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. 1 - Introduction to the Datasets

### 1.0.1 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.

### 1.0.2 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.

### 1.0.3 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. 2 - Explanation of the Code

## 2.1 2.1 - -Read in the Datasets and Required Libraries

db.1 <- read.csv("humvar.csv")   
db.2 <- read.csv("lab\_variants.csv")  
db.3 <- read.csv("variants\_annotated.csv")  
  
library(randomForest)  
library(tidyverse)

Warning: package 'readr' was built under R version 4.4.2

Warning: package 'dplyr' was built under R version 4.4.2

library(pROC)

Warning: package 'pROC' was built under R version 4.4.2

### 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 2.2 - Create the Training and Test Sets for Humvar

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

DANN\_score GM12878\_fitCons\_score GM12878\_fitCons\_score\_rankscore  
 Mode :logical Mode :logical Mode :logical   
 FALSE:38684 FALSE:38684 FALSE:38684   
 GenoCanyon\_score GenoCanyon\_score\_rankscore H1.hESC\_fitCons\_score  
 Mode :logical Mode :logical Mode :logical   
 FALSE:38684 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 MutationAssessor\_score\_rankscore  
 Mode :logical Mode :logical Mode :logical   
 FALSE:38684 FALSE:38684 FALSE:38684   
 REVEL\_score fathmm.MKL\_coding\_score integrated\_fitCons\_score  
 Mode :logical Mode :logical Mode :logical   
 FALSE:38684 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 MutationAssessor\_score\_rankscore.1  
 Mode :logical Mode :logical Mode :logical   
 FALSE:38684 FALSE:38684 FALSE:38684   
 MutationTaster\_converted\_rankscore REVEL\_rankscore  
 Mode :logical Mode :logical   
 FALSE:38684 FALSE:38684   
 SiPhy\_29way\_logOdds\_rankscore VEST3\_rankscore fathmm.MKL\_coding\_rankscore  
 Mode :logical Mode :logical Mode :logical   
 FALSE:38684 FALSE:38684 FALSE:38684   
 integrated\_fitCons\_score\_rankscore.1 phastCons100way\_vertebrate\_rankscore  
 Mode :logical Mode :logical   
 FALSE:38684 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 phastCons20way\_mammalian  
 Mode :logical Mode :logical Mode :logical   
 FALSE:38684 FALSE:38684 FALSE:38684   
 df.freq.gnomAD\_exomes\_AF df.freq.gnomAD\_genomes\_AF labels   
 Mode :logical Mode :logical Mode :logical   
 FALSE:38684 FALSE:38684 FALSE:38684

humvar\_shuffled <- db.1[sample(1:nrow(db.1)),] # Shuffle the data set so it's random and unbiased  
  
humvar\_bound <- nrow(humvar\_shuffled) \* 0.75 # select 75% of the data  
  
humvar\_train <- humvar\_shuffled[1:humvar\_bound,] # Assign first 75% of rows for training  
humvar\_test <- humvar\_shuffled[(humvar\_bound+1):nrow(humvar\_shuffled),] # Assign remaining 25% of rows to the test set

“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.

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 2.4 - Train the Random Forest Model

humvar\_train\_rf <- randomForest(as.factor(labels) ~ ., data = humvar\_train) # Run the random forest to train the ML model  
humvar\_train\_rf # Output the results of the random forest

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: 6.19%  
Confusion matrix:  
 Benign Pathogenic class.error  
Benign 18032 996 0.05234391  
Pathogenic 800 9185 0.08012018

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 class errors depict the proportion of benign and pathogenic labels that were incorrectly predicted.

## 2.4 2.5 - Make the Humvar Prediction, and Find Accuracy, True Positive Rate and False Positive Rate

humvar\_pred <- predict(humvar\_train\_rf, humvar\_test) # Make the prediction on the test set using the trained ML model  
head(humvar\_pred) # View the first six results of the prediction

1654 29991 8790 4139 16118 26959   
Benign Benign Benign Benign Benign Benign   
Levels: Benign Pathogenic

humvar\_cm <- table(humvar\_pred,humvar\_test$labels) # Save the prediction into a confusion matrix  
  
humvar\_cm # View the prediction confusion matrix

humvar\_pred Benign Pathogenic  
 Benign 6009 288  
 Pathogenic 361 3013

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

[2,2] = True positive/TP

[1,2] = False positive/FP

[2,1] = False negative/FN

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

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

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

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]) # Find accuracy  
humvar\_pred\_accuracy

[1] 0.9328922

humvar\_pred\_TPR <- (humvar\_cm[2,2]) / (humvar\_cm[2,1] + humvar\_cm[2,2]) # Find true positive rate  
humvar\_pred\_TPR

[1] 0.8930053

humvar\_pred\_FPR <- (humvar\_cm[1,2]) / (humvar\_cm[1,2] + humvar\_cm[1,1]) # Find false positive rate  
humvar\_pred\_FPR

[1] 0.04573606

## 2.5 2.6 - Join db.2 & db.3 to Show ID, label, and phenotype

pheno\_patho\_joined <- inner\_join(db.2, db.3, by = "ids")

## 2.6 2.7 - Filter New Table To Only Show Epilepsy OR Muscular Conditions

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

summary(is.na(pheno\_patho\_joined)) # Check for missing data as above

ids phenotype DANN\_score GM12878\_fitCons\_score  
 Mode :logical Mode :logical Mode :logical Mode :logical   
 FALSE:857 FALSE:857 FALSE:857 FALSE:857   
   
 GM12878\_fitCons\_score\_rankscore GenoCanyon\_score GenoCanyon\_score\_rankscore  
 Mode :logical Mode :logical Mode :logical   
 FALSE:857 FALSE:832 FALSE:857   
 TRUE :25   
 H1.hESC\_fitCons\_score H1.hESC\_fitCons\_score\_rankscore HUVEC\_fitCons\_score  
 Mode :logical Mode :logical Mode :logical   
 FALSE:857 FALSE:857 FALSE:857   
   
 HUVEC\_fitCons\_score\_rankscore MetaLR\_score MutationAssessor\_score\_rankscore  
 Mode :logical Mode :logical Mode :logical   
 FALSE:857 FALSE:857 FALSE:857   
   
 REVEL\_score fathmm.MKL\_coding\_score integrated\_fitCons\_score  
 Mode :logical Mode :logical Mode :logical   
 FALSE:857 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 MutationAssessor\_score\_rankscore.1  
 Mode :logical Mode :logical Mode :logical   
 FALSE:857 FALSE:857 FALSE:857   
   
 MutationTaster\_converted\_rankscore REVEL\_rankscore  
 Mode :logical Mode :logical   
 FALSE:857 FALSE:857   
   
 SiPhy\_29way\_logOdds\_rankscore VEST3\_rankscore fathmm.MKL\_coding\_rankscore  
 Mode :logical Mode :logical Mode :logical   
 FALSE:857 FALSE:857 FALSE:857   
   
 integrated\_fitCons\_score\_rankscore.1 phastCons100way\_vertebrate\_rankscore  
 Mode :logical Mode :logical   
 FALSE:857 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 phastCons20way\_mammalian  
 Mode :logical Mode :logical Mode :logical   
 FALSE:857 FALSE:857 FALSE:857   
   
 df.freq.gnomAD\_exomes\_AF df.freq.gnomAD\_genomes\_AF labels   
 Mode :logical Mode :logical Mode :logical   
 FALSE:857 FALSE:857 FALSE:857

There was some missing data, but only in 25 rows out of 857 (2.91%) and only in the GenoCanyon\_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.

epilepsy\_no\_na <- pheno\_patho\_joined %>%   
 filter(phenotype == "epilepsy") %>% # Filter the table so it only shows epilepsy phenotypes  
 na.omit() # Remove rows with missing data  
  
muscular\_no\_na <- pheno\_patho\_joined %>%  
 filter(phenotype == "muscular\_conditions") %>% # Filter the table so it only shows muscular condition phenotypes  
 na.omit() # Removes rows with missing data

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 2.8 - Create Shuffled Epilepsy and Muscular Conditions Sets and Run the Predictions

epilepsy\_shuffled <- epilepsy\_no\_na[sample(1:nrow(epilepsy\_no\_na)),] # Shuffles the data sets so it's random and unbiased  
muscular\_shuffled <- muscular\_no\_na[sample(1:nrow(muscular\_no\_na)),]  
  
epilepsy\_pred <- predict(humvar\_train\_rf, epilepsy\_shuffled) # Make predictions on the labels of the epilepsy and muscular condition sets based  
 # on the trained random forest model  
muscular\_pred <- predict(humvar\_train\_rf, muscular\_shuffled)  
  
epilepsy\_cm <- table(epilepsy\_pred,epilepsy\_shuffled$labels) # Create the confusion matrices from the predictions  
muscular\_cm <- table(muscular\_pred,muscular\_shuffled$labels)

epilepsy\_cm # View the confusion matrices

epilepsy\_pred Benign Pathogenic  
 Benign 69 22  
 Pathogenic 20 179

muscular\_cm

muscular\_pred Benign Pathogenic  
 Benign 293 14  
 Pathogenic 22 213

## 2.8 2.9 - 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] 0.8551724

epilepsy\_pred\_TPR <- (epilepsy\_cm[2,2]) / (epilepsy\_cm[2,1] + epilepsy\_cm[2,2])  
epilepsy\_pred\_TPR

[1] 0.8994975

epilepsy\_pred\_FPR <- (epilepsy\_cm[1,2]) / (epilepsy\_cm[1,2] + epilepsy\_cm[1,1])   
epilepsy\_pred\_FPR

[1] 0.2417582

### 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] 0.9335793

muscular\_pred\_TPR <- (muscular\_cm[2,2]) / (muscular\_cm[2,1] + muscular\_cm[2,2])  
muscular\_pred\_TPR

[1] 0.906383

muscular\_pred\_FPR <- (muscular\_cm[1,2]) / (muscular\_cm[1,2] + muscular\_cm[1,1])   
muscular\_pred\_FPR

[1] 0.04560261

## 2.9 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:

epilepsy\_probs <- as.data.frame(predict(humvar\_train\_rf, epilepsy\_shuffled, type = "prob")) # Change from categorical to continuous  
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:

epilepsy\_num\_lab <- epilepsy\_shuffled %>% # Change the labels from text based to numeric  
 mutate(num\_label = case\_when(  
 labels == "Benign" ~ 1,  
 labels == "Pathogenic" ~ 2  
 ))  
  
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.

epilepsy\_labels = c(epilepsy\_num\_lab$num\_label) # Set the response values  
epilepsy\_benign\_predictor = c(epilepsy\_probs$Benign) # Set the predictor values  
  
muscular\_labels = c(muscular\_num\_lab$num\_label) # Set the response values  
muscular\_benign\_predictor = c(muscular\_probs$Benign) # Set the predictor values

Next I ran the ROC calculations:

epilepsy\_roc <- roc(response = epilepsy\_labels, predictor = epilepsy\_benign\_predictor) # calculate ROC

Setting levels: control = 1, case = 2

Setting direction: controls > cases

muscular\_roc <- roc(response = muscular\_labels, predictor = muscular\_benign\_predictor)

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

epilepsy\_roc

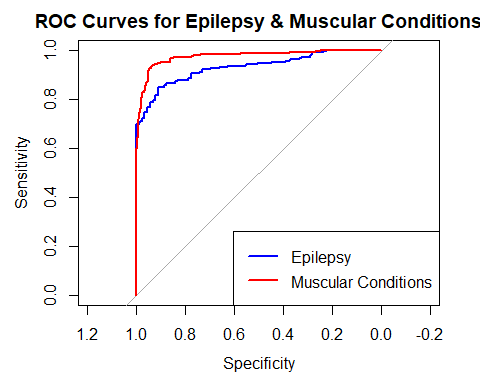
Call:  
roc.default(response = epilepsy\_labels, predictor = epilepsy\_benign\_predictor)  
  
Data: epilepsy\_benign\_predictor in 89 controls (epilepsy\_labels 1) > 201 cases (epilepsy\_labels 2).  
Area under the curve: 0.9304

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\_labels 2).  
Area under the curve: 0.9744

Finally, I plotted the ROC curves and joined them together

plot(epilepsy\_roc, col = "blue", main = "ROC Curves 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. 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. 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.