R Notebook

Travis Lloyd, Emina Belekanich, Mendelina Lopez

2022-06-24

## Loading packages

library(AppliedPredictiveModeling)  
library (tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

## ✔ ggplot2 3.3.5 ✔ purrr 0.3.4  
## ✔ tibble 3.1.0 ✔ dplyr 1.0.5  
## ✔ tidyr 1.1.3 ✔ stringr 1.4.0  
## ✔ readr 1.4.0 ✔ forcats 0.5.1

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()

library(psych)

##   
## Attaching package: 'psych'

## The following objects are masked from 'package:ggplot2':  
##   
## %+%, alpha

library(rpart)  
library (caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(pamr)

## Loading required package: cluster

## Loading required package: survival

##   
## Attaching package: 'survival'

## The following object is masked from 'package:caret':  
##   
## cluster

library(rpart.plot)  
library(pROC)

## Type 'citation("pROC")' for a citation.

##   
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':  
##   
## cov, smooth, var

library(MLmetrics)

##   
## Attaching package: 'MLmetrics'

## The following objects are masked from 'package:caret':  
##   
## MAE, RMSE

## The following object is masked from 'package:psych':  
##   
## AUC

## The following object is masked from 'package:base':  
##   
## Recall

library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(imputeMissings)

##   
## Attaching package: 'imputeMissings'

## The following object is masked from 'package:dplyr':  
##   
## compute

Data Set Information:

This data set contains 416 liver patient records and 167 non liver patient records.The data set was collected from north east of Andhra Pradesh, India. Selector is a class label used to divide into groups(liver patient or not). This data set contains 441 male patient records and 142 female patient records.

Any patient whose age exceeded 89 is listed as being of age “90”.

Attribute Information:

1. Age Age of the patient
2. Gender Gender of the patient
3. TB Total Bilirubin
4. DB Direct Bilirubin
5. Alkphos Alkaline Phosphotase
6. Sgpt Alamine Aminotransferase
7. Sgot Aspartate Aminotransferase
8. TP Total Protiens
9. ALB Albumin
10. A/G Ratio Albumin and Globulin Ratio
11. Selector field used to split the data into two sets (labeled by the experts)

## Loading dataset

ilpd <- read.csv("Indian Liver Patient Dataset (ILPD).csv",  
 header = FALSE)  
colnames(ilpd) = c("age", "gender", "tb", "db", "alkphos", "sgpt", "sgot", "tp", "alb", "ag\_ratio", "selector")  
ilpd$selector <- ifelse(ilpd$selector == 1, "Patient", "NonPatient")

## EDA (graphical and non-graphical representations of relationships between response variable and predictor variables)

glimpse(ilpd)

## Rows: 583  
## Columns: 11  
## $ age <int> 65, 62, 62, 58, 72, 46, 26, 29, 17, 55, 57, 72, 64, 74, 61, 2…  
## $ gender <chr> "Female", "Male", "Male", "Male", "Male", "Male", "Female", "…  
## $ tb <dbl> 0.7, 10.9, 7.3, 1.0, 3.9, 1.8, 0.9, 0.9, 0.9, 0.7, 0.6, 2.7, …  
## $ db <dbl> 0.1, 5.5, 4.1, 0.4, 2.0, 0.7, 0.2, 0.3, 0.3, 0.2, 0.1, 1.3, 0…  
## $ alkphos <int> 187, 699, 490, 182, 195, 208, 154, 202, 202, 290, 210, 260, 3…  
## $ sgpt <int> 16, 64, 60, 14, 27, 19, 16, 14, 22, 53, 51, 31, 61, 22, 53, 9…  
## $ sgot <int> 18, 100, 68, 20, 59, 14, 12, 11, 19, 58, 59, 56, 58, 30, 41, …  
## $ tp <dbl> 6.8, 7.5, 7.0, 6.8, 7.3, 7.6, 7.0, 6.7, 7.4, 6.8, 5.9, 7.4, 7…  
## $ alb <dbl> 3.3, 3.2, 3.3, 3.4, 2.4, 4.4, 3.5, 3.6, 4.1, 3.4, 2.7, 3.0, 3…  
## $ ag\_ratio <dbl> 0.90, 0.74, 0.89, 1.00, 0.40, 1.30, 1.00, 1.10, 1.20, 1.00, 0…  
## $ selector <chr> "Patient", "Patient", "Patient", "Patient", "Patient", "Patie…

describeBy(ilpd, "selector")

##   
## Descriptive statistics by group   
## selector: NonPatient  
## vars n mean sd median trimmed mad min max range  
## age 1 167 41.24 17.00 40.0 41.30 19.27 4.00 85.0 81.00  
## gender\* 2 167 1.70 0.46 2.0 1.75 0.00 1.00 2.0 1.00  
## tb 3 167 1.14 1.00 0.8 0.92 0.15 0.50 7.3 6.80  
## db 4 167 0.40 0.52 0.2 0.28 0.15 0.10 3.6 3.50  
## alkphos 5 167 219.75 140.99 186.0 194.39 37.06 90.00 1580.0 1490.00  
## sgpt 6 167 33.65 25.06 27.0 29.26 11.86 10.00 181.0 171.00  
## sgot 7 167 40.69 36.41 29.0 33.36 14.83 10.00 285.0 275.00  
## tp 8 167 6.54 1.06 6.6 6.57 1.04 3.70 9.2 5.50  
## alb 9 167 3.34 0.78 3.4 3.37 0.89 1.40 5.0 3.60  
## ag\_ratio 10 165 1.03 0.29 1.0 1.02 0.30 0.37 1.9 1.53  
## selector\* 11 167 1.00 0.00 1.0 1.00 0.00 1.00 1.0 0.00  
## skew kurtosis se  
## age 0.06 -0.69 1.32  
## gender\* -0.87 -1.25 0.04  
## tb 4.03 18.08 0.08  
## db 3.88 17.14 0.04  
## alkphos 6.02 50.65 10.91  
## sgpt 3.13 12.66 1.94  
## sgot 3.50 16.21 2.82  
## tp -0.22 -0.28 0.08  
## alb -0.30 -0.57 0.06  
## ag\_ratio 0.53 0.58 0.02  
## selector\* NaN NaN 0.00  
## ------------------------------------------------------------   
## selector: Patient  
## vars n mean sd median trimmed mad min max range skew  
## age 1 416 46.15 15.65 46.00 46.19 17.79 7.0 90.0 83.0 -0.02  
## gender\* 2 416 1.78 0.42 2.00 1.85 0.00 1.0 2.0 1.0 -1.34  
## tb 3 416 4.16 7.14 1.40 2.40 1.04 0.4 75.0 74.6 4.14  
## db 4 416 1.92 3.21 0.50 1.10 0.59 0.1 19.7 19.6 2.62  
## alkphos 5 416 319.01 268.31 229.00 261.60 88.96 63.0 2110.0 2047.0 3.32  
## sgpt 6 416 99.61 212.77 41.00 53.30 28.17 12.0 2000.0 1988.0 5.51  
## sgot 7 416 137.70 337.39 52.50 71.68 42.25 11.0 4929.0 4918.0 8.99  
## tp 8 416 6.46 1.09 6.55 6.49 0.96 2.7 9.6 6.9 -0.30  
## alb 9 416 3.06 0.79 3.00 3.06 0.74 0.9 5.5 4.6 0.05  
## ag\_ratio 10 414 0.91 0.33 0.90 0.89 0.30 0.3 2.8 2.5 1.22  
## selector\* 11 416 1.00 0.00 1.00 1.00 0.00 1.0 1.0 0.0 NaN  
## kurtosis se  
## age -0.56 0.77  
## gender\* -0.21 0.02  
## tb 26.64 0.35  
## db 7.15 0.16  
## alkphos 13.53 13.15  
## sgpt 35.27 10.43  
## sgot 108.62 16.54  
## tp 0.36 0.05  
## alb -0.24 0.04  
## ag\_ratio 4.33 0.02  
## selector\* NaN 0.00

g1 <- ggpairs(data=ilpd, title="Indian Liver Patient Dataset",  
 mapping=ggplot2::aes(colour = as.factor(selector), alpha=0.5),  
 lower=list(combo=wrap("facethist",binwidth=1)))  
g1

## Warning in ggally\_statistic(data = data, mapping = mapping, na.rm = na.rm, :  
## Removed 4 rows containing missing values

## Warning: Removed 4 rows containing non-finite values (stat\_boxplot).

## Warning in ggally\_statistic(data = data, mapping = mapping, na.rm = na.rm, :  
## Removed 4 rows containing missing values  
  
## Warning in ggally\_statistic(data = data, mapping = mapping, na.rm = na.rm, :  
## Removed 4 rows containing missing values  
  
## Warning in ggally\_statistic(data = data, mapping = mapping, na.rm = na.rm, :  
## Removed 4 rows containing missing values  
  
## Warning in ggally\_statistic(data = data, mapping = mapping, na.rm = na.rm, :  
## Removed 4 rows containing missing values  
  
## Warning in ggally\_statistic(data = data, mapping = mapping, na.rm = na.rm, :  
## Removed 4 rows containing missing values  
  
## Warning in ggally\_statistic(data = data, mapping = mapping, na.rm = na.rm, :  
## Removed 4 rows containing missing values  
  
## Warning in ggally\_statistic(data = data, mapping = mapping, na.rm = na.rm, :  
## Removed 4 rows containing missing values

## Warning: Removed 4 rows containing missing values (geom\_point).

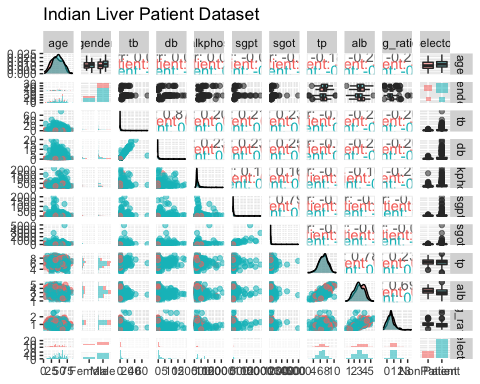
## Warning: Removed 4 rows containing non-finite values (stat\_bin).

## Warning: Removed 4 rows containing missing values (geom\_point).  
  
## Warning: Removed 4 rows containing missing values (geom\_point).  
  
## Warning: Removed 4 rows containing missing values (geom\_point).  
  
## Warning: Removed 4 rows containing missing values (geom\_point).  
  
## Warning: Removed 4 rows containing missing values (geom\_point).  
  
## Warning: Removed 4 rows containing missing values (geom\_point).  
  
## Warning: Removed 4 rows containing missing values (geom\_point).

## Warning: Removed 4 rows containing non-finite values (stat\_density).

## Warning: Removed 4 rows containing non-finite values (stat\_boxplot).

## Warning: Removed 4 rows containing non-finite values (stat\_bin).



## Data wrangling and pre-processing (handling of missing values, outliers, correlated features, etc.)

# Creating a Dummy Variable and converting the Outcome in a factor  
df <- ilpd %>%   
 mutate(male = ifelse(gender == "Male", 1, 0), .after=gender) %>%  
 mutate(selector = as.factor(selector)) %>%   
 select(-gender)  
# Zero- and Near Zero-Variance Predictors  
who = nearZeroVar(df)  
if(length(who) > 0) df <- df[,-who]  
# Identifying Correlated Predictors  
who = df %>%   
 select(-selector) %>%   
 cor(use="complete.obs") %>%   
 findCorrelation(cutoff=.75)  
   
if(length(who) > 0) df <- df[,-who]  
# Linear Dependencies  
who = df %>%   
 select(-selector) %>%   
 cov(use="complete.obs") %>%   
 findLinearCombos()  
if(length(who$remove) > 0) df <- df[,-who$remove]  
  
# Removing Influential Obs by Cooks distance  
mod <- glm(selector ~ ., data=df, family="binomial")  
cooksd <- cooks.distance(mod)  
who <- cooksd > 4\*mean(cooksd, na.rm=T)  
if(sum(who) > 0)   
 df <- df[-who,]  
  
selector <- df[ ,ncol(df)]  
df <- df[,1:ncol(df)-1]

## Data splitting (training, validation, and test sets)

set.seed(123)  
## Partition the data with stratified sampling  
training <- createDataPartition(selector, p = .8, list = FALSE)  
## Partition train and test sets  
df\_train <- df[training, ]  
df\_test <- df[-training, ]  
## Partition for the train and test sets for the response variable  
selector\_train <- selector[training]  
selector\_test <- selector[-training]  
  
## Impute the missing data using knn method built into caret package  
trainimp <- preProcess(df\_train, "knnImpute")  
df\_train <- predict(trainimp, df\_train)  
df\_test <- predict(trainimp, df\_test)  
  
summary(df\_train)

## age male tb alkphos   
## Min. :-2.52549 Min. :-1.7656 Min. :-0.4459 Min. :-0.95942   
## 1st Qu.:-0.71998 1st Qu.: 0.5652 1st Qu.:-0.3803 1st Qu.:-0.46045   
## Median : 0.02712 Median : 0.5652 Median :-0.3475 Median :-0.32653   
## Mean : 0.00000 Mean : 0.0000 Mean : 0.0000 Mean : 0.00000   
## 3rd Qu.: 0.71197 3rd Qu.: 0.5652 3rd Qu.:-0.1179 3rd Qu.: 0.04931   
## Max. : 2.82876 Max. : 0.5652 Max. :11.7886 Max. : 7.88367   
## sgpt tp ag\_ratio   
## Min. :-0.3918 Min. :-3.4324 Min. :-2.060950   
## 1st Qu.:-0.3182 1st Qu.:-0.6129 1st Qu.:-0.785196   
## Median :-0.2535 Median : 0.1148 Median : 0.171619   
## Mean : 0.0000 Mean : 0.0000 Mean : 0.002034   
## 3rd Qu.:-0.1107 3rd Qu.: 0.6605 3rd Qu.: 0.490558   
## Max. : 9.6502 Max. : 2.8433 Max. : 4.955697

summary(df\_test)

## age male tb alkphos   
## Min. :-2.52549 Min. :-1.765557 Min. :-0.42954 Min. :-0.9076   
## 1st Qu.:-0.62660 1st Qu.: 0.565178 1st Qu.:-0.38034 1st Qu.:-0.4993   
## Median : 0.08938 Median : 0.565178 Median :-0.35574 Median :-0.3633   
## Mean : 0.04591 Mean : 0.002587 Mean : 0.15154 Mean : 0.1229   
## 3rd Qu.: 0.96100 3rd Qu.: 0.565178 3rd Qu.: 0.01327 3rd Qu.: 0.1076   
## Max. : 1.89488 Max. : 0.565178 Max. : 4.83493 Max. : 6.9592   
## sgpt tp ag\_ratio   
## Min. :-0.39181 Min. :-2.61379 Min. :-2.06095   
## 1st Qu.:-0.29711 1st Qu.:-0.70380 1st Qu.:-0.78520   
## Median :-0.19338 Median : 0.02381 Median :-0.14732   
## Mean : 0.17094 Mean : 0.04028 Mean : 0.01517   
## 3rd Qu.: 0.06068 3rd Qu.: 0.66047 3rd Qu.: 0.49056   
## Max. :11.57446 Max. : 2.75237 Max. : 5.91251

## Set training control for model building  
ctrl <- trainControl(method = "repeatedcv", 10,   
 repeats = 10,   
 summaryFunction = twoClassSummary,   
 classProbs = TRUE,   
 savePredictions = TRUE)

KNN Model

set.seed(476)  
knnFit <- train(x = df\_train,   
 y = selector\_train,  
 method = "knn",  
 metric = "AUC",  
 trControl = ctrl)  
knnFit

## k-Nearest Neighbors   
##   
## 466 samples  
## 7 predictor  
## 2 classes: 'NonPatient', 'Patient'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 420, 419, 419, 420, 419, 420, ...   
## Resampling results across tuning parameters:  
##   
## k ROC Sens Spec   
## 5 0.6329499 0.3048352 0.7867380  
## 7 0.6403834 0.2614835 0.8064439  
## 9 0.6662756 0.2909890 0.8230303  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was k = 9.

knnFit$finalModel

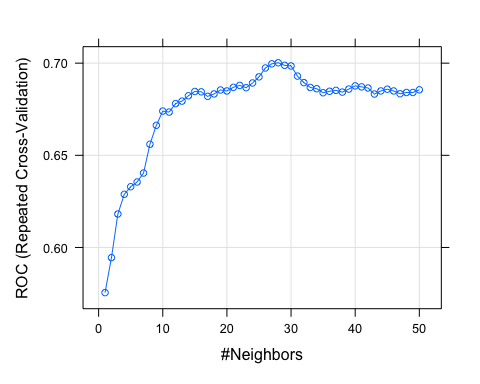
## 9-nearest neighbor model  
## Training set outcome distribution:  
##   
## NonPatient Patient   
## 134 332

## Validation and testing (model tuning and evaluation)

# we will make a grid of values to test in cross-validation.  
knnGrid <- expand.grid(k = c(1:50))  
set.seed(476)  
knnFitTune <- train(x = df\_train,   
 y = selector\_train,  
 method = "knn",  
 metric = "ROC",  
 tuneGrid = knnGrid,  
 trControl = ctrl)  
knnFitTune

## k-Nearest Neighbors   
##   
## 466 samples  
## 7 predictor  
## 2 classes: 'NonPatient', 'Patient'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 420, 419, 419, 420, 419, 420, ...   
## Resampling results across tuning parameters:  
##   
## k ROC Sens Spec   
## 1 0.5755000 0.40906593 0.7419340  
## 2 0.5944783 0.39978022 0.7163102  
## 3 0.6181437 0.41390110 0.7713815  
## 4 0.6288478 0.38120879 0.7644742  
## 5 0.6329499 0.30483516 0.7864349  
## 6 0.6355271 0.29901099 0.7863904  
## 7 0.6403834 0.26148352 0.8058467  
## 8 0.6560074 0.28120879 0.8113280  
## 9 0.6662756 0.29098901 0.8230303  
## 10 0.6739983 0.30796703 0.8153476  
## 11 0.6734586 0.30236264 0.8222282  
## 12 0.6780437 0.31049451 0.8361141  
## 13 0.6794064 0.29390110 0.8400178  
## 14 0.6823173 0.26747253 0.8447772  
## 15 0.6846299 0.24769231 0.8556684  
## 16 0.6844464 0.24917582 0.8604813  
## 17 0.6819883 0.24917582 0.8734403  
## 18 0.6833108 0.23747253 0.8766756  
## 19 0.6854432 0.21769231 0.8951515  
## 20 0.6849174 0.22390110 0.8957576  
## 21 0.6868711 0.20961538 0.9108111  
## 22 0.6879406 0.20587912 0.9077986  
## 23 0.6867014 0.18659341 0.9243494  
## 24 0.6892487 0.18818681 0.9282709  
## 25 0.6925643 0.18000000 0.9351961  
## 26 0.6973733 0.17939560 0.9364171  
## 27 0.6996364 0.17252747 0.9372995  
## 28 0.7002130 0.16346154 0.9355169  
## 29 0.6988232 0.15769231 0.9406239  
## 30 0.6984823 0.15071429 0.9418449  
## 31 0.6929935 0.14692308 0.9487701  
## 32 0.6894410 0.13494505 0.9433512  
## 33 0.6868282 0.12670330 0.9490285  
## 34 0.6861279 0.12142857 0.9466221  
## 35 0.6839485 0.11549451 0.9487522  
## 36 0.6847417 0.11357143 0.9457308  
## 37 0.6852390 0.10423077 0.9508734  
## 38 0.6842874 0.10186813 0.9505882  
## 39 0.6859330 0.09390110 0.9535740  
## 40 0.6876795 0.09263736 0.9532709  
## 41 0.6871371 0.08582418 0.9566043  
## 42 0.6865225 0.08060440 0.9578164  
## 43 0.6832550 0.07906593 0.9662121  
## 44 0.6849201 0.07071429 0.9716667  
## 45 0.6858751 0.06346154 0.9755437  
## 46 0.6849128 0.06274725 0.9713012  
## 47 0.6833960 0.05840659 0.9758645  
## 48 0.6840811 0.05027473 0.9773708  
## 49 0.6840945 0.03884615 0.9788859  
## 50 0.6855471 0.03505495 0.9764973  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was k = 28.

plot(knnFitTune)



knnFitTune$finalModel

## 28-nearest neighbor model  
## Training set outcome distribution:  
##   
## NonPatient Patient   
## 134 332

## Results and final model selection (performance measures, etc.)

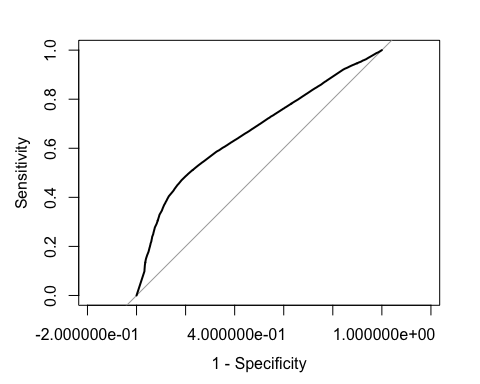
confusionMatrix(knnFitTune, norm = "none", mode="everything")

## Cross-Validated (10 fold, repeated 10 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction NonPatient Patient  
## NonPatient 219 214  
## Patient 1121 3106  
##   
## Accuracy (average) : 0.7135

## Plot the ROC curve for the hold-out set  
knnRoc <- roc(response = knnFitTune$pred$obs,  
 predictor = knnFitTune$pred$Patient,  
 levels = levels(knnFitTune$pred$obs))

## Setting direction: controls < cases

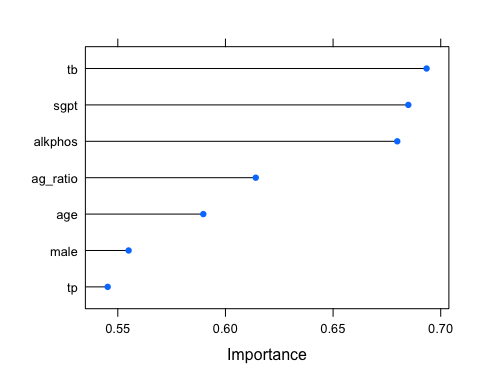
plot(knnRoc, legacy.axes = TRUE)



knnRoc$auc

## Area under the curve: 0.6673

knnImp <- varImp(knnFitTune, scale = FALSE)  
plot(knnImp)

 ## Evaluating the Model

predictions <- predict(knnFitTune,df\_test)  
confusionMatrix(predictions,selector\_test, mode = "prec\_recall")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonPatient Patient  
## NonPatient 4 5  
## Patient 29 78  
##   
## Accuracy : 0.7069   
## 95% CI : (0.6152, 0.7877)  
## No Information Rate : 0.7155   
## P-Value [Acc > NIR] : 0.6262   
##   
## Kappa : 0.0781   
##   
## Mcnemar's Test P-Value : 7.998e-05   
##   
## Precision : 0.44444   
## Recall : 0.12121   
## F1 : 0.19048   
## Prevalence : 0.28448   
## Detection Rate : 0.03448   
## Detection Prevalence : 0.07759   
## Balanced Accuracy : 0.53049   
##   
## 'Positive' Class : NonPatient   
##

## Score Test Data

# let's score the test set using this model  
pred\_class <- predict(knnFitTune, df\_test,'raw')  
probs <- predict(knnFitTune, df\_test,'prob')  
# Merge original scaled test dataset with predicions  
df\_test.scored <- cbind(df[-training,], pred\_class, probs)  
glimpse(df\_test.scored)

## Rows: 116  
## Columns: 10  
## $ age <int> 62, 57, 64, 25, 38, 40, 40, 51, 62, 38, 35, 45, 64, 58, 57,…  
## $ male <dbl> 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1,…  
## $ tb <dbl> 10.9, 0.6, 0.9, 0.6, 1.8, 0.9, 0.9, 2.9, 6.8, 2.6, 1.8, 0.7…  
## $ alkphos <int> 699, 210, 310, 183, 342, 293, 293, 482, 542, 410, 275, 170,…  
## $ sgpt <int> 64, 51, 61, 91, 168, 232, 232, 22, 116, 59, 48, 21, 20, 37,…  
## $ tp <dbl> 7.5, 5.9, 7.0, 5.5, 7.6, 6.8, 6.8, 7.0, 6.4, 5.6, 6.5, 5.7,…  
## $ ag\_ratio <dbl> 0.74, 0.80, 0.90, 0.70, 1.30, 0.80, 0.80, 0.50, 0.90, 0.80,…  
## $ pred\_class <fct> Patient, Patient, Patient, Patient, Patient, Patient, Patie…  
## $ NonPatient <dbl> 0.07142857, 0.32142857, 0.21428571, 0.39285714, 0.35714286,…  
## $ Patient <dbl> 0.9285714, 0.6785714, 0.7857143, 0.6071429, 0.6428571, 0.67…

Logistic Regression Model

set.seed(476)  
lrFit <- train(x = df\_train,  
 y = selector\_train,  
 method = "glm",  
 metric = "AUC",  
 trControl = ctrl)  
lrFit

## Generalized Linear Model   
##   
## 466 samples  
## 7 predictor  
## 2 classes: 'NonPatient', 'Patient'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 420, 419, 419, 420, 419, 420, ...   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.751961 0.2248901 0.9117914

lrFit$finalModel

##   
## Call: NULL  
##   
## Coefficients:  
## (Intercept) age male tb alkphos sgpt   
## 1.815176 0.294742 0.084564 1.636903 0.259526 2.496223   
## tp ag\_ratio   
## 0.001187 -0.174348   
##   
## Degrees of Freedom: 465 Total (i.e. Null); 458 Residual  
## Null Deviance: 559.2   
## Residual Deviance: 468.1 AIC: 484.1

lrCM <- confusionMatrix(lrFit, norm = "none", mode="everything")  
lrCM

## Cross-Validated (10 fold, repeated 10 times) Confusion Matrix   
##   
## (entries are un-normalized aggregated counts)  
##   
## Reference  
## Prediction NonPatient Patient  
## NonPatient 301 293  
## Patient 1039 3027  
##   
## Accuracy (average) : 0.7142

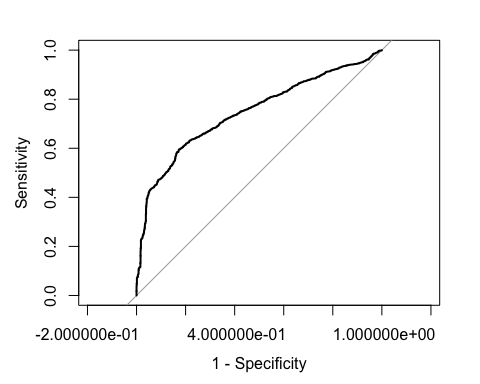
?confusionMatrix

## Help on topic 'confusionMatrix' was found in the following packages:  
##   
## Package Library  
## ModelMetrics /Library/Frameworks/R.framework/Versions/4.0/Resources/library  
## caret /Library/Frameworks/R.framework/Versions/4.0/Resources/library  
##   
##   
## Using the first match ...

## Plot the ROC curve for the hold-out set  
lrRoc <- roc(response = lrFit$pred$obs,  
 predictor = lrFit$pred$Patient,  
 levels = levels(lrFit$pred$obs))

## Setting direction: controls < cases

plot(lrRoc, legacy.axes = TRUE)



lrRoc$auc

## Area under the curve: 0.7462

predictions <- predict(lrFit,df\_test)  
confusionMatrix(predictions,selector\_test, mode = "prec\_recall")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonPatient Patient  
## NonPatient 5 5  
## Patient 28 78  
##   
## Accuracy : 0.7155   
## 95% CI : (0.6243, 0.7954)  
## No Information Rate : 0.7155   
## P-Value [Acc > NIR] : 0.5467676   
##   
## Kappa : 0.1155   
##   
## Mcnemar's Test P-Value : 0.0001283   
##   
## Precision : 0.50000   
## Recall : 0.15152   
## F1 : 0.23256   
## Prevalence : 0.28448   
## Detection Rate : 0.04310   
## Detection Prevalence : 0.08621   
## Balanced Accuracy : 0.54564   
##   
## 'Positive' Class : NonPatient   
##

## LDA Model

set.seed(476)  
ldaFit <- train(x = df\_train,  
 y = selector\_train,  
 method = "lda",  
 preProc = c("center","scale"),  
 metric = "AUC",  
 trControl = ctrl  
 )  
ldaFit

## Linear Discriminant Analysis   
##   
## 466 samples  
## 7 predictor  
## 2 classes: 'NonPatient', 'Patient'   
##   
## Pre-processing: centered (7), scaled (7)   
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 420, 419, 419, 420, 419, 420, ...   
## Resampling results:  
##   
## ROC Sens Spec   
## 0.7199124 0.07489011 0.9500267

ldaFit$finalModel

## Call:  
## lda(x, y)  
##   
## Prior probabilities of groups:  
## NonPatient Patient   
## 0.2875536 0.7124464   
##   
## Group means:  
## age male tb alkphos sgpt tp  
## NonPatient -0.22470119 -0.18274418 -0.3220793 -0.2711136 -0.2579519 0.1208720  
## Patient 0.09069265 0.07375819 0.1299959 0.1094254 0.1041131 -0.0487857  
## ag\_ratio  
## NonPatient 0.2479213  
## Patient -0.1000646  
##   
## Coefficients of linear discriminants:  
## LD1  
## age 0.37174274  
## male 0.26788405  
## tb 0.44035526  
## alkphos 0.34730926  
## sgpt 0.43641416  
## tp -0.04163225  
## ag\_ratio -0.26024522

## Evaluating the Model

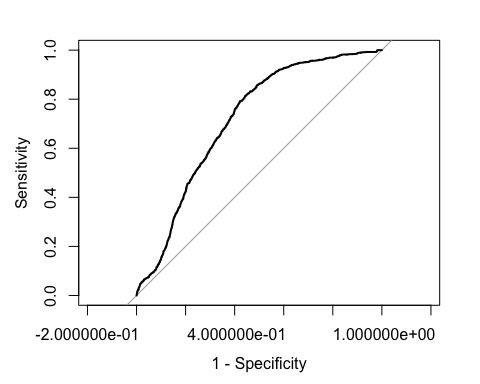
predictions <- predict(ldaFit,df\_test)  
confusionMatrix(predictions,selector\_test, mode = "prec\_recall")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonPatient Patient  
## NonPatient 2 3  
## Patient 31 80  
##   
## Accuracy : 0.7069   
## 95% CI : (0.6152, 0.7877)  
## No Information Rate : 0.7155   
## P-Value [Acc > NIR] : 0.6262   
##   
## Kappa : 0.0329   
##   
## Mcnemar's Test P-Value : 3.649e-06   
##   
## Precision : 0.40000   
## Recall : 0.06061   
## F1 : 0.10526   
## Prevalence : 0.28448   
## Detection Rate : 0.01724   
## Detection Prevalence : 0.04310   
## Balanced Accuracy : 0.51223   
##   
## 'Positive' Class : NonPatient   
##

ldaRoc <- roc(response = ldaFit$pred$obs,  
 predictor = ldaFit$pred$Patient,  
 levels = rev(levels(ldaFit$pred$obs)))

## Setting direction: controls > cases

plot(ldaRoc,legacy.axes = TRUE)



# Random Forest - Bagging

set.seed(476)  
rfFit <- train(x = df\_train,   
 y = selector\_train,  
 method = "rf",  
 metric = "AUC",  
 trControl = ctrl)

## Warning in train.default(x = df\_train, y = selector\_train, method = "rf", : The  
## metric "AUC" was not in the result set. ROC will be used instead.

rfFit

## Random Forest   
##   
## 466 samples  
## 7 predictor  
## 2 classes: 'NonPatient', 'Patient'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 420, 419, 419, 420, 419, 420, ...   
## Resampling results across tuning parameters:  
##   
## mtry ROC Sens Spec   
## 2 0.7280240 0.3073077 0.8681551  
## 4 0.7174376 0.3306593 0.8603298  
## 7 0.7088823 0.3319780 0.8531373  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 2.

rfFit$finalModel

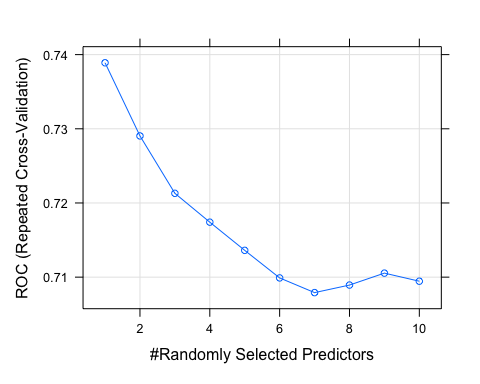
##   
## Call:  
## randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 2  
##   
## OOB estimate of error rate: 29.4%  
## Confusion matrix:  
## NonPatient Patient class.error  
## NonPatient 41 93 0.6940299  
## Patient 44 288 0.1325301

## Validation and testing (model tuning and evaluation)

# we will make a grid of values to test in cross-validation.  
rfGrid <- expand.grid(.mtry = c(1:10))  
set.seed(476)  
rfFitTune <- train(x = df\_train,   
 y = selector\_train,  
 method = "rf",  
 metric = "ROC",  
 tuneGrid = rfGrid,  
 trControl = ctrl)  
rfFitTune

## Random Forest   
##   
## 466 samples  
## 7 predictor  
## 2 classes: 'NonPatient', 'Patient'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 420, 419, 419, 420, 419, 420, ...   
## Resampling results across tuning parameters:  
##   
## mtry ROC Sens Spec   
## 1 0.7389010 0.1392857 0.9409537  
## 2 0.7290389 0.3103297 0.8708824  
## 3 0.7213005 0.3194505 0.8627540  
## 4 0.7174099 0.3316484 0.8598039  
## 5 0.7136120 0.3268681 0.8552228  
## 6 0.7098946 0.3354945 0.8588770  
## 7 0.7079140 0.3385714 0.8537166  
## 8 0.7089338 0.3324176 0.8516132  
## 9 0.7105413 0.3318681 0.8534581  
## 10 0.7094510 0.3350549 0.8525223  
##   
## ROC was used to select the optimal model using the largest value.  
## The final value used for the model was mtry = 1.

plot(rfFitTune)



rfFitTune$finalModel

##   
## Call:  
## randomForest(x = x, y = y, mtry = min(param$mtry, ncol(x)))   
## Type of random forest: classification  
## Number of trees: 500  
## No. of variables tried at each split: 1  
##   
## OOB estimate of error rate: 28.97%  
## Confusion matrix:  
## NonPatient Patient class.error  
## NonPatient 17 117 0.87313433  
## Patient 18 314 0.05421687

## Evaluating the Model

prediction <- predict(rfFitTune,df\_test)  
confusionMatrix(prediction,selector\_test, mode = "prec\_recall")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonPatient Patient  
## NonPatient 8 8  
## Patient 25 75  
##   
## Accuracy : 0.7155   
## 95% CI : (0.6243, 0.7954)  
## No Information Rate : 0.7155   
## P-Value [Acc > NIR] : 0.546768   
##   
## Kappa : 0.1729   
##   
## Mcnemar's Test P-Value : 0.005349   
##   
## Precision : 0.50000   
## Recall : 0.24242   
## F1 : 0.32653   
## Prevalence : 0.28448   
## Detection Rate : 0.06897   
## Detection Prevalence : 0.13793   
## Balanced Accuracy : 0.57302   
##   
## 'Positive' Class : NonPatient   
##

C5.0 - Boosting

set.seed(476)  
CFit <- train(x = df\_train,   
 y = selector\_train,  
 method = "C5.0",  
 metric = "AUC",  
 trControl = ctrl)  
CFit

## C5.0   
##   
## 466 samples  
## 7 predictor  
## 2 classes: 'NonPatient', 'Patient'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 420, 419, 419, 420, 419, 420, ...   
## Resampling results across tuning parameters:  
##   
## model winnow trials ROC Sens Spec   
## rules FALSE 1 0.6394617 0.3892308 0.8335472  
## rules FALSE 10 0.6929753 0.3959341 0.8300357  
## rules FALSE 20 0.6932317 0.3967033 0.8297326  
## rules TRUE 1 0.5465353 0.1400549 0.9196791  
## rules TRUE 10 0.5637949 0.1550000 0.9156774  
## rules TRUE 20 0.5640513 0.1557692 0.9153743  
## tree FALSE 1 0.6940037 0.3802747 0.8322549  
## tree FALSE 10 0.7085382 0.4003297 0.8237701  
## tree FALSE 20 0.7086315 0.4010989 0.8228610  
## tree TRUE 1 0.5792239 0.1461538 0.9163458  
## tree TRUE 10 0.5807037 0.1550000 0.9128877  
## tree TRUE 20 0.5807037 0.1550000 0.9128877  
##   
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were trials = 20, model = tree and winnow  
## = FALSE.

CFit$finalModel

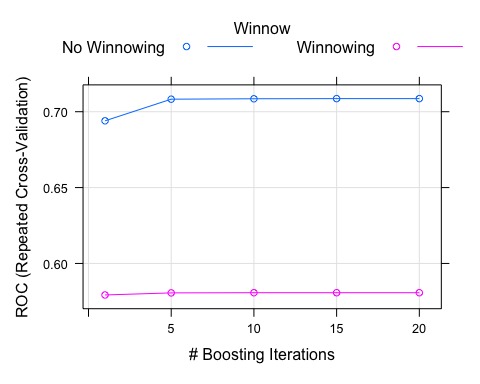
##   
## Call:  
## (function (x, y, trials = 1, rules = FALSE, weights = NULL, control  
## 2, fuzzyThreshold = FALSE, sample = 0, earlyStopping = TRUE, label  
## = "outcome", seed = 1596L))  
##   
## Classification Tree  
## Number of samples: 466   
## Number of predictors: 7   
##   
## Number of boosting iterations: 20 requested; 4 used due to early stopping  
## Average tree size: 6.5   
##   
## Non-standard options: attempt to group attributes

## Validation and testing (model tuning and evaluation)

# we will make a grid of values to test in cross-validation.  
CGrid <- expand.grid(.trials= c(1,5,10,15,20), .winnow= c(TRUE,FALSE), .model= "tree")  
set.seed(476)  
CFitTune <- train(x = df\_train,   
 y = selector\_train,  
 method = "C5.0",  
 metric = "AUC",  
 tuneGrid = CGrid,  
 trControl = ctrl)  
CFitTune

## C5.0   
##   
## 466 samples  
## 7 predictor  
## 2 classes: 'NonPatient', 'Patient'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold, repeated 10 times)   
## Summary of sample sizes: 420, 419, 419, 420, 419, 420, ...   
## Resampling results across tuning parameters:  
##   
## winnow trials ROC Sens Spec   
## FALSE 1 0.6940037 0.3802747 0.8322549  
## FALSE 5 0.7082583 0.4091209 0.8232888  
## FALSE 10 0.7085382 0.4003297 0.8237701  
## FALSE 15 0.7086315 0.4010989 0.8228610  
## FALSE 20 0.7086315 0.4010989 0.8228610  
## TRUE 1 0.5792239 0.1461538 0.9163458  
## TRUE 5 0.5805905 0.1595055 0.9120143  
## TRUE 10 0.5807037 0.1550000 0.9128877  
## TRUE 15 0.5807037 0.1550000 0.9128877  
## TRUE 20 0.5807037 0.1550000 0.9128877  
##   
## Tuning parameter 'model' was held constant at a value of tree  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were trials = 15, model = tree and winnow  
## = FALSE.

plot(CFitTune)



CFitTune$finalModel

##   
## Call:  
## (function (x, y, trials = 1, rules = FALSE, weights = NULL, control  
## 2, fuzzyThreshold = FALSE, sample = 0, earlyStopping = TRUE, label  
## = "outcome", seed = 1597L))  
##   
## Classification Tree  
## Number of samples: 466   
## Number of predictors: 7   
##   
## Number of boosting iterations: 15 requested; 4 used due to early stopping  
## Average tree size: 6.5   
##   
## Non-standard options: attempt to group attributes

## Evaluating the Model

predictions <- predict(CFitTune,df\_test)  
confusionMatrix(predictions,selector\_test, mode = "prec\_recall")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonPatient Patient  
## NonPatient 14 14  
## Patient 19 69  
##   
## Accuracy : 0.7155   
## 95% CI : (0.6243, 0.7954)  
## No Information Rate : 0.7155   
## P-Value [Acc > NIR] : 0.5468   
##   
## Kappa : 0.2678   
##   
## Mcnemar's Test P-Value : 0.4862   
##   
## Precision : 0.5000   
## Recall : 0.4242   
## F1 : 0.4590   
## Prevalence : 0.2845   
## Detection Rate : 0.1207   
## Detection Prevalence : 0.2414   
## Balanced Accuracy : 0.6278   
##   
## 'Positive' Class : NonPatient   
##