations by dividing by number of rows in data set
  weightedSens = weightedSens / nrow(df)
  weightedSpec = weightedSpec / nrow(df)
  accuracy=accuracy/nrow(df)

  if (ccmDebug) {
    print("Stats")
    print(statsMat)
    print(paste("Weighted Avg Sens=",weightedSens))
    print(paste("Weighted Avg Spec=",weightedSpec))
    print(paste("Accuracy         =",accuracy))
  }
  c(weightedSens, weightedSpec, accuracy)
}
```

Calculate Tree Type Helper Function

```r
# Calculate tree types based on passed in threshholds.
# Probabilities were previously calculated
calcTreeTypes <-
function(tds,               # tree data set
         mode,
         AspenThresh,
         CotWillThresh,
         DougFirThresh,
         KrummThresh,
         LodgeThresh,
         PonderThresh,
         SprFirThresh
```

```
         )
{
  tds$EstTreeType="X" # set Estimated tree type to default
  #tds$EstTreeType=as.character(tds$EstTreeType)

  if(1 == 2) {
    print(AspenThresh)
    print(CotWillThresh)
    print(DougFirThresh)
    print(KrummThresh)
    print(LodgeThresh)
    print(PonderThresh)
    print(SprFirThresh)
  }

  print(paste("calcTreeType Mode=",mode))
  # determine tree types applying logistic regression models in order described by mode
  if (mode == 1) { # sensitivity order, highest to lowest, update all
    print(paste("calcTreeType(Mode 1)"))
    tds$EstTreeType[tds$EstTreeType=="X" & tds$PonderProb > PonderThresh] = "Ponderosa"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$DougFirProb > DougFirThresh] = "DouglasFir"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$KrummProb > KrummThresh] = "Krummholz"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$CotWilProb > CotWilThresh] = "Cotton&Willow"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$AspenProb > AspenThresh] = "Aspen"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$SprFirProb > SprFirThresh] = "Spruce&Fir"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$LodgeProb > LodgeThresh] = "Lodgepole"
  } else if (mode == 2) { # specifcity order, highest to lowest, update unassigned only
    print(paste("calcTreeType(Mode 2)"))
    tds$EstTreeType[tds$EstTreeType=="X" & tds$CotWilProb > CotWilThresh]="Cotton&Willow"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$PonderProb > PonderThresh] = "Ponderosa"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$KrummProb > KrummThresh] = "Krummholz"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$DougFirProb > DougFirThresh] = "DouglasFir"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$LodgeProb > LodgeThresh] = "Lodgepole"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$AspenProb > AspenThresh] = "Aspen"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$SprFirProb > SprFirThresh] = "Spruce&Fir"
  } else if (mode ==3) { # specifcity order, lowest to highest, update unassigned only
    print(paste("calcTreeType(Mode 3)"))
    tds$EstTreeType[tds$EstTreeType=="X" & tds$SprFirProb > SprFirThresh] = "Spruce&Fir"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$AspenProb > AspenThresh] = "Aspen"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$LodgeProb > LodgeThresh] = "Lodgepole"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$DougFirProb > DougFirThresh] = "DouglasFir"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$KrummProb > KrummThresh] = "Krummholz"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$PonderProb > PonderThresh] = "Ponderosa"
    tds$EstTreeType[tds$EstTreeType=="X" & tds$CotWilProb > CotWilThresh]="Cotton&Willow"
  } else { # specifcity order, lowest to highest, update all
    print(paste("calcTreeType(Mode 4)"))
    tds$EstTreeType[tds$SprFirProb > SprFirThresh] = "Spruce&Fir"
    tds$EstTreeType[tds$AspenProb > AspenThresh] = "Aspen"
    tds$EstTreeType[tds$LodgeProb > LodgeThresh] = "Lodgepole"
    tds$EstTreeType[tds$DougFirProb > DougFirThresh] = "DouglasFir"
    tds$EstTreeType[tds$KrummProb > KrummThresh] = "Krummholz"
    tds$EstTreeType[tds$PonderProb > PonderThresh] = "Ponderosa"
    tds$EstTreeType[tds$CotWilProb > CotWilThresh]="Cotton&Willow"
```

```
  }

  ccm=calcConfusionMatrix(tds,1) # report stats for the combined 7 logistic regression models
  ccm
}
```

Find Model Threshold Helper Function

A function to search for optimum thresholds for the combined seven logistic regression models is shown next.

```
# find Threshholdsw optimized for the seven combined logistic regression models
findModelThresholds <-
function(tds,
         printLevel,
         findThreshold,
         mode,
         iterations,
         initAspenThresh,
         initCotWillThresh,
         initDougFirThresh,
         initKrummThresh,
         initLodgeThresh,
         initPonderThresh,
         initSprFirThresh
         ) {

  if (printLevel > 1) print(table(tds$EstTreeType))

  # Reset data
  tds$EstTreeType="X"

  threshs =c(initAspenThresh, initCotWillThresh, initDougFirThresh, initKrummThresh,
              initLodgeThresh, initLodgeThresh, initSprFirThresh)

  for (i in 1:iterations) { # number of times to optimize complete set of thresholds

    for (j in 1:7) { # variables to optimize

      start=0.1
      end = 0.9
      increment = 0.1
      curThresh=start
      bestAccuracy = 0.0
      bestThresh = threshs[j]

      for (k in 1:2) { # optimize increments by 0.1, 0.01, 0.001
        more=TRUE
        #bestThresh=threshs[j] # save current threshold for kth tree type

        if (printLevel > 0) {
          print(paste("Start=",start,", end=",end, ", inc=",increment))
          print("--------------------------")
        }
```

```
    while(more) {
      threshs[j]= curThresh

      # determine tree types applying logistic regression models in order described
      # in comments below
      if (mode == 1) { # sensitivity order, highest to lowest, update only if unassigned
        tds$EstTreeType[tds$EstTreeType=="X" & tds$PonderProb > threshs[6]] = "Ponderosa"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$DougFirProb > threshs[3]] = "DouglasFir"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$KrummProb > threshs[4]] = "Krummholz"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$CotWilProb > threshs[2]]="Cotton&Willow"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$AspenProb > threshs[1]] = "Aspen"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$SprFirProb > threshs[7]] = "Spruce&Fir"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$LodgeProb > threshs[5]] = "Lodgepole"
      } else if (mode ==2) { # specfcity order, highest to lowest, update unassigned only
        tds$EstTreeType[tds$EstTreeType=="X" & tds$CotWilProb > threshs[2]]="Cotton&Willow"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$PonderProb > threshs[6]] = "Ponderosa"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$KrummProb > threshs[4]] = "Krummholz"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$DougFirProb > threshs[3]] = "DouglasFir"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$LodgeProb > threshs[5]] = "Lodgepole"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$AspenProb > threshs[1]] = "Aspen"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$SprFirProb > threshs[7]] = "Spruce&Fir"
      } else if (mode ==3) { # specfcity order, lowest to highest, update unassigned only
        tds$EstTreeType[tds$EstTreeType=="X" & tds$SprFirProb > threshs[7]] = "Spruce&Fir"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$AspenProb > threshs[1]] = "Aspen"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$LodgeProb > threshs[5]] = "Lodgepole"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$DougFirProb > threshs[3]] = "DouglasFir"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$KrummProb > threshs[4]] = "Krummholz"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$PonderProb > threshs[6]] = "Ponderosa"
        tds$EstTreeType[tds$EstTreeType=="X" & tds$CotWilProb > threshs[2]]="Cotton&Willow"
      } else { # specfcity order, lowest to highest, update all
        tds$EstTreeType[tds$SprFirProb > threshs[7]] = "Spruce&Fir"
        tds$EstTreeType[tds$AspenProb > threshs[1]] = "Aspen"
        tds$EstTreeType[tds$LodgeProb > threshs[5]] = "Lodgepole"
        tds$EstTreeType[tds$DougFirProb > threshs[3]] = "DouglasFir"
        tds$EstTreeType[tds$KrummProb > threshs[4]] = "Krummholz"
        tds$EstTreeType[tds$PonderProb > threshs[6]] = "Ponderosa"
        tds$EstTreeType[tds$CotWilProb > threshs[2]]="Cotton&Willow"
      }
      #accuracy = (sum(tds$EstTreeType == tds$CovName))/nrow(tds)

      result=calcConfusionMatrix(tds,0)
      # accuracy=result[1]^2 + result[2]^2 # sensitivity^2 + specificity^2
      accuracy=result[1] + result[2]       # sensitivity   + specificity

      # reset data
      tds$EstTreeType="X"

      # print thresholds
      if (printLevel > 0) {
        printAccuracy = as.integer(accuracy * 100000)/1000.0
        print(paste("Accuracy(",threshs[1],threshs[2],threshs[3],
                    threshs[4],threshs[5],threshs[6],threshs[7],")=",
                    printAccuracy, ", i=",i,", j=",j,", bestThresh=",bestThresh))
```

```
        }

        # if accuracy improves, save best accuracy and threshold
        if (accuracy > bestAccuracy) {
          bestAccuracy = accuracy
          bestThresh = curThresh
        }
        curThresh = curThresh + increment
        if (curThresh > end) more = FALSE
      }

      # set new start, end and increment
      start = bestThresh - increment
      end = bestThresh + increment
      increment = increment / 10.0
      if (start <= 0.0) start = 0.0 + increment
      if (end >= 1.0) end = 1.0 - increment

      curThresh = start
    }

    threshs[j]=bestThresh
  }
}

if (printLevel) print(table(tds$EstTreeType))

c(bestAccuracy,threshs)
}
```

Determine Tree Types

Determine tree type using the logistic regression model that were previously developed. Different order of applying the models are presented and discussed.

```
##                                Base Acc  Sens  Spec AUC  Count Thresh
## 24     Ponderosa Sig Ind        93% 92%  97%  92% 98%  10726  0.068      X
## 11   Douglas Fir Sig Agg        97% 87%  97%  86% 95%   5210  0.033      X
## 15     Krummholz Sig Agg        96% 90%  95%  89% 97%   6153  0.029      X
## 7  Cotton/Willow Sig Agg        99% 95%  94%  95% 98%    824  0.008      X
## 4          Aspen Sig Ind        98% 68%  93%  68% 87%   2848  0.011      X
## 19     Lodgepole Sig Agg        51% 75%  79%  72% 82%  84990  0.482      X
## 27    Spruce/Fir Sig Agg        63% 73%  87%  66% 83%  63552  0.307      X
```

Initial testing of the combined regression models uses the thresholds that were found when the individual regression models were built. Thresholds that optimize sensitivity and specificity for the combined models will be discussed later.

```
PonderThresh =0.068
DougFirThresh=0.033
KrummThresh  =0.029 # 0.040 # 0.029
CotWilThresh =0.008 # 0.020 # 0.008
AspenThresh  =0.011 # 0.020 # 0.011
LodgeThresh  =0.482
```

```
SprFirThresh =0.307
```

High Sensitiviy - Update Unassigned Method 1

The first method tested applies the logistic regression models in sensitivity order from highest to lowest. The tree estimates are updated by subsequent models only if the tree coverage type has not already been assigned.

```
# use training set with mode=1
# mode=1: apply regression models in sensitivity order, high to low, update only if unassigned
ctt1=calcTreeTypes(forestTrain, 1, AspenThresh, CotWilThresh, DougFirThresh,
             KrummThresh, LodgeThresh, PonderThresh, SprFirThresh)
```

```
## [1] "calcTreeType Mode= 1"
## [1] "calcTreeType(Mode 1)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##           Aspen_Pre Cot&Wil DougFir Krumm Lodge Ponder Spr&Fir
## Aspen_Act      4132      67    1061     1    98   1080     192
## Cot&Wil           0       0       5     0     0   1918       0
## DougFir          68       0    1456     0     0  10633       0
## Krumm            69       0       0 13726     0     60     494
## Lodge         61571    1179   13614  6819 49201  15468   49879
## Ponder          319       4     364     2     4  24326       0
## SprFir        20712     315    3302 31551  8198    572   83413
## [1] "Stats"
##              TP    FP     FN     TN       Acc      Sens      Spec
## Aspen      4132 82739   2499 316503 0.7899885 0.6231338 0.7927598
## CotWill       0  1565   1923 402385 0.9914062 0.0000000 0.9961258
## DougFir    1456 18346  10701 375370 0.9284333 0.1197664 0.9534030
## Krumm     13726 38373    623 353151 0.9039207 0.9565823 0.9019907
## Lodge     49201  8300 148530 199842 0.6135983 0.2488280 0.9601234
## Ponder    24326 29731    693 351123 0.9250406 0.9723011 0.9219360
## SpruceFir 83413 50565  64650 207245 0.7161304 0.5633615 0.8038672
## [1] "Weighted Avg Sens= 0.433366362681918"
## [1] "Weighted Avg Spec= 0.894105080195047"
## [1] "Accuracy         = 0.433366362681918"
```

```
resultSummary <- data.frame("Description"=character(),
                          "Aspen"=double(), "CotWl"=double(), "DougF"=double(),
                          "Krumm"=double(), "Lodge"=double(), "Pondr"=double(),
                          "SprFr"=double(), "Sens"=double(), "Spec"=double(),
                          "SensPlusSpec"=double(), stringsAsFactors=FALSE)

tsum=as.integer((ctt1[1]+ctt1[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                          c("HiSens-UnAsgn-1",
                          Aspen=AspenThresh, CotWl=CotWilThresh,
                          DougF=DougFirThresh, Krumm=KrummThresh,
                          Lodge=LodgeThresh, Pondr=PonderThresh,
                          SprFr=SprFirThresh,
                          Sens=as.integer(ctt1[1]*1000)/1000.0,
                          Spec=as.integer(ctt1[2]*1000)/1000.0,
                          SensPlusSpec=tsum)
```

The weighted sensitivity/specificity of the 'High Sensitivity-Update Unassigned' method is 43.336% / 89.41%.

It is interesting that the accuracy and the weighted sensitivity are identical. The formulas and code have been doubled checked to ensure the calculations are correct.

Except for the largest populations, the specificities are at least 97% which is good. The large tree populations unfortunately only have specificities of 72% and 81%.

The sensitivities for several of the tree types is pretty low, but they are the smaller populations and don't affect the weighted value much. Unfortunately the large population tree types come in at 75%.

High Specificity - Update Unassigned Method 2

The second method tested applies the logistic regression models in specificity order from highest to lowest. The tree estimates are updated by subsequent models only if the tree coverage type has not already been assigned.

```
# use training set with mode=2
# mode=2: apply regression models in specfcity order, high to low, update only if unassigned
ctt2=calcTreeTypes(forestTrain, 2, AspenThresh, CotWilThresh, DougFirThresh,
             KrummThresh, LodgeThresh, PonderThresh, SprFirThresh)
```

```
## [1] "calcTreeType Mode= 2"
## [1] "calcTreeType(Mode 2)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##            Aspen_Pre Cot&Wil DougFir Krumm  Lodge Ponder Spr&Fir
## Aspen_Act       133     153    1008     1   4244   1047      45
## Cot&Wil           0    1851       1     0      0     71       0
## DougFir           5    4172    1360     0     63   6557       0
## Krumm            68       0       0 13726     60     60     435
## Lodge          8687    2648   13193  6819 131850  14420   20114
## Ponder          154   10168     315     2    169  14211       0
## SprFir         9375     567    3176 31551  40767    446   62181
## [1] "Stats"
##               TP    FP    FN     TN       Acc       Sens      Spec
## Aspen        133 18289  6498 380953 0.9389292 0.02005731 0.9541907
## CotWill     1851 17708    72 386242 0.9561932 0.96255850 0.9561629
## DougFir     1360 17693 10797 376023 0.9298056 0.11186970 0.9550615
## Krumm      13726 38373   623 353151 0.9039207 0.95658234 0.9019907
## Lodge     131850 45303 65881 162839 0.7260621 0.66681502 0.7823457
## Ponder     14211 22601 10808 358253 0.9176861 0.56800831 0.9406570
## SpruceFir  62181 20594 85882 237216 0.7376618 0.41996312 0.9201195
## [1] "Weighted Avg Sens= 0.553988232372532"
## [1] "Weighted Avg Spec= 0.853640399441076"
## [1] "Accuracy         = 0.553988232372532"
```

```
tsum=as.integer((ctt2[1]+ctt2[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                          c("HiSpec-UnAsgn-1",
                          Aspen=AspenThresh, CotWl=CotWilThresh,
                          DougF=DougFirThresh, Krumm=KrummThresh,
                          Lodge=LodgeThresh, Pondr=PonderThresh,
                          SprFr=SprFirThresh,
                          Sens=as.integer(ctt2[1]*1000)/1000.0,
                          Spec=as.integer(ctt2[2]*1000)/1000.0,
                          SensPlusSpec=tsum)
```

The weighted sensitivity/specificity of the 'High Specificity - Update Unassigned' model is 55.398% / 85.364%.

The specificity of this model has improved over the first by 6% but the sensitivity has decreased by 10%. It is not an overall improvement.

Low Specificity - Update Unassigned Method 3

The third method tested applies the logistic regression models in specificity order from lowest to highest. The tree estimates are updated by subsequent models only if the tree coverage type has not already been assigned.

```
 # use training set with mode=3
# mode=3: apply regression models in specfcity order, low to high, update only if unassigned
ctt3=calcTreeTypes(forestTrain, 3, AspenThresh, CotWilThresh, DougFirThresh,
              KrummThresh, LodgeThresh, PonderThresh, SprFirThresh)
```

```
## [1] "calcTreeType Mode= 3"
## [1] "calcTreeType(Mode 3)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##             Aspen_Pre Cot&Wil DougFir Krumm Lodge Ponder Spr&Fir
## Aspen_Act        5940       0      18    12   174     63     424
## Cot&Wil             0       0    1653     0     2    268       0
## DougFir          5161       0    6706     0   114    174       2
## Krumm              95       0       0   420     5      0   13829
## Lodge           69630       1    2080   302 51966   1056   72696
## Ponder           9607       4   13941    15    76   1376       0
## SprFir           9627       0     109   579  8941      2  128805
## [1] "Stats"
##                TP    FP     FN     TN       Acc       Sens      Spec
## Aspen        5940 94120    691 305122 0.7664023 0.89579249 0.7642533
## CotWill         0     5   1923 403945 0.9952497 0.00000000 0.9999876
## DougFir      6706 17801   5451 375915 0.9427111 0.55161635 0.9547872
## Krumm         420   908  13929 390616 0.9634442 0.02927033 0.9976809
## Lodge       51966  9312 145765 198830 0.6179174 0.26281160 0.9552613
## Ponder       1376  1563  23643 379291 0.9378968 0.05499820 0.9958961
## SpruceFir 128805 86951  19258 170859 0.7383196 0.86993374 0.6627322
## [1] "Weighted Avg Sens= 0.479982001873575"
## [1] "Weighted Avg Spec= 0.847881489568384"
## [1] "Accuracy         = 0.479982001873575"
```

```
tsum=as.integer((ctt3[1]+ctt3[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                          c("LoSpec-UnAsgn-1",
                          Aspen=AspenThresh, CotWl=CotWilThresh,
                          DougF=DougFirThresh, Krumm=KrummThresh,
                          Lodge=LodgeThresh, Pondr=PonderThresh,
                          SprFr=SprFirThresh,
                          Sens=as.integer(ctt3[1]*1000)/1000.0,
                          Spec=as.integer(ctt3[2]*1000)/1000.0,
                          SensPlusSpec=tsum)
```

The weighted sensitivity/specificity of the 'Low Specificity - Update Unassigned' model is 47.998% / 84.788%.

This is a further degradation from the first and second methods.

Low Specificity - Update All Method 4

The fourth method tested applies the logistic regression models in specificity order from lowest to highest. The tree estimates are updated by subsequent models even if the tree coverage type has been previously assigned.

```
# use training set with mode=4
# mode=4: apply regression models in specfcity order, low to high, update all
ctt4=calcTreeTypes(forestTrain, 4, AspenThresh, CotWilThresh, DougFirThresh,
              KrummThresh, LodgeThresh, PonderThresh, SprFirThresh)
```

```
## [1] "calcTreeType Mode= 4"
## [1] "calcTreeType(Mode 4)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##            Aspen_Pre Cot&Wil DougFir Krumm  Lodge Ponder Spr&Fir
## Aspen_Act        133     153    1008     1   4244   1047      45
## Cot&Wil            0    1851       1     0      0     71       0
## DougFir            5    4172    1360     0     63   6557       0
## Krumm             68       0       0 13726     60     60     435
## Lodge           8687    2648   13193  6819 131850  14420   20114
## Ponder           154   10168     315     2    169  14211       0
## SprFir          9375     567    3176 31551  40767    446   62181
## [1] "Stats"
##                TP    FP    FN     TN       Acc       Sens      Spec
## Aspen         133 18289  6498 380953 0.9389292 0.02005731 0.9541907
## CotWill      1851 17708    72 386242 0.9561932 0.96255850 0.9561629
## DougFir      1360 17693 10797 376023 0.9298056 0.11186970 0.9550615
## Krumm       13726 38373   623 353151 0.9039207 0.95658234 0.9019907
## Lodge      131850 45303 65881 162839 0.7260621 0.66681502 0.7823457
## Ponder      14211 22601 10808 358253 0.9176861 0.56800831 0.9406570
## SpruceFir   62181 20594 85882 237216 0.7376618 0.41996312 0.9201195
## [1] "Weighted Avg Sens= 0.553988232372532"
## [1] "Weighted Avg Spec= 0.853640399441076"
## [1] "Accuracy        = 0.553988232372532"
```

```
tsum=as.integer((ctt4[1]+ctt4[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                            c("LoSpec-All-1",
                            Aspen=AspenThresh, CotWl=CotWilThresh,
                            DougF=DougFirThresh, Krumm=KrummThresh,
                            Lodge=LodgeThresh, Pondr=PonderThresh,
                            SprFr=SprFirThresh,
                            Sens=as.integer(ctt4[1]*1000)/1000.0,
                            Spec=as.integer(ctt4[2]*1000)/1000.0,
                            SensPlusSpec=tsum)
```

The weighted sensitivity/specificity of the 'Low Specificity - Update All' model is 55.398% / 85.364%.

Low Specificity - Update All Method 4 - Manual Thresholds

The fourth method is applied again but using thresholds that were chosen by visually examining the ROC graphs for the point at which a 45 degree tangent appear on the graph. The tree estimates are updated by subsequent models even if the tree coverage type has been previously assigned.

```r
# Alternate manual threshold selection
# 0.01, 0.01, 0.02, 0.05, (0.50,0.60),  0.08, (0.40, 0.50)
AspenThresh=0.01
CotWilThresh=0.01
DougFirThresh=0.02
KrummThresh=0.05
LodgeThresh=0.50
PonderThresh=0.08
SprFirThresh=0.40

cttM=calcTreeTypes(forestTest,       # tree data set
              4,                # mode
              AspenThresh,
              CotWilThresh,
              DougFirThresh,
              KrummThresh,
              LodgeThresh,
              PonderThresh,
              SprFirThresh
             )
```

```
## [1] "calcTreeType Mode= 4"
## [1] "calcTreeType(Mode 4)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##            Aspen_Pre Cot&Wil DougFir Krumm Lodge Ponder Spr&Fir
## Aspen_Act         42      41     733     0  1628    384      20
## Cot&Wil            0     766       2     0     0     56       0
## DougFir            1    1643     677     0     7   2882       0
## Krumm             45       0       0  5547    37     22     455
## Lodge           4224     716   11276  1495 51346   5388   10289
## Ponder            17    3902     307     0    60   6438       0
## SprFir          5014     141    2605  9966 15103    178   30349
## [1] "Stats"
##              TP    FP    FN     TN       Acc       Sens      Spec
## Aspen        42  9301  2806 161653 0.9303403 0.01474719 0.9455936
## CotWill     766  6443    58 166535 0.9625954 0.92961165 0.9627525
## DougFir     677 14923  4533 153669 0.8880565 0.12994242 0.9114845
## Krumm      5547 11461   559 156235 0.9308408 0.90845070 0.9316561
## Lodge     51346 16835 33388  72233 0.7110332 0.60596691 0.8109871
## Ponder     6438  8910  4286 154168 0.9240745 0.60033570 0.9453636
## SpruceFir 30349 10764 33007  99682 0.7481559 0.47902330 0.9025406
## [1] "Weighted Avg Sens= 0.545974538590845"
## [1] "Weighted Avg Spec= 0.860349555208483"
## [1] "Accuracy         = 0.545974538590845"
```

```r
cttM
```

```
## [1] 0.5459745 0.8603496 0.5459745
```

```r
tsum=as.integer((cttM[1]+cttM[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                c("LoSpec-All ROC",
                 Aspen=AspenThresh, CotWl=CotWilThresh,
                 DougF=DougFirThresh, Krumm=KrummThresh,
                 Lodge=LodgeThresh, Pondr=PonderThresh,
```

```
                    SprFr=SprFirThresh,
                    Sens=as.integer(cttM[1]*1000)/1000.0,
                    Spec=as.integer(cttM[2]*1000)/1000.0,
                    SensPlusSpec=tsum)
```

The weighted sensitivity/specificity of the 'Low Specificity - Update All - ROC' model is 54.597% / 86.034%.

It looks like the first model is the best. But before choosing a final model, the thresholds are adjusted to optimize each model.

```
#knitr::knit_exit()
```

Find Optimum Thresholds for Model on Training Set

Find Thresholds - High Sensitivity-Update Unassigned Model 1

Find optimized thresholds for the 'High Sensitivity-Update Unassigned' model. Start with the thresholds originally found for the individually developed regression models. Only the first threshold search shows the steps in the search. The remaining threshold searches show just the results.

```
  PonderThresh =0.068
  DougFirThresh=0.033
  KrummThresh  =0.029
  CotWilThresh =0.008
  AspenThresh  =0.011
  LodgeThresh  =0.482
  SprFirThresh =0.307

  result1 = findModelThresholds(
     forestTrain, # data set
     1,           # print Level 0:none, 1:details
     1,           # find threshold: 0=no, 1=yes
     1,           # model
     2,           # iterations to revise thresholds
     AspenThresh,
     CotWilThresh,
     DougFirThresh,
     KrummThresh,
     LodgeThresh,
     PonderThresh,
     SprFirThresh)
```

```
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "--------------------------"
## [1] "Accuracy( 0.1 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.54 , i= 1 , j= 1 , bestThresh= 0.011"
## [1] "Accuracy( 0.2 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.828 , i= 1 , j= 1 , bestThresh= 0.1"
## [1] "Accuracy( 0.3 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.826 , i= 1 , j= 1 , bestThresh= 0.2"
## [1] "Accuracy( 0.4 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.848 , i= 1 , j= 1 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.4"
## [1] "Accuracy( 0.6 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.7 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.8 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.9 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Start= 0.4 , end= 0.6 , inc= 0.01"
```

```
## [1] "-------------------------"
## [1] "Accuracy( 0.4 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.848 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.41 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.851 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.42 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.852 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.43 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.852 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.44 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.45 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.44"
## [1] "Accuracy( 0.46 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.44"
## [1] "Accuracy( 0.47 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.48 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.49 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.5 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.51 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.52 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.53 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.54 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.55 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.56 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.57 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.58 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.59 0.008 0.033 0.029 0.482 0.482 0.307 )= 140.853 , i= 1 , j= 1 , bestThresh= 0.46"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.46 0.1 0.033 0.029 0.482 0.482 0.307 )= 140.949 , i= 1 , j= 2 , bestThresh= 0.008"
## [1] "Accuracy( 0.46 0.2 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.3 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.4 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.5 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.6 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.7 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.8 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.9 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Start= 0.01 , end= 0.2 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.46 0.01 0.033 0.029 0.482 0.482 0.307 )= 140.926 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.02 0.033 0.029 0.482 0.482 0.307 )= 140.97 , i= 1 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.033 0.029 0.482 0.482 0.307 )= 140.979 , i= 1 , j= 2 , bestThresh= 0.02"
## [1] "Accuracy( 0.46 0.04 0.033 0.029 0.482 0.482 0.307 )= 140.976 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.05 0.033 0.029 0.482 0.482 0.307 )= 140.968 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.06 0.033 0.029 0.482 0.482 0.307 )= 140.962 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.07 0.033 0.029 0.482 0.482 0.307 )= 140.955 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.08 0.033 0.029 0.482 0.482 0.307 )= 140.952 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.09 0.033 0.029 0.482 0.482 0.307 )= 140.95 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.1 0.033 0.029 0.482 0.482 0.307 )= 140.949 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.11 0.033 0.029 0.482 0.482 0.307 )= 140.948 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.12 0.033 0.029 0.482 0.482 0.307 )= 140.948 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.13 0.033 0.029 0.482 0.482 0.307 )= 140.947 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.14 0.033 0.029 0.482 0.482 0.307 )= 140.946 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.15 0.033 0.029 0.482 0.482 0.307 )= 140.946 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.16 0.033 0.029 0.482 0.482 0.307 )= 140.946 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.17 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.18 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.19 0.033 0.029 0.482 0.482 0.307 )= 140.945 , i= 1 , j= 2 , bestThresh= 0.03"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
```

```
## [1] "--------------------------"
## [1] "Accuracy( 0.46 0.03 0.1 0.029 0.482 0.482 0.307 )= 140.729 , i= 1 , j= 3 , bestThresh= 0.033"
## [1] "Accuracy( 0.46 0.03 0.2 0.029 0.482 0.482 0.307 )= 139.144 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.3 0.029 0.482 0.482 0.307 )= 138.157 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.4 0.029 0.482 0.482 0.307 )= 137.654 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.5 0.029 0.482 0.482 0.307 )= 137.462 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.6 0.029 0.482 0.482 0.307 )= 137.419 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.7 0.029 0.482 0.482 0.307 )= 137.416 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.8 0.029 0.482 0.482 0.307 )= 137.416 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.9 0.029 0.482 0.482 0.307 )= 137.416 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Start= 0.01 , end= 0.2 , inc= 0.01"
## [1] "--------------------------"
## [1] "Accuracy( 0.46 0.03 0.01 0.029 0.482 0.482 0.307 )= 136.072 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.02 0.029 0.482 0.482 0.307 )= 139.302 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.03 0.029 0.482 0.482 0.307 )= 140.731 , i= 1 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.04 0.029 0.482 0.482 0.307 )= 141.302 , i= 1 , j= 3 , bestThresh= 0.03"
## [1] "Accuracy( 0.46 0.03 0.05 0.029 0.482 0.482 0.307 )= 141.429 , i= 1 , j= 3 , bestThresh= 0.04"
## [1] "Accuracy( 0.46 0.03 0.06 0.029 0.482 0.482 0.307 )= 141.359 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.07 0.029 0.482 0.482 0.307 )= 141.261 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.08 0.029 0.482 0.482 0.307 )= 141.106 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.09 0.029 0.482 0.482 0.307 )= 140.922 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.1 0.029 0.482 0.482 0.307 )= 140.729 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.11 0.029 0.482 0.482 0.307 )= 140.556 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.12 0.029 0.482 0.482 0.307 )= 140.387 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.13 0.029 0.482 0.482 0.307 )= 140.222 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.14 0.029 0.482 0.482 0.307 )= 140.059 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.15 0.029 0.482 0.482 0.307 )= 139.885 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.16 0.029 0.482 0.482 0.307 )= 139.739 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.17 0.029 0.482 0.482 0.307 )= 139.587 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.18 0.029 0.482 0.482 0.307 )= 139.432 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.46 0.03 0.19 0.029 0.482 0.482 0.307 )= 139.284 , i= 1 , j= 3 , bestThresh= 0.05"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "--------------------------"
## [1] "Accuracy( 0.46 0.03 0.05 0.1 0.482 0.482 0.307 )= 143.79 , i= 1 , j= 4 , bestThresh= 0.029"
## [1] "Accuracy( 0.46 0.03 0.05 0.2 0.482 0.482 0.307 )= 144.158 , i= 1 , j= 4 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.3 0.482 0.482 0.307 )= 144.308 , i= 1 , j= 4 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.4 0.482 0.482 0.307 )= 144.385 , i= 1 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.46 0.03 0.05 0.5 0.482 0.482 0.307 )= 144.189 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.6 0.482 0.482 0.307 )= 143.787 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.7 0.482 0.482 0.307 )= 143.32 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.8 0.482 0.482 0.307 )= 143.121 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.9 0.482 0.482 0.307 )= 142.948 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Start= 0.3 , end= 0.5 , inc= 0.01"
## [1] "--------------------------"
## [1] "Accuracy( 0.46 0.03 0.05 0.3 0.482 0.482 0.307 )= 144.308 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.31 0.482 0.482 0.307 )= 144.323 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.32 0.482 0.482 0.307 )= 144.352 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.33 0.482 0.482 0.307 )= 144.378 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.34 0.482 0.482 0.307 )= 144.388 , i= 1 , j= 4 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.35 0.482 0.482 0.307 )= 144.399 , i= 1 , j= 4 , bestThresh= 0.34"
## [1] "Accuracy( 0.46 0.03 0.05 0.36 0.482 0.482 0.307 )= 144.401 , i= 1 , j= 4 , bestThresh= 0.35"
## [1] "Accuracy( 0.46 0.03 0.05 0.37 0.482 0.482 0.307 )= 144.413 , i= 1 , j= 4 , bestThresh= 0.36"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.482 0.482 0.307 )= 144.42 , i= 1 , j= 4 , bestThresh= 0.37"
## [1] "Accuracy( 0.46 0.03 0.05 0.39 0.482 0.482 0.307 )= 144.399 , i= 1 , j= 4 , bestThresh= 0.38"
```

```
## [1] "Accuracy( 0.46 0.03 0.05 0.4 0.482 0.482 0.307 )= 144.385 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.41 0.482 0.482 0.307 )= 144.366 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.42 0.482 0.482 0.307 )= 144.358 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.43 0.482 0.482 0.307 )= 144.344 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.44 0.482 0.482 0.307 )= 144.344 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.45 0.482 0.482 0.307 )= 144.315 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.46 0.482 0.482 0.307 )= 144.282 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.47 0.482 0.482 0.307 )= 144.261 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.48 0.482 0.482 0.307 )= 144.234 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.46 0.03 0.05 0.49 0.482 0.482 0.307 )= 144.212 , i= 1 , j= 4 , bestThresh= 0.38"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.1 0.482 0.307 )= 146.495 , i= 1 , j= 5 , bestThresh= 0.482"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.2 0.482 0.307 )= 145.744 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.3 0.482 0.307 )= 145.319 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.4 0.482 0.307 )= 144.98 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.5 0.482 0.307 )= 144.283 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.6 0.482 0.307 )= 142.609 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.7 0.482 0.307 )= 132.734 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.8 0.482 0.307 )= 115.3 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.9 0.482 0.307 )= 97.176 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Start= 0.01 , end= 0.2 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.482 0.307 )= 147.078 , i= 1 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.02 0.482 0.307 )= 147.055 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.03 0.482 0.307 )= 147.011 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.04 0.482 0.307 )= 146.979 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.05 0.482 0.307 )= 146.92 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.06 0.482 0.307 )= 146.809 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.07 0.482 0.307 )= 146.714 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.08 0.482 0.307 )= 146.622 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.09 0.482 0.307 )= 146.548 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.1 0.482 0.307 )= 146.495 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.11 0.482 0.307 )= 146.441 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.12 0.482 0.307 )= 146.379 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.13 0.482 0.307 )= 146.315 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.14 0.482 0.307 )= 146.249 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.15 0.482 0.307 )= 146.183 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.16 0.482 0.307 )= 146.088 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.17 0.482 0.307 )= 145.998 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.18 0.482 0.307 )= 145.908 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.19 0.482 0.307 )= 145.827 , i= 1 , j= 5 , bestThresh= 0.01"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.1 0.307 )= 147.374 , i= 1 , j= 6 , bestThresh= 0.482"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.2 0.307 )= 147.819 , i= 1 , j= 6 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.3 0.307 )= 147.762 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.4 0.307 )= 147.4 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.5 0.307 )= 147.026 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.6 0.307 )= 146.703 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.7 0.307 )= 146.212 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.8 0.307 )= 145.585 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.9 0.307 )= 144.775 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Start= 0.1 , end= 0.3 , inc= 0.01"
```

```
## [1] "-------------------------"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.1 0.307 )= 147.374 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.11 0.307 )= 147.478 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.12 0.307 )= 147.553 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.13 0.307 )= 147.612 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.14 0.307 )= 147.66 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.15 0.307 )= 147.698 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.16 0.307 )= 147.733 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.17 0.307 )= 147.758 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.18 0.307 )= 147.777 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.19 0.307 )= 147.798 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.2 0.307 )= 147.819 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.21 0.307 )= 147.845 , i= 1 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.307 )= 147.857 , i= 1 , j= 6 , bestThresh= 0.21"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.23 0.307 )= 147.856 , i= 1 , j= 6 , bestThresh= 0.22"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.24 0.307 )= 147.856 , i= 1 , j= 6 , bestThresh= 0.22"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.25 0.307 )= 147.848 , i= 1 , j= 6 , bestThresh= 0.22"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.26 0.307 )= 147.846 , i= 1 , j= 6 , bestThresh= 0.22"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.27 0.307 )= 147.836 , i= 1 , j= 6 , bestThresh= 0.22"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.28 0.307 )= 147.807 , i= 1 , j= 6 , bestThresh= 0.22"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.29 0.307 )= 147.789 , i= 1 , j= 6 , bestThresh= 0.22"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.1 )= 128.71 , i= 1 , j= 7 , bestThresh= 0.307"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.2 )= 140.269 , i= 1 , j= 7 , bestThresh= 0.1"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.3 )= 147.51 , i= 1 , j= 7 , bestThresh= 0.2"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.4 )= 150.25 , i= 1 , j= 7 , bestThresh= 0.3"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.5 )= 148.391 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.6 )= 141.822 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.7 )= 134.049 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.8 )= 125.311 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.9 )= 117.757 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Start= 0.3 , end= 0.5 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.3 )= 147.51 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.31 )= 147.966 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.32 )= 148.383 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.33 )= 148.741 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.34 )= 149.084 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.35 )= 149.413 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.36 )= 149.651 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.37 )= 149.913 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.38 )= 150.075 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.39 )= 150.176 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.4 )= 150.25 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 1 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.42 )= 150.228 , i= 1 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.43 )= 150.151 , i= 1 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.44 )= 150.047 , i= 1 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.45 )= 149.911 , i= 1 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.46 )= 149.702 , i= 1 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.47 )= 149.461 , i= 1 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.48 )= 149.155 , i= 1 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.49 )= 148.752 , i= 1 , j= 7 , bestThresh= 0.41"
```

```
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.1 0.03 0.05 0.38 0.01 0.22 0.41 )= 149.804 , i= 2 , j= 1 , bestThresh= 0.46"
## [1] "Accuracy( 0.2 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.179 , i= 2 , j= 1 , bestThresh= 0.1"
## [1] "Accuracy( 0.3 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.218 , i= 2 , j= 1 , bestThresh= 0.2"
## [1] "Accuracy( 0.4 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.253 , i= 2 , j= 1 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.4"
## [1] "Accuracy( 0.6 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.7 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.8 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.9 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Start= 0.4 , end= 0.6 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.4 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.253 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.41 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.255 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.42 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.43 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.44 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.258 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.45 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.258 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.46 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.47 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.48 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.49 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.5 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.51 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.52 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.53 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.54 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.55 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.56 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.57 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.58 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Accuracy( 0.59 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 1 , bestThresh= 0.5"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.1 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.5 0.2 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.3 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.4 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.5 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.6 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.7 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.8 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.9 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Start= 0.01 , end= 0.2 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.01 0.05 0.38 0.01 0.22 0.41 )= 150.146 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.02 0.05 0.38 0.01 0.22 0.41 )= 150.241 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.03 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 2 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 2 , bestThresh= 0.03"
## [1] "Accuracy( 0.5 0.05 0.05 0.38 0.01 0.22 0.41 )= 150.258 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.06 0.05 0.38 0.01 0.22 0.41 )= 150.258 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.07 0.05 0.38 0.01 0.22 0.41 )= 150.258 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.08 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
```

```
## [1] "Accuracy( 0.5 0.09 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.1 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.11 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.12 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.13 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.14 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.15 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.16 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.17 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.18 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Accuracy( 0.5 0.19 0.05 0.38 0.01 0.22 0.41 )= 150.257 , i= 2 , j= 2 , bestThresh= 0.04"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.1 0.38 0.01 0.22 0.41 )= 151.053 , i= 2 , j= 3 , bestThresh= 0.05"
## [1] "Accuracy( 0.5 0.04 0.2 0.38 0.01 0.22 0.41 )= 150.798 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.3 0.38 0.01 0.22 0.41 )= 150.609 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.4 0.38 0.01 0.22 0.41 )= 150.508 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.5 0.38 0.01 0.22 0.41 )= 150.48 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.6 0.38 0.01 0.22 0.41 )= 150.478 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.7 0.38 0.01 0.22 0.41 )= 150.478 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.8 0.38 0.01 0.22 0.41 )= 150.478 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.9 0.38 0.01 0.22 0.41 )= 150.478 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Start= 0.01 , end= 0.2 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.01 0.38 0.01 0.22 0.41 )= 142.641 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.02 0.38 0.01 0.22 0.41 )= 146.864 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.03 0.38 0.01 0.22 0.41 )= 148.794 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.04 0.38 0.01 0.22 0.41 )= 149.782 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.05 0.38 0.01 0.22 0.41 )= 150.259 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.06 0.38 0.01 0.22 0.41 )= 150.591 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.07 0.38 0.01 0.22 0.41 )= 150.822 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.08 0.38 0.01 0.22 0.41 )= 150.944 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.09 0.38 0.01 0.22 0.41 )= 151.009 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.1 0.38 0.01 0.22 0.41 )= 151.053 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.11 0.38 0.01 0.22 0.41 )= 151.082 , i= 2 , j= 3 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.38 0.01 0.22 0.41 )= 151.094 , i= 2 , j= 3 , bestThresh= 0.11"
## [1] "Accuracy( 0.5 0.04 0.13 0.38 0.01 0.22 0.41 )= 151.079 , i= 2 , j= 3 , bestThresh= 0.12"
## [1] "Accuracy( 0.5 0.04 0.14 0.38 0.01 0.22 0.41 )= 151.059 , i= 2 , j= 3 , bestThresh= 0.12"
## [1] "Accuracy( 0.5 0.04 0.15 0.38 0.01 0.22 0.41 )= 151.029 , i= 2 , j= 3 , bestThresh= 0.12"
## [1] "Accuracy( 0.5 0.04 0.16 0.38 0.01 0.22 0.41 )= 150.983 , i= 2 , j= 3 , bestThresh= 0.12"
## [1] "Accuracy( 0.5 0.04 0.17 0.38 0.01 0.22 0.41 )= 150.938 , i= 2 , j= 3 , bestThresh= 0.12"
## [1] "Accuracy( 0.5 0.04 0.18 0.38 0.01 0.22 0.41 )= 150.887 , i= 2 , j= 3 , bestThresh= 0.12"
## [1] "Accuracy( 0.5 0.04 0.19 0.38 0.01 0.22 0.41 )= 150.846 , i= 2 , j= 3 , bestThresh= 0.12"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.12 0.1 0.01 0.22 0.41 )= 150.858 , i= 2 , j= 4 , bestThresh= 0.38"
## [1] "Accuracy( 0.5 0.04 0.12 0.2 0.01 0.22 0.41 )= 150.997 , i= 2 , j= 4 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.3 0.01 0.22 0.41 )= 151.058 , i= 2 , j= 4 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.4 0.01 0.22 0.41 )= 151.042 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.5 0.01 0.22 0.41 )= 150.782 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.6 0.01 0.22 0.41 )= 150.376 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.7 0.01 0.22 0.41 )= 149.915 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.8 0.01 0.22 0.41 )= 149.72 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.9 0.01 0.22 0.41 )= 149.55 , i= 2 , j= 4 , bestThresh= 0.3"
```

```
## [1] "Start= 0.2 , end= 0.4 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.12 0.2 0.01 0.22 0.41 )= 150.997 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.21 0.01 0.22 0.41 )= 150.998 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.22 0.01 0.22 0.41 )= 151.007 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.23 0.01 0.22 0.41 )= 151.008 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.24 0.01 0.22 0.41 )= 151.013 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.25 0.01 0.22 0.41 )= 151.031 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.26 0.01 0.22 0.41 )= 151.028 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.27 0.01 0.22 0.41 )= 151.041 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.28 0.01 0.22 0.41 )= 151.046 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.29 0.01 0.22 0.41 )= 151.051 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.3 0.01 0.22 0.41 )= 151.058 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.31 0.01 0.22 0.41 )= 151.063 , i= 2 , j= 4 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.32 0.01 0.22 0.41 )= 151.082 , i= 2 , j= 4 , bestThresh= 0.31"
## [1] "Accuracy( 0.5 0.04 0.12 0.33 0.01 0.22 0.41 )= 151.093 , i= 2 , j= 4 , bestThresh= 0.32"
## [1] "Accuracy( 0.5 0.04 0.12 0.34 0.01 0.22 0.41 )= 151.092 , i= 2 , j= 4 , bestThresh= 0.33"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.22 0.41 )= 151.096 , i= 2 , j= 4 , bestThresh= 0.33"
## [1] "Accuracy( 0.5 0.04 0.12 0.36 0.01 0.22 0.41 )= 151.092 , i= 2 , j= 4 , bestThresh= 0.35"
## [1] "Accuracy( 0.5 0.04 0.12 0.37 0.01 0.22 0.41 )= 151.096 , i= 2 , j= 4 , bestThresh= 0.35"
## [1] "Accuracy( 0.5 0.04 0.12 0.38 0.01 0.22 0.41 )= 151.094 , i= 2 , j= 4 , bestThresh= 0.35"
## [1] "Accuracy( 0.5 0.04 0.12 0.39 0.01 0.22 0.41 )= 151.065 , i= 2 , j= 4 , bestThresh= 0.35"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.1 0.22 0.41 )= 150.191 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.2 0.22 0.41 )= 149.653 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.3 0.22 0.41 )= 149.438 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.4 0.22 0.41 )= 149.025 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.5 0.22 0.41 )= 147.691 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.6 0.22 0.41 )= 140.369 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.7 0.22 0.41 )= 124.057 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.8 0.22 0.41 )= 104.969 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.9 0.22 0.41 )= 85.637 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Start= 0.01 , end= 0.2 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.22 0.41 )= 151.096 , i= 2 , j= 5 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.02 0.22 0.41 )= 151.064 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.03 0.22 0.41 )= 150.945 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.04 0.22 0.41 )= 150.759 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.05 0.22 0.41 )= 150.613 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.06 0.22 0.41 )= 150.461 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.07 0.22 0.41 )= 150.363 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.08 0.22 0.41 )= 150.284 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.09 0.22 0.41 )= 150.235 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.1 0.22 0.41 )= 150.191 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.11 0.22 0.41 )= 150.142 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.12 0.22 0.41 )= 150.079 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.13 0.22 0.41 )= 150.02 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.14 0.22 0.41 )= 149.961 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.15 0.22 0.41 )= 149.899 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.16 0.22 0.41 )= 149.819 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.17 0.22 0.41 )= 149.752 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.18 0.22 0.41 )= 149.708 , i= 2 , j= 5 , bestThresh= 0.01"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.19 0.22 0.41 )= 149.679 , i= 2 , j= 5 , bestThresh= 0.01"
```

```
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.1 0.41 )= 150.671 , i= 2 , j= 6 , bestThresh= 0.22"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.2 0.41 )= 151.113 , i= 2 , j= 6 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.3 0.41 )= 150.763 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.4 0.41 )= 150.103 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.5 0.41 )= 149.532 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.6 0.41 )= 149.044 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.7 0.41 )= 148.4 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.8 0.41 )= 147.598 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.9 0.41 )= 146.658 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Start= 0.1 , end= 0.3 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.1 0.41 )= 150.671 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.11 0.41 )= 150.792 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.12 0.41 )= 150.877 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.13 0.41 )= 150.943 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.14 0.41 )= 151.009 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.15 0.41 )= 151.053 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.16 0.41 )= 151.088 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.17 0.41 )= 151.104 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.18 0.41 )= 151.107 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.41 )= 151.114 , i= 2 , j= 6 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.2 0.41 )= 151.113 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.21 0.41 )= 151.108 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.22 0.41 )= 151.096 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.23 0.41 )= 151.069 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.24 0.41 )= 151.039 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.25 0.41 )= 150.998 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.26 0.41 )= 150.967 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.27 0.41 )= 150.923 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.28 0.41 )= 150.866 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.29 0.41 )= 150.818 , i= 2 , j= 6 , bestThresh= 0.19"
## [1] "Start= 0.1 , end= 0.9 , inc= 0.1"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.1 )= 128.676 , i= 2 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.2 )= 140.848 , i= 2 , j= 7 , bestThresh= 0.1"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.3 )= 148.29 , i= 2 , j= 7 , bestThresh= 0.2"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.4 )= 151.101 , i= 2 , j= 7 , bestThresh= 0.3"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.5 )= 149.313 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.6 )= 142.927 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.7 )= 135.257 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.8 )= 126.555 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.9 )= 119.017 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Start= 0.3 , end= 0.5 , inc= 0.01"
## [1] "-------------------------"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.3 )= 148.29 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.31 )= 148.755 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.32 )= 149.177 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.33 )= 149.548 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.34 )= 149.904 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.35 )= 150.238 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.36 )= 150.48 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.37 )= 150.749 , i= 2 , j= 7 , bestThresh= 0.4"
```

```
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.38 )= 150.915 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.39 )= 151.02 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.4 )= 151.101 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.41 )= 151.114 , i= 2 , j= 7 , bestThresh= 0.4"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.42 )= 151.088 , i= 2 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.43 )= 151.018 , i= 2 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.44 )= 150.918 , i= 2 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.45 )= 150.785 , i= 2 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.46 )= 150.586 , i= 2 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.47 )= 150.357 , i= 2 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.48 )= 150.06 , i= 2 , j= 7 , bestThresh= 0.41"
## [1] "Accuracy( 0.5 0.04 0.12 0.35 0.01 0.19 0.49 )= 149.665 , i= 2 , j= 7 , bestThresh= 0.41"
##
##      X
## 406709
```

```
result1
```

```
## [1] 1.511144 0.500000 0.040000 0.120000 0.350000 0.010000 0.190000 0.410000
```

```
accuracy=result1[1]
AspenThresh=result1[2]
CotWilThresh=result1[3]
DougFirThresh=result1[4]
KrummThresh=result1[5]
LodgeThresh=result1[6]
PonderThresh=result1[7]
SprFirThresh=result1[8]

ctt5=calcTreeTypes(forestTrain,      # tree data set
              1,                 # mode
              AspenThresh,
              CotWilThresh,
              DougFirThresh,
              KrummThresh,
              LodgeThresh,
              PonderThresh,
              SprFirThresh
             )
```

```
## [1] "calcTreeType Mode= 1"
## [1] "calcTreeType(Mode 1)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##             Aspen_Pre Cot&Wil DougFir Krumm  Lodge Ponder Spr&Fir
## Aspen_Act           0       6     154     0   5870    471     144
## Cot&Wil             0       1      25     0      9   1873       0
## DougFir             0      78    2230     0   1554   8283       0
## Krumm               0       0       0  8748    974     56    4554
## Lodge               0      29    3583   326 142076   4946   47351
## Ponder              0      23     682     0   2312  21888       0
## SprFir              0      51     415  7146  33741    103  106816
## [1] "Stats"
##               TP    FP    FN     TN       Acc        Sens      Spec
## Aspen          0     0  6645 399873 0.9836539 0.000000000 1.0000000
## CotWill        1   187  1907 404423 0.9948489 0.000524109 0.9995378
## DougFir     2230  4859  9915 389514 0.9636572 0.183614656 0.9876792
```

```
## Krumm       8748  7472  5584 384714 0.9678833 0.610382361 0.9809478
## Lodge     142076 44460 56235 163747 0.7522988 0.716430253 0.7864625
## Ponder     21888 15732  3017 365881 0.9538790 0.878859667 0.9587750
## SpruceFir 106816 52049 41456 206197 0.7699856 0.720405741 0.7984519
## [1] "Weighted Avg Sens= 0.692777882958086"
## [1] "Weighted Avg Spec= 0.818366298037202"
## [1] "Accuracy         = 0.692777882958086"
```

```
tsum=as.integer((ctt5[1]+ctt5[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                           c("HiSens-UnAsgn-2",
                           Aspen=AspenThresh, CotWl=CotWilThresh,
                           DougF=DougFirThresh, Krumm=KrummThresh,
                           Lodge=LodgeThresh, Pondr=PonderThresh,
                           SprFr=SprFirThresh,
                           Sens=as.integer(ctt5[1]*1000)/1000.0,
                           Spec=as.integer(ctt5[2]*1000)/1000.0,
                           SensPlusSpec=tsum)
```

Find Thresholds - High Specificity-Update Unassigned Model 2

Find optimized thresholds for the 'High Specificity-Update Unassigned' model. Start with the thresholds originally found for the individually developed regression models.

```
PonderThresh =0.068
DougFirThresh=0.033
KrummThresh  =0.029
CotWilThresh =0.008
AspenThresh  =0.011
LodgeThresh  =0.482
SprFirThresh =0.307

result2 = findModelThresholds(
   forestTrain, # data set
   0,           # print Level 0:none, 1:details
   1,           # find threshold: 0=no, 1=yes
   2,           # mode
   2,           # iterations to revise thresholds
   AspenThresh,
   CotWilThresh,
   DougFirThresh,
   KrummThresh,
   LodgeThresh,
   PonderThresh,
   SprFirThresh)

result2
```

```
## [1] 1.50568 0.03000 0.74000 0.12000 0.38000 0.53000 0.16000 0.01000
```

```
accuracy=result2[1]
AspenThresh=result2[2]
CotWilThresh=result2[3]
DougFirThresh=result2[4]
KrummThresh=result2[5]
```

```r
LodgeThresh=result2[6]
PonderThresh=result2[7]
SprFirThresh=result2[8]

ctt6=calcTreeTypes(forestTrain,    # tree data set
              2,               # mode
              AspenThresh,
              CotWilThresh,
              DougFirThresh,
              KrummThresh,
              LodgeThresh,
              PonderThresh,
              SprFirThresh
             )
```

```
## [1] "calcTreeType Mode= 2"
## [1] "calcTreeType(Mode 2)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##            Aspen_Pre Cot&Wil DougFir Krumm  Lodge Ponder Spr&Fir
## Aspen_Act        260       0     119     0   5126    551     585
## Cot&Wil            0     119       8     0      0   1772       3
## DougFir          105       0    1778     0   1095   8877     292
## Krumm             57       0       0  8272     80     56    5892
## Lodge           2187       0    3231   270 140331   5959   46250
## Ponder           581      60     503     0    705  22392     624
## SprFir          1553       0     406  6292  34866    118  105053
## [1] "Stats"
##               TP    FP    FN     TN       Acc       Sens      Spec
## Aspen        260  4483  6381 395304 0.9732696 0.03915073 0.9887865
## CotWill      119    60  1783 404466 0.9954654 0.06256572 0.9998517
## DougFir     1778  4267 10369 390014 0.9639887 0.14637359 0.9891778
## Krumm       8272  6562  6085 385509 0.9688826 0.57616494 0.9832632
## Lodge     140331 41872 57897 166328 0.7545223 0.70792724 0.7988857
## Ponder     22392 17333  2473 364230 0.9512681 0.90054293 0.9545737
## SpruceFir 105053 53646 43235 204494 0.7616281 0.70843898 0.7921825
## [1] "Weighted Avg Sens= 0.684039448352508"
## [1] "Weighted Avg Spec= 0.82164066329442"
## [1] "Accuracy          = 0.684039448352508"
```

```r
tsum=as.integer((ctt6[1]+ctt6[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                          c("HiSpec-UnAsgn-2",
                          Aspen=AspenThresh, CotWl=CotWilThresh,
                          DougF=DougFirThresh, Krumm=KrummThresh,
                          Lodge=LodgeThresh, Pondr=PonderThresh,
                          SprFr=SprFirThresh,
                          Sens=as.integer(ctt6[1]*1000)/1000.0,
                          Spec=as.integer(ctt6[2]*1000)/1000.0,
                          SensPlusSpec=tsum)
```

Find Thresholds - Low Specificity-Update Unassigned Model 3

Find optimized thresholds for the 'Low Specificity-Update Unassigned' model. Start with the thresholds originally found for the individually developed regression models.

```
PonderThresh =0.068
DougFirThresh=0.033
KrummThresh  =0.029
CotWilThresh =0.008
AspenThresh  =0.011
LodgeThresh  =0.482
SprFirThresh =0.307

result3 = findModelThresholds(
   forestTrain, # data set
   0,           # print Level 0:none, 1:details
   1,           # find threshold: 0=no, 1=yes
   3,           # mode
   2,           # iterations to revise thresholds
   AspenThresh,
   CotWilThresh,
   DougFirThresh,
   KrummThresh,
   LodgeThresh,
   PonderThresh,
   SprFirThresh)

result3
```

```
## [1] 1.491626 0.440000 0.100000 0.860000 0.010000 0.440000 0.010000 0.420000
```

```
accuracy=result3[1]
AspenThresh=result3[2]
CotWilThresh=result3[3]
DougFirThresh=result3[4]
KrummThresh=result3[5]
LodgeThresh=result3[6]
PonderThresh=result3[7]
SprFirThresh=result3[8]

ctt7=calcTreeTypes(forestTrain,      # tree data set
              3,                # mode
              AspenThresh,
              CotWilThresh,
              DougFirThresh,
              KrummThresh,
              LodgeThresh,
              PonderThresh,
              SprFirThresh
             )
```

```
## [1] "calcTreeType Mode= 3"
## [1] "calcTreeType(Mode 3)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##           Aspen_Pre Cot&Wil DougFir Krumm  Lodge Ponder Spr&Fir
## Aspen_Act         0       0       0   248   5541    626     125
```

```
## Cot&Wil            0        0        0      0      4    1919        0
## DougFir            0        0       92      0   4310    7755        0
## Krumm              0        0        0   1568     51       0    12738
## Lodge              2        0        0   1400 145045    5443    45698
## Ponder             8        0       45     97   3542   21336        0
## SprFir             0        0        0   2594  32979     272   112070
## [1] "Stats"
##                TP    FP    FN     TN       Acc        Sens      Spec
## Aspen           0    10  6540 398958 0.9838474 0.000000000 0.9999749
## CotWill         0     0  1923 403585 0.9952578 0.000000000 1.0000000
## DougFir        92    45 12065 393306 0.9701362 0.007567656 0.9998856
## Krumm        1568  4339 12789 386812 0.9577616 0.109215017 0.9889071
## Lodge      145045 46427 52543 161493 0.7559358 0.734077980 0.7767074
## Ponder      21336 16015  3692 364465 0.9514017 0.852485217 0.9579084
## SpruceFir 112070 58561 35845 199032 0.7671908 0.757664875 0.7726607
## [1] "Weighted Avg Sens= 0.688725845752122"
## [1] "Weighted Avg Spec= 0.802900576316593"
## [1] "Accuracy         = 0.688725845752122"
```

```
tsum=as.integer((ctt7[1]+ctt7[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                           c("LoSpec-UnAsgn-2",
                           Aspen=AspenThresh, CotWl=CotWilThresh,
                           DougF=DougFirThresh, Krumm=KrummThresh,
                           Lodge=LodgeThresh, Pondr=PonderThresh,
                           SprFr=SprFirThresh,
                           Sens=as.integer(ctt7[1]*1000)/1000.0,
                           Spec=as.integer(ctt7[2]*1000)/1000.0,
                           SensPlusSpec=tsum)
```

Find Thresholds - Low Specificity-Update All Model 4

Find optimized thresholds for the 'Low Specificity-Update All' model. Start with the thresholds originally found for the individually developed regression models.

```
PonderThresh =0.068
DougFirThresh=0.033
KrummThresh  =0.029
CotWilThresh =0.008
AspenThresh  =0.011
LodgeThresh  =0.482
SprFirThresh =0.307

result4 = findModelThresholds(
   forestTrain, # data set
   0,           # print Level 0:none, 1:details
   1,           # find threshold: 0=no, 1=yes
   4,           # mode
   2,           # iterations to revise thresholds
   AspenThresh,
   CotWilThresh,
   DougFirThresh,
   KrummThresh,
   LodgeThresh,
```

```
    PonderThresh,
    SprFirThresh)

result4
```

```
## [1] 1.50568 0.03000 0.74000 0.12000 0.38000 0.53000 0.16000 0.01000
```

```
accuracy=result4[1]
AspenThresh=result4[2]
CotWilThresh=result4[3]
DougFirThresh=result4[4]
KrummThresh=result4[5]
LodgeThresh=result4[6]
PonderThresh=result4[7]
SprFirThresh=result4[8]

ctt8=calcTreeTypes(forestTrain,      # tree data set
              4,                  # mode
              AspenThresh,
              CotWilThresh,
              DougFirThresh,
              KrummThresh,
              LodgeThresh,
              PonderThresh,
              SprFirThresh
             )
```

```
## [1] "calcTreeType Mode= 4"
## [1] "calcTreeType(Mode 4)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##            Aspen_Pre Cot&Wil DougFir Krumm  Lodge Ponder Spr&Fir
## Aspen_Act        260       0     119     0   5126    551     585
## Cot&Wil            0     119       8     0      0   1772       3
## DougFir          105       0    1778     0   1095   8877     292
## Krumm             57       0       0  8272     80     56    5892
## Lodge           2187       0    3231   270 140331   5959   46250
## Ponder           581      60     503     0    705  22392     624
## SprFir          1553       0     406  6292  34866    118  105053
## [1] "Stats"
##                TP    FP    FN     TN       Acc       Sens      Spec
## Aspen         260  4483  6381 395304 0.9732696 0.03915073 0.9887865
## CotWill       119    60  1783 404466 0.9954654 0.06256572 0.9998517
## DougFir      1778  4267 10369 390014 0.9639887 0.14637359 0.9891778
## Krumm        8272  6562  6085 385509 0.9688826 0.57616494 0.9832632
## Lodge      140331 41872 57897 166328 0.7545223 0.70792724 0.7988857
## Ponder      22392 17333  2473 364230 0.9512681 0.90054293 0.9545737
## SpruceFir  105053 53646 43235 204494 0.7616281 0.70843898 0.7921825
## [1] "Weighted Avg Sens= 0.684039448352508"
## [1] "Weighted Avg Spec= 0.82164066329442"
## [1] "Accuracy          = 0.684039448352508"
```

```
tsum=as.integer((ctt8[1]+ctt8[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                           c("LoSpec-All-2",
                           Aspen=AspenThresh, CotWl=CotWilThresh,
```

```
                              DougF=DougFirThresh, Krumm=KrummThresh,
                              Lodge=LodgeThresh, Pondr=PonderThresh,
                              SprFr=SprFirThresh,
                              Sens=as.integer(ctt8[1]*1000)/1000.0,
                              Spec=as.integer(ctt8[2]*1000)/1000.0,
                              SensPlusSpec=tsum)

resultSummary$Aspen=as.double(resultSummary$Aspen)
resultSummary$CotWl=as.double(resultSummary$CotWl)
resultSummary$DougF=as.double(resultSummary$DougF)
resultSummary$Krumm=as.double(resultSummary$Krumm)
resultSummary$Lodge=as.double(resultSummary$Lodge)
resultSummary$Pondr=as.double(resultSummary$Pondr)
resultSummary$SprFr=as.double(resultSummary$SprFr)
resultSummary$Sens=as.double(resultSummary$Sens)
resultSummary$Spec=as.double(resultSummary$Spec)
resultSummary$SensPlusSpec=as.double(resultSummary$SensPlusSpec)
```

A Summary of the different models is shown below.

```
resultSummary
```

```
##       Description Aspen CotWl DougF Krumm Lodge Pondr SprFr  Sens  Spec
## 1 HiSens-UnAsgn-1 0.011 0.008 0.033 0.029 0.482 0.068 0.307 0.433 0.894
## 2 HiSpec-UnAsgn-1 0.011 0.008 0.033 0.029 0.482 0.068 0.307 0.553 0.853
## 3 LoSpec-UnAsgn-1 0.011 0.008 0.033 0.029 0.482 0.068 0.307 0.479 0.847
## 4    LoSpec-All-1 0.011 0.008 0.033 0.029 0.482 0.068 0.307 0.553 0.853
## 5  LoSpec-All ROC 0.010 0.010 0.020 0.050 0.500 0.080 0.400 0.545 0.860
## 6 HiSens-UnAsgn-2 0.500 0.040 0.120 0.350 0.010 0.190 0.410 0.692 0.818
## 7 HiSpec-UnAsgn-2 0.030 0.740 0.120 0.380 0.530 0.160 0.010 0.684 0.821
## 8 LoSpec-UnAsgn-2 0.440 0.100 0.860 0.010 0.440 0.010 0.420 0.688 0.802
## 9    LoSpec-All-2 0.030 0.740 0.120 0.380 0.530 0.160 0.010 0.684 0.821
##   SensPlusSpec
## 1        1.327
## 2        1.407
## 3        1.327
## 4        1.407
## 5        1.406
## 6        1.511
## 7        1.505
## 8        1.491
## 9        1.505
```

The 6th model looks the best from a statistics point of view but no aspen are predicted in this model. The 9th model will be used on the test data to report the model performance.

Apply Preferred Model to Test Data

```
indx=9

ctt9=calcTreeTypes(forestTest,       # tree data set
              4,                # mode
             resultSummary$Aspen[indx],
             resultSummary$CotWl[indx],
```

```r
               resultSummary$DougF[indx],
               resultSummary$Krumm[indx],
               resultSummary$Lodge[indx],
               resultSummary$Pondr[indx],
               resultSummary$SprFr[indx]
               )
```

```
## [1] "calcTreeType Mode= 4"
## [1] "calcTreeType(Mode 4)"
## [1] "Confusion Matrix (rows are actual, columns are predicted) ="
##            Aspen_Pre Cot&Wil DougFir Krumm Lodge Ponder Spr&Fir
## Aspen_Act        121       0      50     0  2199    237     240
## Cot&Wil            0      38       9     0     0    769       0
## DougFir           44       0     781     0   491   3772     116
## Krumm             24       0       0  3495    38     22    2574
## Lodge            954       0    1485    99 60175   2533   19706
## Ponder           236      18     224     0   331   9587     257
## SprFir           633       0     162  2805 14617     52   45283
## [1] "Stats"
##              TP    FP    FN     TN       Acc       Sens      Spec
## Aspen       121  1891  2726 169439 0.9734925 0.04250088 0.9889628
## CotWill      38    18   778 173343 0.9954299 0.04656863 0.9998962
## DougFir     781  1930  4423 167043 0.9635256 0.15007686 0.9885781
## Krumm      3495  2904  2658 165120 0.9680670 0.56801560 0.9827168
## Lodge     60175 17676 24777  71549 0.7562652 0.70834118 0.8018941
## Ponder     9587  7385  1066 156139 0.9514804 0.89993429 0.9548384
## SpruceFir 45283 22893 18269  87732 0.7636772 0.71253462 0.7930576
## [1] "Weighted Avg Sens= 0.685472998169854"
## [1] "Weighted Avg Spec= 0.823379445775709"
## [1] "Accuracy         = 0.685472998169854"
```

```r
ctt9
```

```
## [1] 0.6854730 0.8233794 0.6854730
```

```r
tsum=as.integer((ctt9[1]+ctt9[2])*1000)/1000.0
resultSummary[nrow(resultSummary)+1,]<-
                c("LoSpec-All Test",
                 Aspen=resultSummary$Aspen[indx], CotWl=resultSummary$CotWl[indx],
                 DougF=resultSummary$DougF[indx], Krumm=resultSummary$Krumm[indx],
                 Lodge=resultSummary$Lodge[indx], Pondr=resultSummary$Pondr[indx],
                 SprFr=resultSummary$SprFr[indx],
                 Sens=as.integer(ctt9[1]*1000)/1000.0,
                 Spec=as.integer(ctt9[2]*1000)/1000.0,
                 SensPlusSpec=tsum)

resultSummary
```

```
##       Description Aspen CotWl DougF Krumm Lodge Pondr SprFr  Sens  Spec
## 1  HiSens-UnAsgn-1 0.011 0.008 0.033 0.029 0.482 0.068 0.307 0.433 0.894
## 2  HiSpec-UnAsgn-1 0.011 0.008 0.033 0.029 0.482 0.068 0.307 0.553 0.853
## 3  LoSpec-UnAsgn-1 0.011 0.008 0.033 0.029 0.482 0.068 0.307 0.479 0.847
## 4     LoSpec-All-1 0.011 0.008 0.033 0.029 0.482 0.068 0.307 0.553 0.853
## 5   LoSpec-All ROC  0.01  0.01  0.02  0.05   0.5  0.08   0.4 0.545  0.86
## 6  HiSens-UnAsgn-2   0.5  0.04  0.12  0.35  0.01  0.19  0.41 0.692 0.818
```

```
## 7  HiSpec-UnAsgn-2  0.03  0.74  0.12  0.38  0.53  0.16  0.01 0.684 0.821
## 8  LoSpec-UnAsgn-2  0.44   0.1  0.86  0.01  0.44  0.01  0.42 0.688 0.802
## 9     LoSpec-All-2  0.03  0.74  0.12  0.38  0.53  0.16  0.01 0.684 0.821
## 10 LoSpec-All Test  0.03  0.74  0.12  0.38  0.53  0.16  0.01 0.685 0.823
##    SensPlusSpec
## 1         1.327
## 2         1.407
## 3         1.327
## 4         1.407
## 5         1.406
## 6         1.511
## 7         1.505
## 8         1.491
## 9         1.505
## 10        1.508
```

The performance of the model strategy on the test data is nearly the same as the training data. This is not surprising since the large amount of data allows a similar distribution of data features between the training and test sets using the split function.

Conclusion

The accuracy of the model with the best sensitivity and specificity is 0.685473 which is about 1.5% less than the 70% accuracy of the neural network that this project is based. It does not improve on the accuracy of the neural network but comes very close.

Looking at the model performance during the individual model build phase it looked like the overall performance could perform better than the neural network. But the performance of the combined models ould not be predicted. They had to be combined to determine the overall performance.

While the neural network gives the better result, building and comparing the logistic models helps show which features are important to predict the model coverage and would be recommended even if the logistic regression models are not to be used.

```
#out2file="C:/Users/Tom/git/datasciencefoundation/ForestCoverage/forestTestPredict.csv"
#write.csv(forestTest, file=out2file,row.names=FALSE)
```