Feature Importance Exploration

# Import libraries

# Load Data

This block performs the following operations:

1. Loads the raw data.
2. Selects the desired predictor variables.
3. Drops rows with missing data.
4. Prepares the target variable, transforming it from 4 classes to 2.

set.seed(88)  
data\_csv <- here("data", "raw", "merged\_cmv.csv")  
df <- read.csv(data\_csv)  
  
df <- df %>%   
 select(-Sample) %>%   
 drop\_na() %>%   
 mutate(Cohort = factor(if\_else(Cohort %in% c("PP", "NP"), "Primary", "Latent")))

# Preprocessing

The following block uses the recipes library to log scale and normalize the predictors between 0 and 1.

rec\_obj <- recipe(Cohort ~ ., data = df)  
  
normalizer <- rec\_obj %>%   
 step\_log(all\_predictors()) %>%   
 step\_range(all\_predictors(), min = 0, max = 1)  
  
trained\_rec <- prep(normalizer, training = df)  
  
norm\_df <- bake(trained\_rec, df)

# Predicting

The following block performs 10-fold cross validation using lasso regression with the glmnet package. mod\_df removes all of the IgM predictors, as these outweigh every other predictor substantially.

myGrid = expand.grid(  
 alpha = 1,  
 lambda = seq(0.0001, 1, length = 100)  
)  
  
# mod\_df <- norm\_df  
mod\_df <- norm\_df[, -grep("^IgM", colnames(norm\_df))]  
  
model <- train(  
 Cohort ~ .,   
 mod\_df,   
 method = 'glmnet',  
 tuneGrid = myGrid,  
 trControl = trainControl(  
 method = "cv",  
 number = 10,  
 summaryFunction = twoClassSummary,  
 classProbs = TRUE,  
 verboseIter = TRUE  
 )  
)

## Warning in train.default(x, y, weights = w, ...): The metric "Accuracy" was  
## not in the result set. ROC will be used instead.

## + Fold01: alpha=1, lambda=1   
## - Fold01: alpha=1, lambda=1   
## + Fold02: alpha=1, lambda=1   
## - Fold02: alpha=1, lambda=1   
## + Fold03: alpha=1, lambda=1   
## - Fold03: alpha=1, lambda=1   
## + Fold04: alpha=1, lambda=1   
## - Fold04: alpha=1, lambda=1   
## + Fold05: alpha=1, lambda=1   
## - Fold05: alpha=1, lambda=1   
## + Fold06: alpha=1, lambda=1   
## - Fold06: alpha=1, lambda=1   
## + Fold07: alpha=1, lambda=1   
## - Fold07: alpha=1, lambda=1   
## + Fold08: alpha=1, lambda=1   
## - Fold08: alpha=1, lambda=1   
## + Fold09: alpha=1, lambda=1   
## - Fold09: alpha=1, lambda=1   
## + Fold10: alpha=1, lambda=1   
## - Fold10: alpha=1, lambda=1   
## Aggregating results  
## Selecting tuning parameters  
## Fitting alpha = 1, lambda = 0.0607 on full training set

model

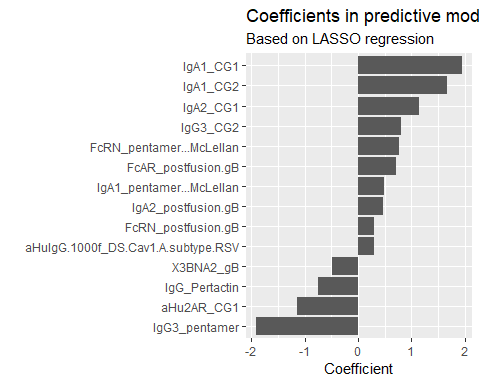
## glmnet   
##   
## 99 samples  
## 224 predictors  
## 2 classes: 'Latent', 'Primary'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 89, 90, 89, 90, 88, 89, ...   
## Resampling results across tuning parameters:  
##   
## lambda ROC Sens Spec   
## 0.0001 0.9440000 0.9433333 0.845  
## 0.0102 0.9440000 0.9633333 0.845  
## 0.0203 0.9480000 0.9633333 0.845  
## 0.0304 0.9486667 0.9633333 0.820  
## 0.0405 0.9526667 0.9433333 0.840  
## 0.0506 0.9566667 0.9433333 0.820  
## 0.0607 0.9566667 0.9433333 0.845  
## 0.0708 0.9476667 0.9433333 0.845  
## 0.0809 0.9476667 0.9433333 0.845  
## 0.0910 0.9526667 0.9433333 0.845  
## 0.1011 0.9485000 0.9433333 0.820  
## 0.1112 0.9531667 0.9433333 0.820  
## 0.1213 0.9490000 0.9433333 0.820  
## 0.1314 0.9490000 0.9433333 0.820  
## 0.1415 0.9490000 0.9433333 0.820  
## 0.1516 0.9450000 0.9433333 0.820  
## 0.1617 0.9450000 0.9433333 0.800  
## 0.1718 0.9410000 0.9433333 0.800  
## 0.1819 0.9410000 0.9433333 0.800  
## 0.1920 0.9400000 0.9433333 0.800  
## 0.2021 0.9400000 0.9433333 0.800  
## 0.2122 0.9318333 0.9433333 0.800  
## 0.2223 0.9236667 0.9433333 0.800  
## 0.2324 0.9186667 0.9233333 0.800  
## 0.2425 0.9146667 0.9233333 0.800  
## 0.2526 0.9055000 0.9233333 0.710  
## 0.2627 0.9055000 0.9233333 0.690  
## 0.2728 0.9015000 0.9233333 0.625  
## 0.2829 0.8975000 0.9233333 0.625  
## 0.2930 0.8933333 0.9233333 0.585  
## 0.3031 0.8883333 0.9233333 0.520  
## 0.3132 0.8883333 0.9233333 0.380  
## 0.3233 0.8883333 0.9633333 0.220  
## 0.3334 0.8853333 0.9833333 0.080  
## 0.3435 0.8821667 1.0000000 0.020  
## 0.3536 0.7455000 1.0000000 0.000  
## 0.3637 0.5941667 1.0000000 0.000  
## 0.3738 0.5000000 1.0000000 0.000  
## 0.3839 0.5000000 1.0000000 0.000  
## 0.3940 0.5000000 1.0000000 0.000  
## 0.4041 0.5000000 1.0000000 0.000  
## 0.4142 0.5000000 1.0000000 0.000  
## 0.4243 0.5000000 1.0000000 0.000  
## 0.4344 0.5000000 1.0000000 0.000  
## 0.4445 0.5000000 1.0000000 0.000  
## 0.4546 0.5000000 1.0000000 0.000  
## 0.4647 0.5000000 1.0000000 0.000  
## 0.4748 0.5000000 1.0000000 0.000  
## 0.4849 0.5000000 1.0000000 0.000  
## 0.4950 0.5000000 1.0000000 0.000  
## 0.5051 0.5000000 1.0000000 0.000  
## 0.5152 0.5000000 1.0000000 0.000  
## 0.5253 0.5000000 1.0000000 0.000  
## 0.5354 0.5000000 1.0000000 0.000  
## 0.5455 0.5000000 1.0000000 0.000  
## 0.5556 0.5000000 1.0000000 0.000  
## 0.5657 0.5000000 1.0000000 0.000  
## 0.5758 0.5000000 1.0000000 0.000  
## 0.5859 0.5000000 1.0000000 0.000  
## 0.5960 0.5000000 1.0000000 0.000  
## 0.6061 0.5000000 1.0000000 0.000  
## 0.6162 0.5000000 1.0000000 0.000  
## 0.6263 0.5000000 1.0000000 0.000  
## 0.6364 0.5000000 1.0000000 0.000  
## 0.6465 0.5000000 1.0000000 0.000  
## 0.6566 0.5000000 1.0000000 0.000  
## 0.6667 0.5000000 1.0000000 0.000  
## 0.6768 0.5000000 1.0000000 0.000  
## 0.6869 0.5000000 1.0000000 0.000  
## 0.6970 0.5000000 1.0000000 0.000  
## 0.7071 0.5000000 1.0000000 0.000  
## 0.7172 0.5000000 1.0000000 0.000  
## 0.7273 0.5000000 1.0000000 0.000  
## 0.7374 0.5000000 1.0000000 0.000  
## 0.7475 0.5000000 1.0000000 0.000  
## 0.7576 0.5000000 1.0000000 0.000  
## 0.7677 0.5000000 1.0000000 0.000  
## 0.7778 0.5000000 1.0000000 0.000  
## 0.7879 0.5000000 1.0000000 0.000  
## 0.7980 0.5000000 1.0000000 0.000  
## 0.8081 0.5000000 1.0000000 0.000  
## 0.8182 0.5000000 1.0000000 0.000  
## 0.8283 0.5000000 1.0000000 0.000  
## 0.8384 0.5000000 1.0000000 0.000  
## 0.8485 0.5000000 1.0000000 0.000  
## 0.8586 0.5000000 1.0000000 0.000  
## 0.8687 0.5000000 1.0000000 0.000  
## 0.8788 0.5000000 1.0000000 0.000  
## 0.8889 0.5000000 1.0000000 0.000  
## 0.8990 0.5000000 1.0000000 0.000  
## 0.9091 0.5000000 1.0000000 0.000  
## 0.9192 0.5000000 1.0000000 0.000  
## 0.9293 0.5000000 1.0000000 0.000  
## 0.9394 0.5000000 1.0000000 0.000  
## 0.9495 0.5000000 1.0000000 0.000  
## 0.9596 0.5000000 1.0000000 0.000  
## 0.9697 0.5000000 1.0000000 0.000  
## 0.9798 0.5000000 1.0000000 0.000  
## 0.9899 0.5000000 1.0000000 0.000  
## 1.0000 0.5000000 1.0000000 0.000  
##   
## Tuning parameter 'alpha' was held constant at a value of 1  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were alpha = 1 and lambda = 0.0607.

This block trains an XGBoost model in order to investigate feature importance.

labels = ifelse(mod\_df$Cohort == "Primary", 1, 0)  
  
dtrain <- mod\_df %>%  
 select(-Cohort) %>%   
 as.matrix() %>%   
 xgb.DMatrix(label = labels)  
  
xgb\_params <- list(  
 booster = "gbtree",  
 objective = "binary:logistic"  
)  
  
model\_xgb <- xgb.train(  
 params = xgb\_params,  
 data = dtrain,  
 print\_every\_n = 5,  
 nrounds = 60,  
 verbose = 1  
)

The following block plots the coefficients from the best model identified by glmnet.

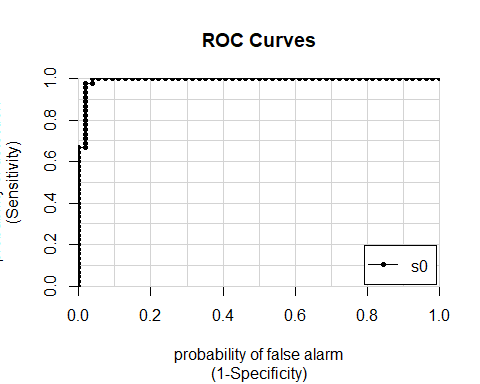
# Predictor  
feature\_matrix <- mod\_df %>%   
 select(-Cohort) %>%   
 as.matrix()  
  
# Target  
cohort <- mod\_df$Cohort  
  
lasso\_fit <- glmnet(feature\_matrix, cohort, family = "binomial", alpha = 1, lambda = model$bestTune$lambda)  
  
lasso\_fit %>%   
 tidy() %>%   
 arrange(desc(estimate)) %>%   
 filter(term != "(Intercept)") %>%   
 mutate(term = fct\_reorder(term, estimate)) %>%   
 ggplot(aes(term, estimate)) +  
 geom\_col() +  
 coord\_flip() +  
 labs(title = "Coefficients in predictive model",  
 subtitle = "Based on LASSO regression",  
 x = "",  
 y = "Coefficient")



# Evaluating

This block evaluates the performance of the model using the ROC metric.

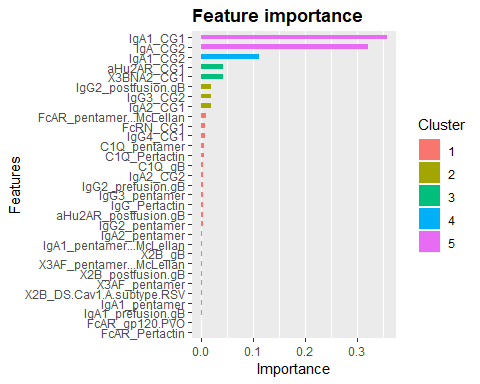
library(caTools)  
preds <- predict(lasso\_fit, feature\_matrix, type = "response")  
  
colAUC(preds, cohort, plotROC = TRUE)



## s0  
## Latent vs. Primary 0.9934156

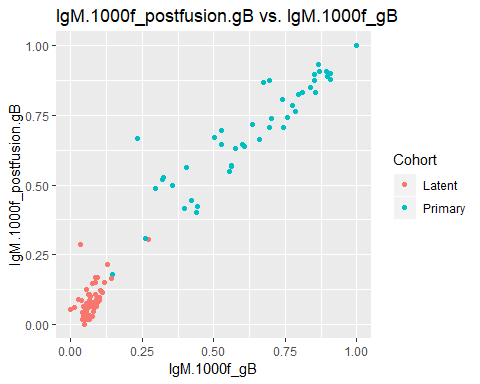
This block displays the feature importance metric from XGBoost. This displays the number of times the tree-based model split on a particular feature in order to perform classification. The more times a particular feature is chosen for a split, the more crucial it is to the model.

importance\_matrix <- xgb.importance(model = model\_xgb)  
xgb.ggplot.importance(importance\_matrix)

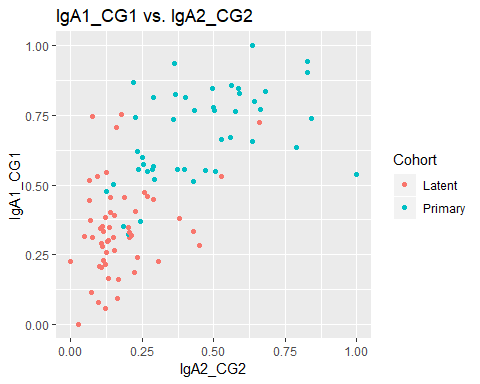


The following three plots scatter features chosen by the models as important against each other. As you can see, there is a lot of separation in many cases.

ggplot(norm\_df, aes(x = IgM.1000f\_gB, y = IgM.1000f\_postfusion.gB, color = Cohort)) +  
 geom\_point() +  
 labs(title = "IgM.1000f\_postfusion.gB vs. IgM.1000f\_gB")



ggplot(norm\_df, aes(x = IgA2\_CG2, y = IgA1\_CG1, color = Cohort)) +  
 geom\_point() +  
 labs(title = "IgA1\_CG1 vs. IgA2\_CG2")



ggplot(norm\_df, aes(x = IgG3\_pentamer, y = IgA1\_CG1, color = Cohort)) +  
 geom\_point() +  
 labs(title = "IgA1\_CG1 vs. IgG3\_pentamer")

