PM-272 Assignment 2: Using Machine Learning to Predict the Pathogenic Outcomes of Genetic Mutations for Epilepsy and Muscular Condition Phenotypes

Harley Burrow (2318180)

Table of contents

1 Introduction to the Datasets **2**

1.1 Dataset 1 - Humvar . . . 2

1.2 Dataset 2 - Lab Variants . . . 2

1.3 Dataset 3 - Variants Annotated . . . 2

2 Explanation of the Code **2**

2.1 Read in the Datasets and Required Libraries . . . 2

2.1.1 randomForest . . . 2

2.1.2 tidyverse . . . 3

2.1.3 pROC . . . 3

2.2 Create the Training and Test Sets for Humvar . . . 3

2.3 Train the Random Forest Model . . . 5

2.4 Make the Humvar Prediction, and Find Accuracy, True Positive Rate and False Positive Rate . . . 6

2.5 Join db.2 & db.3 to Show ID, label, and phenotype . . . 8

2.6 Filter New Table To Only Show Epilepsy OR Muscular Conditions . . . 9

2.7 Create Shuffled Epilepsy and Muscular Conditions Sets and Run the Predictions 11

2.8 Report Each Set's Accuracy, True Positive Rate and False Positive Rate . . . 12

2.8.1 Epilepsy . . . 12

2.8.2 Muscular Conditions . . . 13

2.9 Analysis of the Prediction Data . . . 14

2.10 Plotting the Receiver Operating Characteristics Curve . . . 14

3 Relevancy of Results **17**

1 Introduction to the Datasets

1.1 Dataset 1 - Humvar

This dataset describes the aspects of 36,684 genetic variants and whether their presence results in a benign or pathogenic outcome for health. It will be used to train the machine learning (ML) model which will then be able to specifically predict the likelihood of genetic variants causing epilepsy and muscular conditions.

1.2 Dataset 2 - Lab Variants

This shows a list of mutations (including mutation ID) and whether they cause epilepsy or muscular conditions. It will be used to match these same mutation IDs on the third dataset with their corresponding variant benignity predictors.

1.3 Dataset 3 - Variants Annotated

This shows the variant benignity predictors for each mutation ID. It will be merged with the Lab Variants dataset to run the ML model that will predict whether each mutation ID and their predictors will be likely to result in a benign or pathogenic epilepsy/muscular condition phenotype.

2 Explanation of the Code

2.1 Read in the Datasets and Required Libraries

```r
db.1 <- read.csv("humvar.csv")
db.2 <- read.csv("lab_variants.csv")
db.3 <- read.csv("variants_annotated.csv")

library(randomForest)
library(tidyverse)
library(pROC)
```

2.1.1 randomForest

I loaded in the "randomForest" library so I could run the random forest algorithms to train the ML model and make predictions.

2.1.2 tidyverse

I loaded in the "tidyverse" library to help me transform and present data.

2.1.3 pROC

I loaded in the "pROC" library so I could produce the receiver operating characteristic (ROC) curve.

2.2 Create the Training and Test Sets for Humvar

```
# Check for missing data
summary(is.na(db.1))
```

```
#>  DANN_score      GM12878_fitCons_score
#>  Mode :logical   Mode :logical
#>  FALSE:38684     FALSE:38684
#>  GM12878_fitCons_score_rankscore GenoCanyon_score
#>  Mode :logical                   Mode :logical
#>  FALSE:38684                     FALSE:38684
#>  GenoCanyon_score_rankscore H1.hESC_fitCons_score
#>  Mode :logical              Mode :logical
#>  FALSE:38684                FALSE:38684
#>  H1.hESC_fitCons_score_rankscore HUVEC_fitCons_score
#>  Mode :logical                   Mode :logical
#>  FALSE:38684                     FALSE:38684
#>  HUVEC_fitCons_score_rankscore MetaLR_score
#>  Mode :logical                 Mode :logical
#>  FALSE:38684                   FALSE:38684
#>  MutationAssessor_score_rankscore REVEL_score
#>  Mode :logical                    Mode :logical
#>  FALSE:38684                      FALSE:38684
#>  fathmm.MKL_coding_score integrated_fitCons_score
#>  Mode :logical           Mode :logical
#>  FALSE:38684             FALSE:38684
#>  integrated_fitCons_score_rankscore PolyPhen_score  SIFT_score
#>  Mode :logical                      Mode :logical   Mode :logical
#>  FALSE:38684                        FALSE:38684     FALSE:38684
#>  CADD_raw_rankscore DANN_rankscore  Eigen.PC.raw_rankscore
#>  Mode :logical      Mode :logical   Mode :logical
#>  FALSE:38684        FALSE:38684     FALSE:38684
#>  FATHMM_converted_rankscore GERP.._RS_rankscore
#>  Mode :logical              Mode :logical
#>  FALSE:38684                FALSE:38684
#>  GM12878_fitCons_score_rankscore.1 GenoCanyon_score_rankscore.1
#>  Mode :logical                     Mode :logical
#>  FALSE:38684                       FALSE:38684
#>  H1.hESC_fitCons_score_rankscore.1 HUVEC_fitCons_score_rankscore.1
#>  Mode :logical                     Mode :logical
#>  FALSE:38684                       FALSE:38684
#>  MetaLR_rankscore MetaSVM_rankscore
#>  Mode :logical    Mode :logical
#>  FALSE:38684      FALSE:38684
#>  MutationAssessor_score_rankscore.1
#>  Mode :logical
#>  FALSE:38684
#>  MutationTaster_converted_rankscore REVEL_rankscore
#>  Mode :logical                      Mode :logical
#>  FALSE:38684                        FALSE:38684
#>  SiPhy_29way_logOdds_rankscore VEST3_rankscore
#>  Mode :logical                 Mode :logical
#>  FALSE:38684                   FALSE:38684
#>  fathmm.MKL_coding_rankscore integrated_fitCons_score_rankscore.1
#>  Mode :logical               Mode :logical
#>  FALSE:38684                 FALSE:38684
#>  phastCons100way_vertebrate_rankscore
#>  Mode :logical
#>  FALSE:38684
#>  phastCons20way_mammalian_rankscore phyloP100way_vertebrate_rankscore
#>  Mode :logical                      Mode :logical
#>  FALSE:38684                        FALSE:38684
#>  phyloP20way_mammalian_rankscore Reliability_index GERP.._NR
#>  Mode :logical                   Mode :logical     Mode :logical
#>  FALSE:38684                     FALSE:38684       FALSE:38684
#>  SiPhy_29way_logOdds SiPhy_29way_logOdds_rankscore.1
#>  Mode :logical       Mode :logical
#>  FALSE:38684         FALSE:38684
#>  phastCons20way_mammalian df.freq.gnomAD_exomes_AF
#>  Mode :logical            Mode :logical
#>  FALSE:38684              FALSE:38684
#>  df.freq.gnomAD_genomes_AF   labels
#>  Mode :logical             Mode :logical
#>  FALSE:38684               FALSE:38684
```

```r
# Shuffle the data set so it's random and unbiased
set.seed(123)
humvar_shuffled <- db.1[sample(1:nrow(db.1)),]

# select 75% of the data
humvar_bound <- nrow(humvar_shuffled) * 0.75

# Assign first 75% of rows for training
humvar_train <- humvar_shuffled[1:humvar_bound,]

# Assign remaining 25% of rows to the test set
humvar_test <- humvar_shuffled[(humvar_bound+1):nrow(humvar_shuffled),]
```

"summary(is.na(db.1))" checks the Humvar dataset for any missing data that may have to be removed/completed. Fortunately there was no missing data so no further completion was required.

I then shuffled the dataset using the sample() function. I did this because it made sure to eliminate any bias that might have arisen from the order that the genetic variants had been listed. Setting a seed made sure that each time the dataset is shuffled, the data points are put into the exact same order. This ensures reproducibility of the model.

Using "nrow(humvar_shuffled) * 0.75" I selected 75% of the rows in the shuffled training data. I then assigned this 75% to the training set ("humvar_train <- humvar_shuffled[1:humvar_bound,]"), and the remaining 25% to the test set ("humvar_test <- humvar_shuffled[(humvar_bound+1):nrow(humvar_shuffled),]").

Using a 75:25 ratio for the training and testing sets was important because having a higher amount of data to train the ML model means it will be more accurate, but I also needed enough data in the test set to ensure the accuracy, and true positive and false positive rates were accurate enough to the true rates.

2.3 Train the Random Forest Model

```r
# set seed for reproducibility
set.seed(456)
# Run the random forest to train the ML model
humvar_train_rf <- randomForest(as.factor(labels) ~ ., data = humvar_train)

# Output the results of the random forest
humvar_train_rf
```

```
#>
#> Call:
#>  randomForest(formula = as.factor(labels) ~ ., data = humvar_train)
#>                Type of random forest: classification
#>                      Number of trees: 500
#> No. of variables tried at each split: 6
#>
#>         OOB estimate of  error rate: 6.25%
#> Confusion matrix:
#>            Benign Pathogenic class.error
#> Benign      17997       1004  0.05283932
#> Pathogenic    809       9203  0.08080304
```

```
# save the model so it will create the same output every time
saveRDS(humvar_train_rf,"humvar_train_rf.rds")
# read the saved model to ensure reproducibility
humvar_train_rf <- readRDS("humvar_train_rf.rds")
```

The as.factor() function tells the randomForest that "labels" is categorical data (data that can take one of a certain number of categories) and not continuous data. This was important to make sure the random forest was classification based rather than regression based. I used random forest as it runs many decision trees, meaning it will be highly accurate and will plot a more accurate ROC curve later.

The OOB estimate of error rate represents the percentage of the predictions made on out-of-bag samples (the third of the data not used in each decision tree) that were incorrect. The rate was 6.25% which means that 6.25% of the OOB predictions were incorrect. The class errors depict the proportion of benign and pathogenic labels that were incorrectly predicted. 5.28% of benign outcomes were incorrectly labelled, and 8.08% of pathogenic outcomes were incorrectly labelled.

Because I used save(RDS) to save the random forest ML model, the model will not have to be trained every time a new prediction is made using the model. This ensures reproducibility and consistency in the results outputted by the model.

2.4 Make the Humvar Prediction, and Find Accuracy, True Positive Rate and False Positive Rate

```
# Make the prediction on the test set using the trained ML model
humvar_pred <- predict(humvar_train_rf, humvar_test)
```

```r
 # View the first six results of the prediction
head(humvar_pred)
```

```
#>      23272       2891      28349      19923      17660       8045
#> Pathogenic Pathogenic Pathogenic     Benign     Benign     Benign
#> Levels: Benign Pathogenic
```

```r
# Save the prediction into a confusion matrix
humvar_cm <- table(humvar_pred,humvar_test$labels)

# View the prediction confusion matrix
humvar_cm
```

```
#>
#> humvar_pred  Benign Pathogenic
#>   Benign       6061        273
#>   Pathogenic    336       3001
```

Creating a confusion matrix and saving it as an object meant that I could calculate the accuracy, true positive rate and false positive rate for the model based on the coordinates of the test data's confusion matrix which made coding the calculations quicker.

The top label of the confusion matrix shows what label the model predicted, and the left-hand label of the confusion matrix shows the true label. As such…

[1,1] = True negative/TN (6061)

[2,2] = True positive/TP (3001)

[1,2] = False positive/FP (273)

[2,1] = False negative/FN (336)

Accuracy = (TN + TP) / (TN + TP + FN + FP) = 93.7% of variants' pathogenic outcome that were predicted correctly

True positive rate = (TP) / (TP + FN) = 89.9% of pathogenic variants that were labelled as pathogenic

False positive rate = (FP) / (FP + TN) = 4.31% of benign variants that were labelled as pathogenic

```
# Find accuracy
humvar_pred_accuracy <- (
  humvar_cm[1, 1] + humvar_cm[2, 2]
) / (
  humvar_cm[1, 1] + humvar_cm[1, 2] +
  humvar_cm[2, 1] + humvar_cm[2, 2]
)
humvar_pred_accuracy
```

```
#> [1] 0.9370282
```

```
# Find true positive rate
humvar_pred_TPR <- (
  humvar_cm[2,2]
) / (
humvar_cm[2,1] + humvar_cm[2,2]
)
humvar_pred_TPR
```

```
#> [1] 0.8993108
```

```
# Find false positive rate
humvar_pred_FPR <- (
  humvar_cm[1,2]
) / (
humvar_cm[1,2] + humvar_cm[1,1]
)
humvar_pred_FPR
```

```
#> [1] 0.04310073
```

2.5 Join db.2 & db.3 to Show ID, label, and phenotype

```
pheno_patho_joined <- inner_join(db.2, db.3, by = "ids")
```

2.6 Filter New Table To Only Show Epilepsy OR Muscular Conditions

First I checked to see if there were any rows with missing data:

```
 # Check for missing data as above
summary(is.na(pheno_patho_joined))
```

```
#>     ids           phenotype       DANN_score     
#>  Mode :logical   Mode :logical   Mode :logical  
#>  FALSE:857       FALSE:857       FALSE:857      
#>                                                 
#>  GM12878_fitCons_score GM12878_fitCons_score_rankscore
#>  Mode :logical         Mode :logical                  
#>  FALSE:857             FALSE:857                      
#>                                                       
#>  GenoCanyon_score GenoCanyon_score_rankscore H1.hESC_fitCons_score
#>  Mode :logical    Mode :logical              Mode :logical        
#>  FALSE:832        FALSE:857                  FALSE:857            
#>  TRUE :25                                                         
#>  H1.hESC_fitCons_score_rankscore HUVEC_fitCons_score
#>  Mode :logical                   Mode :logical      
#>  FALSE:857                       FALSE:857          
#>                                                     
#>  HUVEC_fitCons_score_rankscore MetaLR_score   
#>  Mode :logical                 Mode :logical  
#>  FALSE:857                     FALSE:857      
#>                                               
#>  MutationAssessor_score_rankscore REVEL_score    
#>  Mode :logical                    Mode :logical  
#>  FALSE:857                        FALSE:857      
#>                                                  
#>  fathmm.MKL_coding_score integrated_fitCons_score
#>  Mode :logical           Mode :logical           
#>  FALSE:857               FALSE:857               
#>                                                  
#>  integrated_fitCons_score_rankscore PolyPhen_score  SIFT_score     
#>  Mode :logical                      Mode :logical   Mode :logical  
#>  FALSE:857                          FALSE:857       FALSE:857      
#>                                                                    
#>  CADD_raw_rankscore DANN_rankscore  Eigen.PC.raw_rankscore
#>  Mode :logical      Mode :logical   Mode :logical         
#>  FALSE:857          FALSE:857       FALSE:857             
```

```
#>
#>  FATHMM_converted_rankscore GERP.._RS_rankscore
#>  Mode :logical              Mode :logical
#>  FALSE:857                  FALSE:857
#>
#>  GM12878_fitCons_score_rankscore.1 GenoCanyon_score_rankscore.1
#>  Mode :logical                     Mode :logical
#>  FALSE:857                         FALSE:857
#>
#>  H1.hESC_fitCons_score_rankscore.1 HUVEC_fitCons_score_rankscore.1
#>  Mode :logical                     Mode :logical
#>  FALSE:857                         FALSE:857
#>
#>  MetaLR_rankscore MetaSVM_rankscore
#>  Mode :logical    Mode :logical
#>  FALSE:857        FALSE:857
#>
#>  MutationAssessor_score_rankscore.1
#>  Mode :logical
#>  FALSE:857
#>
#>  MutationTaster_converted_rankscore REVEL_rankscore
#>  Mode :logical                      Mode :logical
#>  FALSE:857                          FALSE:857
#>
#>  SiPhy_29way_logOdds_rankscore VEST3_rankscore
#>  Mode :logical                 Mode :logical
#>  FALSE:857                     FALSE:857
#>
#>  fathmm.MKL_coding_rankscore integrated_fitCons_score_rankscore.1
#>  Mode :logical               Mode :logical
#>  FALSE:857                   FALSE:857
#>
#>  phastCons100way_vertebrate_rankscore
#>  Mode :logical
#>  FALSE:857
#>
#>  phastCons20way_mammalian_rankscore phyloP100way_vertebrate_rankscore
#>  Mode :logical                      Mode :logical
#>  FALSE:857                          FALSE:857
#>
#>  phyloP20way_mammalian_rankscore Reliability_index GERP.._NR
#>  Mode :logical                   Mode :logical     Mode :logical
#>  FALSE:857                       FALSE:857         FALSE:857
#>
#>  SiPhy_29way_logOdds SiPhy_29way_logOdds_rankscore.1
#>  Mode :logical       Mode :logical
#>  FALSE:857           FALSE:857
#>
#>  phastCons20way_mammalian df.freq.gnomAD_exomes_AF
#>  Mode :logical            Mode :logical
#>  FALSE:857                FALSE:857
#>
#>  df.freq.gnomAD_genomes_AF   labels
#>  Mode :logical             Mode :logical
#>  FALSE:857                 FALSE:857
#>
```

There was some missing data, but only in 25 rows out of 857 (2.91%) and only in the Geno-Canyon_score column. Because there was such little data missing, I decided to simply remove the rows with missing data rather than using a multiple imputation method such as MICE as the missing data would not skew the results.

```r
epilepsy_no_na <- pheno_patho_joined %>%
  # Filter the table so it only shows epilepsy phenotypes
  filter(phenotype == "epilepsy") %>%
    # Remove rows with missing data
  na.omit()

muscular_no_na <- pheno_patho_joined %>%
  filter(phenotype == "muscular_conditions") %>%
  na.omit()
```

I filtered the overall joined table to make two new separate ones with just epilepsy and just muscular conditions because it would mean I could run their own random forest models and compare the results.

2.7 Create Shuffled Epilepsy and Muscular Conditions Sets and Run the Predictions

```r
# Shuffles the data sets so it's random and unbiased
set.seed(789)
epilepsy_shuffled <- epilepsy_no_na[sample(1:nrow(epilepsy_no_na)),]
```

```
set.seed(246)
muscular_shuffled <- muscular_no_na[sample(1:nrow(muscular_no_na)),]

# Make epilepsy prediction
epilepsy_pred <- predict(humvar_train_rf, epilepsy_shuffled)
# Make muscular conditions prediction
muscular_pred <- predict(humvar_train_rf, muscular_shuffled)

# Create the confusion matrices from the predictions
epilepsy_cm <- table(epilepsy_pred,epilepsy_shuffled$labels)
muscular_cm <- table(muscular_pred,muscular_shuffled$labels)
```

```
# View the confusion matrices
epilepsy_cm
```

```
#>
#> epilepsy_pred Benign Pathogenic
#>    Benign         70         22
#>    Pathogenic     19        179
```

```
muscular_cm
```

```
#>
#> muscular_pred Benign Pathogenic
#>    Benign        296         12
#>    Pathogenic     19        215
```

2.8 Report Each Set's Accuracy, True Positive Rate and False Positive Rate

2.8.1 Epilepsy

```
epilepsy_pred_accuracy <- (
epilepsy_cm[1,1] + epilepsy_cm[2,2]
) / (
epilepsy_cm[1,1] + epilepsy_cm[1,2]
+ epilepsy_cm[2,1] + epilepsy_cm[2,2]
)
epilepsy_pred_accuracy
```

```
#> [1] 0.8586207
```

```
epilepsy_pred_TPR <- (
epilepsy_cm[2,2]
) / (
epilepsy_cm[2,1] + epilepsy_cm[2,2]
)
epilepsy_pred_TPR
```

```
#> [1] 0.9040404
```

```
epilepsy_pred_FPR <- (
epilepsy_cm[1,2]
) / (
epilepsy_cm[1,2] + epilepsy_cm[1,1]
)
epilepsy_pred_FPR
```

```
#> [1] 0.2391304
```

2.8.2 Muscular Conditions

```
muscular_pred_accuracy <- (
muscular_cm[1,1] + muscular_cm[2,2]
) / (
muscular_cm[1,1] + muscular_cm[1,2] +
muscular_cm[2,1] + muscular_cm[2,2])
muscular_pred_accuracy
```

```
#> [1] 0.9428044
```

```
muscular_pred_TPR <- (
muscular_cm[2,2]
) / (
muscular_cm[2,1] + muscular_cm[2,2]
)
muscular_pred_TPR
```

```
#> [1] 0.9188034
```

```
muscular_pred_FPR <- (
muscular_cm[1,2]
) / (
muscular_cm[1,2] + muscular_cm[1,1]
)
muscular_pred_FPR
```

```
#> [1] 0.03896104
```

2.9 Analysis of the Prediction Data

This data shows that it is easier to predict if an outcome will cause muscular conditions than it is to predict if it will cause epilepsy. This is because the prediction for muscular outcomes has a higher accuracy (0.9428044 vs 0.8586207) and true positive rate (0.9188034 vs 0.9040404),as well as a lower false positive rate (0.03896104 vs 0.2391304).

2.10 Plotting the Receiver Operating Characteristics Curve

The first thing I had to do was turn the epilepsy and muscular conditions predictions from categorical data into continuous probability data so it could be plotted as a regression:

```
# Change from categorical data to continuous
epilepsy_probs <- as.data.frame(predict
(humvar_train_rf, epilepsy_shuffled, type = "prob")
)
muscular_probs <- as.data.frame(predict
(humvar_train_rf, muscular_shuffled, type = "prob")
)
```

I then assigned the benign and pathogenic labels the value of 1 or 2 rather than having text-based labels as this made it easier for the ROC function to interpret the data and avoid mistakes:

```
# Change the labels from text based to numeric
epilepsy_num_lab <- epilepsy_shuffled %>%
  mutate(num_label = case_when(
    labels == "Benign" ~ 1,
    labels == "Pathogenic" ~ 2
```

```r
  ))

muscular_num_lab <- muscular_shuffled %>%
  mutate(num_label = case_when(
    labels == "Benign" ~ 1,
    labels == "Pathogenic" ~ 2
  ))
```

Next, for each phenotype, I had to outline the response and predictor values for the ROC curves. The first line for each creates the response values, which are the numerical labels for "benign" and "pathogenic". The second line creates the predictor values which are based on the probability for each label. For the predictors it did not matter which pathogenicity I chose as they are both proportional to each other.

```r
# Set the response values
epilepsy_labels = c(epilepsy_num_lab$num_label)
# Set the predictor values
epilepsy_benign_predictor = c(epilepsy_probs$Benign)

# Set the response values
muscular_labels = c(muscular_num_lab$num_label)
# Set the predictor values
muscular_benign_predictor = c(muscular_probs$Benign)
```

Next I ran the ROC calculations:

```r
# calculate ROC
epilepsy_roc <- roc(
  response = epilepsy_labels, predictor = epilepsy_benign_predictor
  )
```

```
#> Setting levels: control = 1, case = 2

#> Setting direction: controls > cases
```

```r
muscular_roc <- roc(
  response = muscular_labels, predictor = muscular_benign_predictor
  )
```

```
#> Setting levels: control = 1, case = 2
#> Setting direction: controls > cases
```

```
epilepsy_roc
```

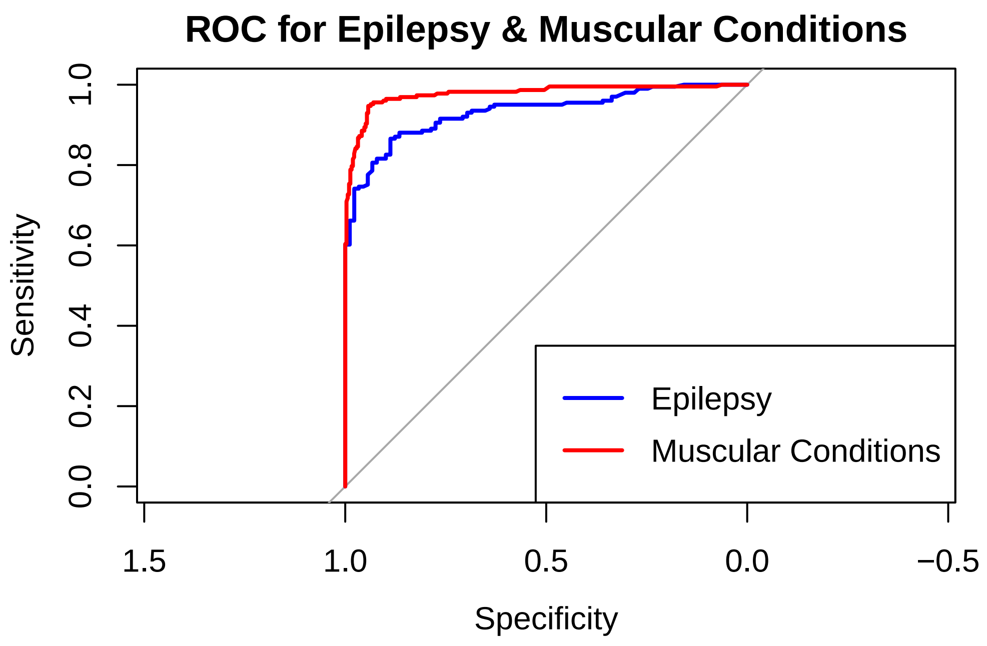
```
#>
#> Call:
#> roc.default(response = epilepsy_labels, predictor = epilepsy_benign_predictor)
#>
#> Data: epilepsy_benign_predictor in 89 controls (epilepsy_labels 1) > 201 cases (epilepsy_l
#> Area under the curve: 0.932
```

```
muscular_roc
```

```
#>
#> Call:
#> roc.default(response = muscular_labels, predictor = muscular_benign_predictor)
#>
#> Data: muscular_benign_predictor in 315 controls (muscular_labels 1) > 227 cases (muscular_
#> Area under the curve: 0.9776
```

Finally, I plotted the ROC curves and joined them together

```
plot(
  epilepsy_roc, col = "blue", main = "ROC for Epilepsy & Muscular Conditions"
  )
lines(muscular_roc, col = "red")
legend("bottomright", legend = c("Epilepsy", "Muscular Conditions"),
       col = c("blue", "red"), lwd = 2)
```



Sensitivity in the ROC curves represents the true positive rate (TPR) and specificity represents the true negative rate (TNR). These values are plotted for certain thresholds for labelling variants as pathogenic or benign. As high values for sensitivity (TPR) and specificity (TNR) mean a more accurate ML model, having both values as close to 1.0 as possible on the ROC curve will represent this more accurate model. As such, a dataset's curve that has a higher area under the curve (AUC) will mean the predictions made for this dataset are more accurate and reliable. From the plot, we can see that the curve for muscular conditions has a higher AUC than the curve for epilepsy (0.9776 compared to 0.932, respectively). This means that the predictions for pathogenic outcomes of genetic variants in muscular conditions are more accurate and reliable in this model than they are for epilepsy. This is also represented by the accuracies, TPRs and FPRs calculated using the confusion matrices.

3 Relevancy of Results

These findings show that the accuracies of this model are not high enough to be used as the sole basis of diagnosing a patient for muscular conditions or epilepsy purely based on the presence of mutation IDs that have been labelled as pathogenic. This is because the accuracies for muscular conditions and epilepsy are not high enough to make decisions based on the ML model alone, particularly for epilepsy as all of the results show it is harder to predict a pathogenic outcome for it than muscular conditions. Instead, this model should be used to encourage clinician based examinations of patients with pathogenically labelled mutations, as there is still a calculatedly high probability that these mutations can lead to a pathogenic outcome.