Cleaning and Classifications

Madeline Chang Lorena Dorado Marvin Moran

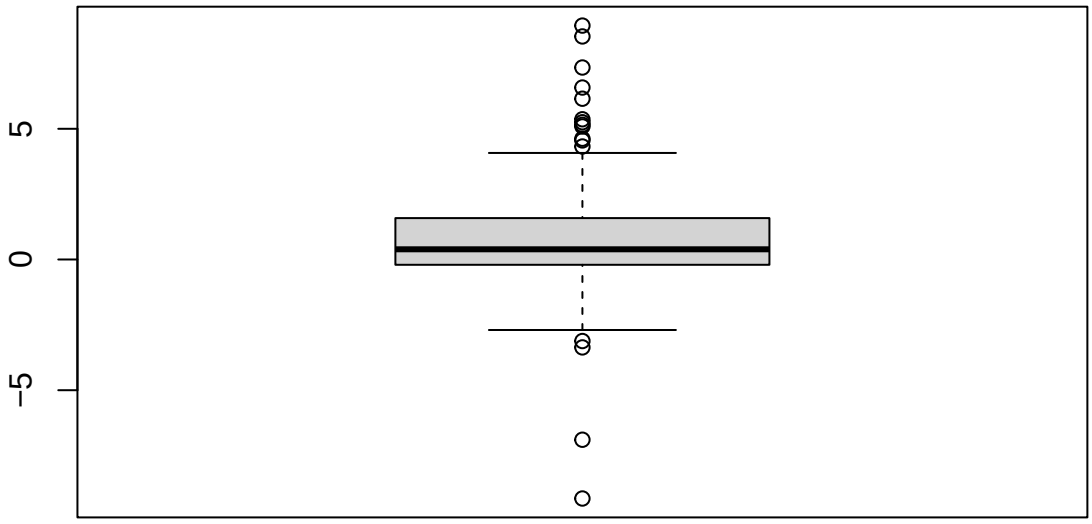
6/12/2024

```
library(AppliedPredictiveModeling)
library(e1071)
library(caret)
library(rpart)
library(rpart.plot)
library(partykit)
library(earth)
library(kernlab)
library(mlbench)
library(randomForest)
library(dplyr)
library(corrplot)
library(pROC)
library(RANN)
library(glmnet)
library(ggplot2)
library(igraph)
library(tidyverse)
library(gt)
library(RCurl)
```

```
# Read data
url <- "https://github.com/renaqd/team3-ADS503/raw/main/data.csv"
changer_data<- read.csv(url, header = TRUE) %>%
  mutate(Class = as.factor(Class))
```

```
# Check for and remove zero variance predictors
degen<- nearZeroVar(changer_data)
changer_new<- changer_data[,-degen]

# Remove highly correlated variables of 75%
corr<- cor(changer_new[,1:1094])
high_corr <- findCorrelation(corr, cutoff = 0.75)
changer_new<- changer_new[,-high_corr]

# Create vector to store 216 values
# For each column of predictors (1 to 216), calculate skew and store in skewed
skewed<- length(216)
for(i in 1:216){
  skewed[i]<- skewness(changer_new[,i], na.rm = TRUE, type = 1)
}
```

```r
# See where most of the data falls in range
boxplot(skewed)
```




```r
# Group data by class
changer_new %>%
  group_by(Class) %>%
  summarise(n = n()) %>%
  gt::gt()
```

Class	n
Changer	27
NoChanger	63

```r
#write.csv(changer_new, "changer_new.csv", row.names = FALSE)
```

Exploratory Data Analysis

```r
# data summary
# summary(changer_new)
```

```r
######################## suggestion ##########################
######## sum(is.na.data.frame(changer_new)) ################
#############################################################


# missing values
missing_values <- sapply(changer_new, function(x) sum(is.na(x)))

if(any(missing_values > 0)) {
  print(missing_values[missing_values > 0])
} else {
  print("No features found with missing values")
}
```

```
## [1] "No features found with missing values"
```

```
# identify numerical features
numerical_features <- changer_new[, sapply(changer_new, is.numeric)]

# print distribution for numerical features to PDF <<-- already done and found in GitHub
pdf("numerical_distributions.pdf")
lapply(names(numerical_features), function(feature) {
    ggplot(changer_new, aes_string(x = feature)) +
        geom_histogram(bins = 30, fill = "blue", color = "black") +
        ggtitle(paste("Distribution of", feature))
})
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## [[1]]

##
## [[2]]

##
## [[3]]

##
## [[4]]

##
## [[5]]

##
## [[6]]

##
## [[7]]

##
## [[8]]

##
## [[9]]

##
## [[10]]

##
## [[11]]
```

```
##
## [[12]]

##
## [[13]]

##
## [[14]]

##
## [[15]]

##
## [[16]]

##
## [[17]]

##
## [[18]]

##
## [[19]]

##
## [[20]]

##
## [[21]]

##
## [[22]]

##
## [[23]]

##
## [[24]]

##
## [[25]]

##
## [[26]]

##
## [[27]]

##
## [[28]]
```

```
##
## [[29]]

##
## [[30]]

##
## [[31]]

##
## [[32]]

##
## [[33]]

##
## [[34]]

##
## [[35]]

##
## [[36]]

##
## [[37]]

##
## [[38]]

##
## [[39]]

##
## [[40]]

##
## [[41]]

##
## [[42]]

##
## [[43]]

##
## [[44]]

##
## [[45]]
```

```
##
## [[46]]

##
## [[47]]

##
## [[48]]

##
## [[49]]

##
## [[50]]

##
## [[51]]

##
## [[52]]

##
## [[53]]

##
## [[54]]

##
## [[55]]

##
## [[56]]

##
## [[57]]

##
## [[58]]

##
## [[59]]

##
## [[60]]

##
## [[61]]

##
## [[62]]
```

```
##
## [[63]]

##
## [[64]]

##
## [[65]]

##
## [[66]]

##
## [[67]]

##
## [[68]]

##
## [[69]]

##
## [[70]]

##
## [[71]]

##
## [[72]]

##
## [[73]]

##
## [[74]]

##
## [[75]]

##
## [[76]]

##
## [[77]]

##
## [[78]]

##
## [[79]]
```

```
##
## [[80]]

##
## [[81]]

##
## [[82]]

##
## [[83]]

##
## [[84]]

##
## [[85]]

##
## [[86]]

##
## [[87]]

##
## [[88]]

##
## [[89]]

##
## [[90]]

##
## [[91]]

##
## [[92]]

##
## [[93]]

##
## [[94]]

##
## [[95]]

##
## [[96]]
```

```
##
## [[97]]

##
## [[98]]

##
## [[99]]

##
## [[100]]

##
## [[101]]

##
## [[102]]

##
## [[103]]

##
## [[104]]

##
## [[105]]

##
## [[106]]

##
## [[107]]

##
## [[108]]

##
## [[109]]

##
## [[110]]

##
## [[111]]

##
## [[112]]

##
## [[113]]
```

```
##
## [[114]]

##
## [[115]]

##
## [[116]]

##
## [[117]]

##
## [[118]]

##
## [[119]]

##
## [[120]]

##
## [[121]]

##
## [[122]]

##
## [[123]]

##
## [[124]]

##
## [[125]]

##
## [[126]]

##
## [[127]]

##
## [[128]]

##
## [[129]]

##
## [[130]]
```

```
##
## [[131]]

##
## [[132]]

##
## [[133]]

##
## [[134]]

##
## [[135]]

##
## [[136]]

##
## [[137]]

##
## [[138]]

##
## [[139]]

##
## [[140]]

##
## [[141]]

##
## [[142]]

##
## [[143]]

##
## [[144]]

##
## [[145]]

##
## [[146]]

##
## [[147]]
```

```
##
## [[148]]

##
## [[149]]

##
## [[150]]

##
## [[151]]

##
## [[152]]

##
## [[153]]

##
## [[154]]

##
## [[155]]

##
## [[156]]

##
## [[157]]

##
## [[158]]

##
## [[159]]

##
## [[160]]

##
## [[161]]

##
## [[162]]

##
## [[163]]

##
## [[164]]
```

```
##
## [[165]]

##
## [[166]]

##
## [[167]]

##
## [[168]]

##
## [[169]]

##
## [[170]]

##
## [[171]]

##
## [[172]]

##
## [[173]]

##
## [[174]]

##
## [[175]]

##
## [[176]]

##
## [[177]]

##
## [[178]]

##
## [[179]]

##
## [[180]]

##
## [[181]]
```

```
##
## [[182]]

##
## [[183]]

##
## [[184]]

##
## [[185]]

##
## [[186]]

##
## [[187]]

##
## [[188]]

##
## [[189]]

##
## [[190]]

##
## [[191]]

##
## [[192]]

##
## [[193]]

##
## [[194]]

##
## [[195]]

##
## [[196]]

##
## [[197]]

##
## [[198]]
```

```
##
## [[199]]

##
## [[200]]

##
## [[201]]

##
## [[202]]

##
## [[203]]

##
## [[204]]

##
## [[205]]

##
## [[206]]

##
## [[207]]

##
## [[208]]

##
## [[209]]

##
## [[210]]

##
## [[211]]

##
## [[212]]

##
## [[213]]

##
## [[214]]

##
## [[215]]

##
## [[216]]
```

```r
dev.off()
```

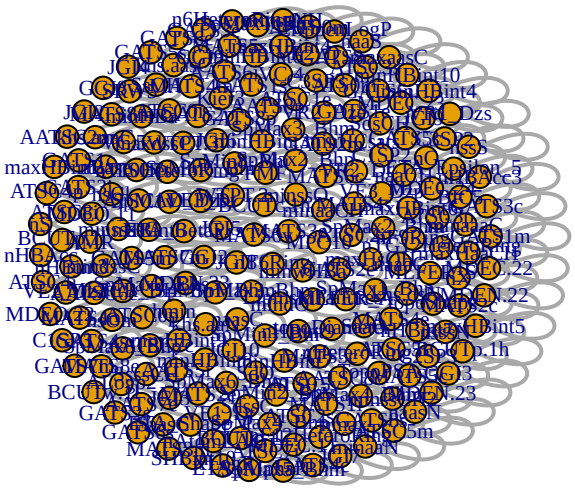
```
## pdf
##   2
```

```r
# corr matrix
cor_matrix <- cor(numerical_features, use = "complete.obs")

# plot correlation matrix
#png("correlation_matrix.png")
#corrplot(cor_matrix, method = "color", type = "upper", order = "hclust",
         #tl.col = "black", tl.srt = 45)
#dev.off()
```

```r
# set threshold for significant correlations
threshold <- 0.75

# create adjacency matrix where correlations above the threshold are marked
adjacency_matrix <- abs(cor_matrix) > threshold

# convert adjacency matrix to graph
graph <- graph.adjacency(adjacency_matrix, mode = "undirected", weighted = TRUE)
```

```
## Warning: ‘graph.adjacency()‘ was deprecated in igraph 2.0.0.
## i Please use ‘graph_from_adjacency_matrix()‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```r
E(graph)$weight <- cor_matrix[adjacency_matrix]

# plot the graph
plot(graph, vertex.size = 10, vertex.label.cex = 0.7, edge.width = E(graph)$weight * 2)
```



```r
# class distribution <<- code above shows numeric values only, this is a visual
class_distribution <- table(changer_new$Class)
barplot(class_distribution, main = "Class Distribution", col = "cyan",
        xlab = "Class", ylab = "Frequency")
```

Data Splitting

```r
set.seed(720)

# Create stratified random splits of the data (based on the classes)
trainingRows <- createDataPartition(changer_new$Class, p = .5, list = FALSE)

# Subset data into train and test
train <- changer_new[trainingRows, ]
test <- changer_new[-trainingRows, ]

# Create control method for cross validation
ctrl <- trainControl(method = "cv",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)
```

Model Building Strategies

```r
# Model method
class_function<- function(method){
  model<- train(x = train[,1:216],
                y = train$Class,
                method = method,
```

```
                preProc = c("center", "scale"),
                metric = "ROC",
                trControl = ctrl)
}

# Grid
class_grid<- function(method, grid){
  model<- train(x = train[,1:216],
                y = train$Class,
                method = method,
                preProc = c("center", "scale"),
                tuneGrid = grid,
                metric = "ROC",
                trControl = ctrl)
}

# MaxIt
class_nnet<- function(method, grid, maxit){
  model<- train(x = train[,1:216],
                y = train$Class,
                method = method,
                preProc = c("center", "scale"),
                tuneGrid = grid,
                maxit = maxit,
                trace = FALSE,
                metric = "ROC",
                trControl = ctrl)
}

# Length
class_length<- function(method, length){
  model<- train(x = train[,1:216],
                y = train$Class,
                method = method,
                preProc = c("center", "scale"),
                tuneLength = length,
                metric = "ROC",
                trControl = ctrl)
}

# Number of bags
class_bag<- function(method, nbagg){
  model<- train(x = train[,1:216],
                y = train$Class,
                method = method,
                preProc = c("center", "scale"),
                nbagg = nbagg,
                metric = "ROC",
                trControl = ctrl)
}

# Number of trees
class_rf<- function(method, grid, ntree){
```

```
  model<- train(x = train[,1:216],
                y = train$Class,
                method = method,
                preProc = c("center", "scale"),
                tuneGrid = grid,
                ntree = ntree,
                metric = "ROC",
                trControl = ctrl)
}
```

Hyper Parameter Tuning Defined in Grids

```
set.seed(720)

# TuneGrid for Penalized Logistic Regrssion
glmnGrid <- expand.grid(alpha = c(0,  .1,  .2, .4, .6, .8, 1),
                        lambda = seq(.01, .2, length = 10))

# NSC TuneGrid
nsc_grid<- data.frame(threshold = seq(0, 25, length = 30))

# mda_grid<- data.frame(subclasses=1)

# NNET TuneGrid ################################################################
nnet_grid <- expand.grid(size=1:4, decay=c(0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4))

# SVM TuneGrid
sigmaEst <- kernlab::sigest(as.matrix(train[,1:216]))
svmgrid <- expand.grid(sigma = sigmaEst, C = 2^seq(-4,+4))

# RF TuneGrid
mtryValues <- data.frame(mtry = seq(1,10,1))
```

Train Models

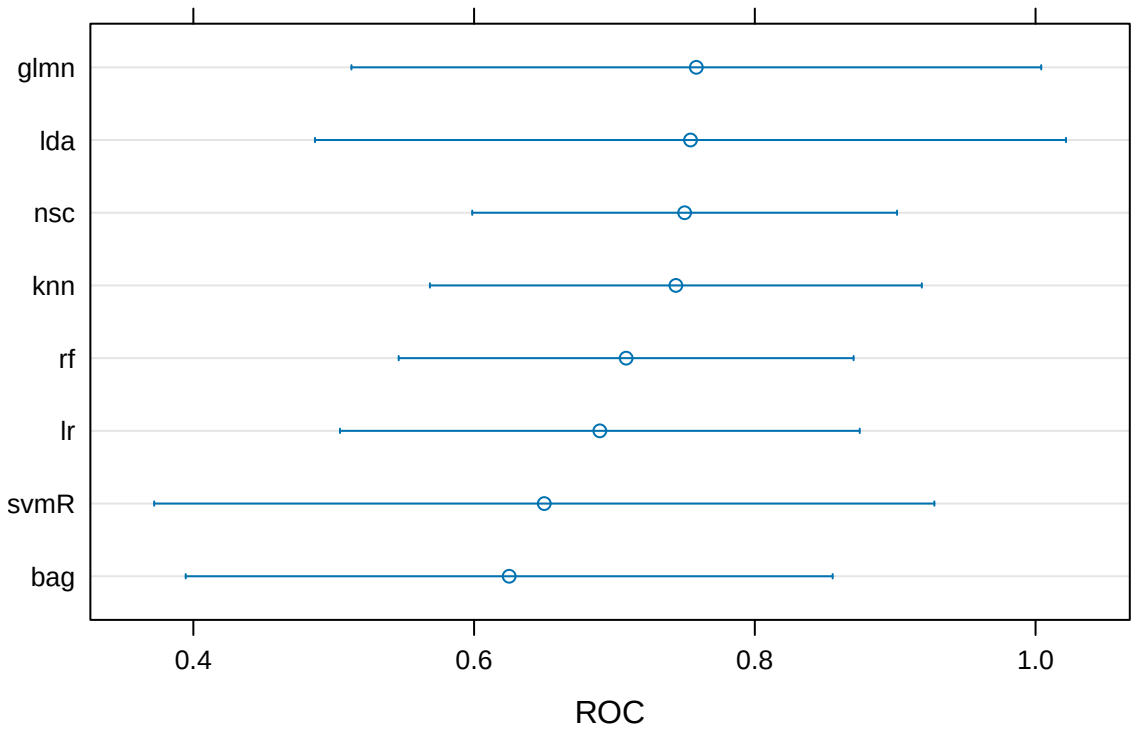
```
set.seed(720)

# Generalized Linear Model: Logistic Regression
lr<- class_function("glm")

# Linear Discriminant Analysis
lda<- class_function("lda")

# Penalized LR
glmn<- class_grid("glmnet", glmnGrid)

# Nearest Shrunken Centroids
nsc<- class_grid("pam", nsc_grid)
```

```
## 11111111111
```

```
# mda<- class_grid("mda", mda_grid)
svmR<- class_grid("svmRadial", svmgrid)

# NNET
nnet<- class_nnet("nnet", nnet_grid, 100)

# K Nearest Neighbors
knn<- class_length("knn", 20)

# Bagging
bag<- class_bag("treebag", 50)

# Random Forest
rf<- class_rf("rf", mtryValues, 100)
```

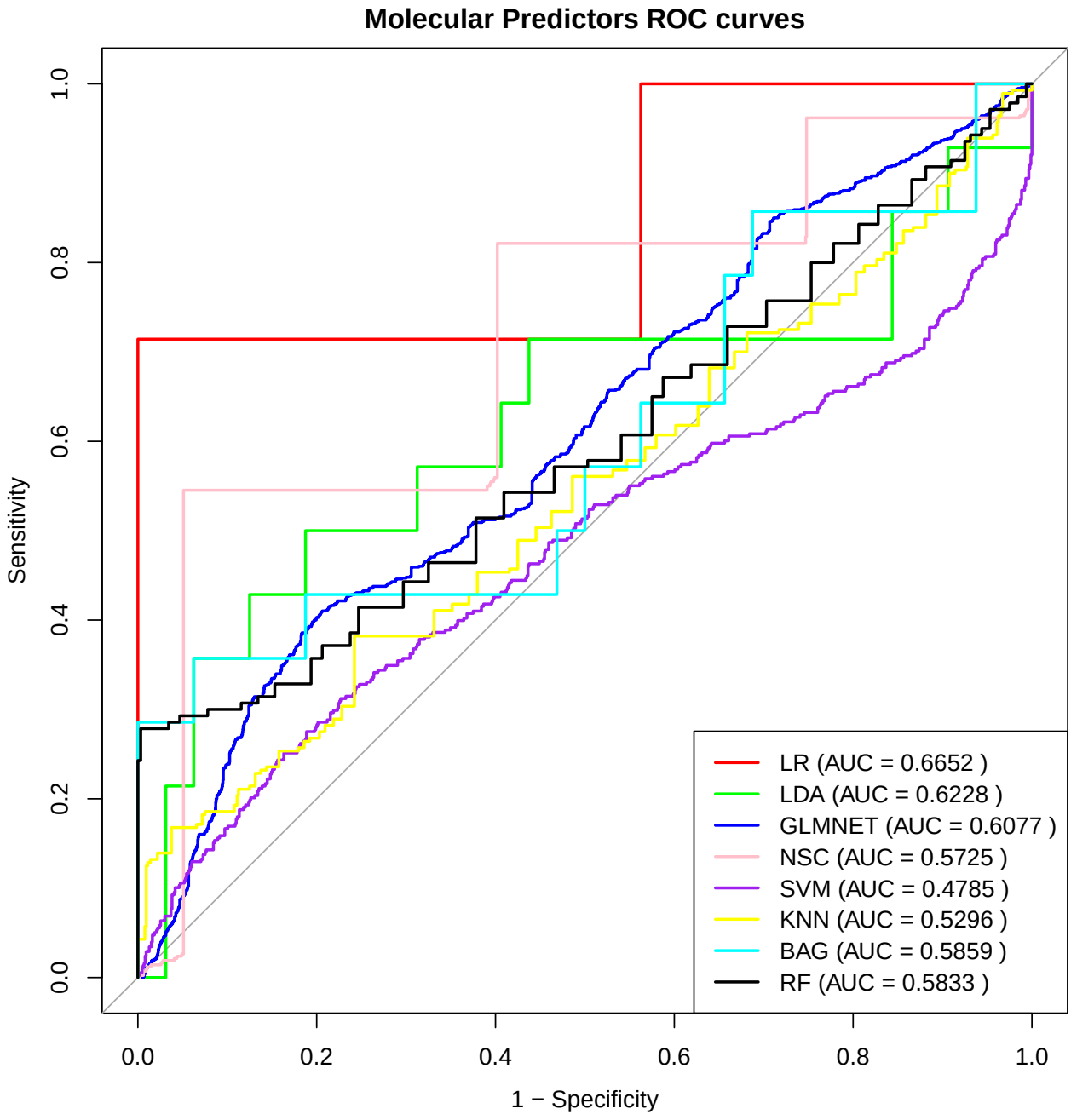
Model Performance

```
model_roc <- function(model){
  roc(response = model$pred$obs,
              predictor = model$pred$Changer,
              levels = rev(levels(model$pred$obs)))
}

lrROC <- model_roc(lr)
ldaROC <- model_roc(lda)
glmnROC <- model_roc(glmn)
nscROC <- model_roc(nsc)
svmRROC <- model_roc(svmR)
knnROC <- model_roc(knn)
bagROC <- model_roc(bag)
rfROC <- model_roc(rf)


train_re<- resamples(list(
  lr = lr,
  lda = lda,
  glmn = glmn,
  nsc = nsc,
  svmR = svmR,
  knn = knn,
  bag = bag,
  rf = rf
))

dotplot(train_re, metric = "ROC")
```



Confidence Level: 0.95

```
plot(lrROC, type = "s", col = 'red', legacy.axes = TRUE,
     main = "Molecular Predictors ROC curves")
plot(ldaROC, type = "s", add = TRUE, col = 'green', legacy.axes = TRUE)
plot(glmnROC, type = "s", add = TRUE, col = 'blue', legacy.axes = TRUE)
plot(nscROC, type = "s", add = TRUE, col = 'pink', legacy.axes = TRUE)
plot(svmRROC, type = "s", add = TRUE, col = 'purple', legacy.axes = TRUE)
plot(knnROC, type = "s", add = TRUE, col = 'yellow', legacy.axes = TRUE)
plot(bagROC, type = "s", add = TRUE, col = 'cyan', legacy.axes = TRUE)
plot(rfROC , type = "s", add = TRUE, legacy.axes = TRUE)
legend("bottomright", legend=c(paste("LR (AUC =", round(lrROC$auc,4), ")"),
                               paste("LDA (AUC =", round(ldaROC$auc,4), ")"),
                               paste("GLMNET (AUC =", round(glmnROC$auc,4), ")"),
                               paste("NSC (AUC =", round(nscROC$auc,4), ")"),
                               paste("SVM (AUC =", round(svmRROC$auc,4), ")"),
                               paste("KNN (AUC =", round(knnROC$auc,4), ")"),
                               paste("BAG (AUC =", round(bagROC$auc,4), ")"),
                               paste("RF (AUC =", round(rfROC $auc,4), ")")),
       col=c("red", "green", "blue", "pink", "purple", "yellow", "cyan", "black"), lwd=2)
```



Top Predictors

```
model_imp <- varImp(bag)
model_top10 <- model_imp$importance %>%
  as.data.frame() %>%
  rownames_to_column(var = 'Predictor') %>%
  arrange(desc(Overall)) %>%
  head(10) %>%
  select(Predictor, Importance = Overall)

model_top10 %>%
  gt()
```

Predictor	Importance
MATS3c	100.00000
GATS4s	70.49618
MATS4c	69.44122
MATS4e	59.61791
FMF	32.06564
GATS1m	30.43520
MATS4s	29.25486
ETA_BetaP_s	26.52950
GATS4c	23.45218
VE3_Dt	22.71387

```
# MATS4e, MATS4s match previous research
```