library(shiny)

library(bslib)

library(dplyr)

library(pROC)

library(ggplot2)

library(ggbiplot)

library(svglite)

library(tidyr)

library(bio3d)

library(mclust)

library(naivebayes)

library(psych)

library(randomForest)

library(caret)

library(purrr)

library(tidyverse)

library("klaR")

library(gridExtra)

# this app performs an initial analysis on integrating multiple MAVEs for the same gene

# Define UI for slider demo app ----

ui <- fluidPage(

# App title ----

titlePanel("Integrate multiple MAVE datasets with sample methods | current version 2.0"),

# Recent updates for version 2.0

## random forest now includes cross validation during training

## also add metrics for supervised learning that are specific to train and test splits

# Recent updates for version 1.9

## make separate upload buttons for MAVE data vs truth set

### make separate versions of merged dataframes so can download ML predictions for non-truth set variants too

# Recent updates for version 1.8

## fixed OddsPath calculation --> ask Malvika to double-check

### add ROC AUC to output model metric csv #done

# future updates

## add ROC curve plot capability if I can figure out how to get Shiny to cooperate with 'arbitrary' number of plots

## output Naive Bayes cross validation error score

## reimplement GFMM

# Sidebar with a slider input for number of bins

sidebarLayout(

sidebarPanel(

# Input: Select csv files ----

fileInput(

"files",

"Choose MAVE csv files | Multiple MAVE datasets, same gene",

multiple = TRUE,

accept = c(".csv")

),

fileInput(

"files2",

"Choose Truth Set csv file for gene of interest",

multiple = FALSE,

accept = c(".csv")

),

# Horizontal line ----

hr(),

numericInput("whatSeed", label = h6("What seed to set for random train/test split? default=1234"), value = 1234),

numericInput("num", label = h6("How many clusters for k-means? default=2"), value = 2),

numericInput("num2", label = h6("How many random sets for k-means? default=20"), value = 20),

#radioButtons("radio", label = h6("Which algorithm for GFMM?"),

# choices = list("MclustDA (Default)" = 1, "EDDA" = 2),

# selected = 1),

numericInput("num3", label = h6("What percent of variants for training (0.1-0.9)? default=0.8"), value = 0.8),

numericInput("num4", label = h6("How many trees for random forest? default=500"), value = 500),

#numericInput("num5", label = h6("What stepFactor for random forest? default=1.2"), value = 1.2),

#numericInput("num6", label = h6("What relative improvement to continue search? default=0.01"), value = 0.01),

hr(),

fluidPage(

#tags$head(tags$script(src = "message-handler.js")),

actionButton("do", "Click here to perform analysis")

)

),

# Show a plot of the generated distribution

mainPanel(

tabsetPanel(

tabPanel("Instructions",

br(),

h3("Input Requirements"),

h4("📁 MAVE CSV Files"),

tags$ul(

tags$li(strong("Format:"), "CSV files with variant effect scores"),

tags$li(strong("Required Column:"), code("hgvs\_pro"), "- variant identifier"),

tags$li(strong("Score Columns:"), "Any number of columns with numeric variant scores"),

tags$li(strong("Multiple Files:"), "Upload 2 or more MAVE datasets for the same gene")

),

h4("📋 Truth Set CSV File"),

tags$ul(

tags$li(strong("Format:"), "CSV file with known variant classifications"),

tags$li(strong("Required Columns:")),

tags$ul(

tags$li(code("hgvs\_pro"), "- variant identifier (must match MAVE files)"),

tags$li(code("binary\_clinvar\_class"), "- classification with values 'P' (pathogenic) or 'B' (benign)")

)

),

h4("⚙️ Workflow"),

tags$ol(

tags$li("Upload your MAVE CSV files"),

tags$li("Upload your truth set CSV file"),

tags$li("Click 'Perform Analysis'"),

tags$li("View results and download outputs")

)

),

tabPanel("Analysis Results",

plotOutput('plot1'),

plotOutput('plot2'),

plotOutput('plot3'),

plotOutput('plot4'),

#uiOutput("ROCplots"), # trying to use code from here to enable ROC plots but so far no luck: https://www.reddit.com/r/rstats/comments/nwyh3z/how\_to\_print\_multiple\_plots\_using\_plot\_function/

#textOutput("NB\_intro"),

#textOutput("NV\_cv"),

textOutput("RF\_intro"),

textOutput("RF\_cv")

),

tabPanel("Downloads",

br(),

h4("Download Results"),

br(),

fluidPage(

downloadButton("downloadData", "Download model metrics")

),

br(),

fluidPage(

downloadButton("downloadData2", "Download truth set variant level dataframe")

),

br(),

fluidPage(

downloadButton("downloadData3", "Download full variant level dataframe")

),

br(),

fluidPage(

downloadButton("downloadData4", "Download model metrics for supervised learning train and test splits")

),

br(),

fluidPage(

downloadButton("downloadPlot", "Download SVG of principal component plots")

),

br(),

fluidPage(

downloadButton("downloadPlot2", "Download SVG of OddsPath plots")

)

)

)

)

)

)

# Define server logic to read selected file ----

server <- function(input, output, session) {

observeEvent(input$do, {

req(input$files)

req(input$files2)

####### input app code here ########

# we are going to specify in the instructions how users format their input data

## to match the style of our TP53 scores csv

### and any duplicates need to be collapsed too with dplyr summarise or other method

# sample code for collapsing duplicates

#tibble1 <- df1 %>%

# group\_by(hgvs\_pro) %>%

# dplyr::summarise(mean = mean(score), n = n())

# read in 2 or more MAVE files and merge

print(length(input$files[,1])) # troubleshooting

print(input$files) # troubleshooting

lst = list()

for(i in 1:length(input$files[,1])){

#print(read.csv(file=input$files[[i, 'datapath']],sep=",")) # troubleshooting

lst[[i]] <- read.csv(file=input$files[[i, 'datapath']],sep=",")

#print(lst[[i]]) # troubleshooting

}

# read in truth set csv file

#print(input$files2) #troubleshooting

#print(input$files2$datapath) #troubleshooting

truthSet\_df <- read.csv(input$files2$datapath, sep=",") # Did I fix it?

#str(truthSet\_df)

# merge multiple dataframes in a list

#list(x, y, z) %>% reduce(left\_join, by = "i")

mergedMAVE.data.frame <- lst %>% reduce(left\_join, by = "hgvs\_pro")

#merged.data.frame <- Reduce(function(...) merge(..., all=T), lst)

#print(head(mergedMAVE.data.frame)) # troubleshooting

#print(tail(mergedMAVE.data.frame)) # troubleshooting

#print(sum(is.na(mergedMAVE.data.frame))) # any NAs in this dataframe?

# now merge truth set and MAVE dataframes

print('testing1') # troubleshooting

#df1 <- read.csv(file="MAVE1.csv", sep=",")

#df2 <- read.csv(file="MAVE2.csv", sep=",")

TSandMAVE\_df <- merge(mergedMAVE.data.frame,truthSet\_df,by="hgvs\_pro")

# need to drop NAs

# no\_na\_df is truth set and MAVE df without NAs

no\_na\_df <- TSandMAVE\_df %>% drop\_na()

# no\_na\_MAVEonly\_df is MAVE df without NAs

no\_na\_MAVEonly\_df <- mergedMAVE.data.frame %>% drop\_na()

print('testing2') # troubleshooting

################

################

################

# read in data

df1 <- no\_na\_df

# make a version of dataframe without the class column or variant name column

drops <- c("binary\_clinvar\_class","hgvs\_pro")

df\_subset <- df1[ , !(names(df1) %in% drops)]

# do same for no TruthSet df

df\_subset\_MAVEonly <- no\_na\_MAVEonly\_df[ , !(names(no\_na\_MAVEonly\_df) %in% drops)]

######################

######################

###################### PCA / kmeans

# scale the data

data\_scaled = scale(df\_subset)

data\_scaled\_MAVEonly <- scale(df\_subset\_MAVEonly)

# w scaling #PCA

pc <- prcomp(data\_scaled,

center = TRUE,

scale. = TRUE)

pc2 <- prcomp(data\_scaled\_MAVEonly,

center = TRUE,

scale. = TRUE)

# add PC1 to newdataframe

df\_wPC1 <- df\_subset

df\_wPC1$PC1 <- pc$x[,1]

no\_na\_df$PC1 <- df\_wPC1$PC1

no\_na\_MAVEonly\_df$PC1 <- pc2$x[,1]

print('testing3') # troubleshooting

# Kmeans ## w data scaling

set.seed(input$whatSeed)

kCluster <- kmeans(data\_scaled, input$num, nstart = input$num2)

kCluster

kCluster$cluster <- as.factor(kCluster$cluster)

table(kCluster$cluster, df1$binary\_clinvar\_class)

no\_na\_df$kMeansCluster <- kCluster$cluster

# kmeans clustering plot ## color points by kmeans assign cluster

k\_meansPLOT1 <- ggbiplot(pc,

obs.scale = 1,

var.scale = 1,

groups = kCluster$cluster,

ellipse = TRUE,

circle = TRUE,

ellipse.prob = 0.68)

k\_meansPLOT1 <- k\_meansPLOT1 + scale\_color\_discrete(name = '')

k\_meansPLOT1 <- k\_meansPLOT1 + theme(legend.direction = 'horizontal',

legend.position = 'top') +

ggtitle('Variants grouped by k-means clusters')

output$plot1 <- renderPlot({

k\_meansPLOT1

})

# kmeans clustering plot ## color points by class

k\_meansPLOT2 <- ggbiplot(pc,

obs.scale = 1,

var.scale = 1,

groups = as.factor(df1$binary\_clinvar\_class),

ellipse = TRUE,

circle = TRUE,

ellipse.prob = 0.68)

k\_meansPLOT2 <- k\_meansPLOT2 + scale\_color\_discrete(name = '')

k\_meansPLOT2 <- k\_meansPLOT2 + theme(legend.direction = 'horizontal',

legend.position = 'top') +

ggtitle('Variants grouped by clinvar class')

output$plot2 <- renderPlot({

k\_meansPLOT2

})

kCluster2 <- kmeans(data\_scaled\_MAVEonly, input$num, nstart = input$num2)

str(kCluster2) #troubleshooting

str(no\_na\_MAVEonly\_df) #troubleshooting

no\_na\_MAVEonly\_df$kMeansCluster <- kCluster2$cluster

print('testing4') # troubleshooting

######################

######################

###################### GFMM # not working right now, troubleshoot eventually

# maybe with Shiny add way for user to select default or EDDA

# on scaled data ## default

#if (input$radio == 1) {

#mod <- MclustDA(data\_scaled, df1$binary\_clinvar\_class)

#summary(mod)}

# on scaled data ## EDDA

#if (input$radio == 2) {

#mod <- MclustDA(data\_scaled, df1$binary\_clinvar\_class, modelType = "EDDA")

#summary(mod)}

##cross validation for GFMM

#cv <- cvMclustDA(mod, nfold = 10)

#unlist(cv[3:6])

# ce se.ce brier se.brier

# 0.04347826 0.01653451 0.03763904 0.01277613

######################

######################

###################### naive bayes

# make dataframe with class

str(data\_scaled)

data\_scaled\_df2 <- as.data.frame(data\_scaled)

data\_scaled\_df2$class <- as.factor(df1$binary\_clinvar\_class)

str(data\_scaled\_df2)

# train test split

set.seed(input$whatSeed)

ind <- sample(2, nrow(data\_scaled\_df2), replace = T, prob = c(input$num3, 1-input$num3))

train <- data\_scaled\_df2[ind == 1,]

test <- data\_scaled\_df2[ind == 2,]

str(train)

str(test)

# drop class from train ## crap gotta fix this, can't hard code with Shiny app!!!

drops <- c("class")

train\_noClass <- train[ , !(names(train) %in% drops)]

test\_noClass <- test[ , !(names(test) %in% drops)]

# train model

#model <- naive\_bayes(class ~ ., data = train, usekernel = T) # previous implementation of naive bayes

model = train(train\_noClass,train$class,'nb',trControl=trainControl(method='cv',number=3)) # current implementation with cross-validation

model

plot(model)

# predict

p <- predict(model, train, type = 'prob')

head(cbind(p, train))

# CM

p1 <- predict(model, train)

train\_noClass\_df2 <- train\_noClass

train\_noClass\_df2$NBpred <- p1

tab1 <- table(p1, train$class)

tab1

1 - sum(diag(tab1)) / sum(tab1)

p2 <- predict(model, test)

test\_noClass\_df2 <- test\_noClass

test\_noClass\_df2$NBpred <- p2

tab2 <- table(p2, test$class)

tab2

1 - sum(diag(tab2)) / sum(tab2)

p <- predict(model, data\_scaled\_df2, type = 'prob')

str(p)

p\_df <- as.data.frame(p)

p\_df$variant <- df1$hgvs\_pro

p\_df$class <- df1$binary\_clinvar\_class

str(p\_df)

no\_na\_df$NBpred <- p

# make dataframe with class

data\_scaled\_MAVEonly\_df <- as.data.frame(data\_scaled\_MAVEonly)

data\_scaled\_MAVEonly\_df$class <- as.factor(vector(mode="character", length=length(rownames(data\_scaled\_MAVEonly\_df))))

str(data\_scaled\_MAVEonly\_df) # troubleshooting

p <- predict(model, data\_scaled\_MAVEonly\_df, type = 'prob')

str(p) # troubleshooting

no\_na\_MAVEonly\_df$NBpred <- p

print('testing5') # troubleshooting

######################

######################

###################### random forest in R

# going to try slight different implementation of RF

## based on https://rpubs.com/jvaldeleon/forest\_repeat\_cv

str(train)

## Define repeated cross validation with 3 folds and three repeats

repeat\_cv <- trainControl(method='repeatedcv', number=3, repeats=3)

forest <- train(

# Formula. We are using all variables to predict Species

#class~.,

# Source of data; remove the Species variable

train\_noClass,

# outcomes

train$class,

# `rf` method for random forest

method='rf',

# Add repeated cross validation as trControl

trControl=repeat\_cv,

# Accuracy to measure the performance of the model

metric='Accuracy',

# adjust number of trees

ntree=input$num4)

#bestmtry <- tuneRF(train,train$class,ntreeTry=input$num4,stepFactor = input$num5, improve = input$num6, trace=T, plot= T)

#model <- randomForest(class~.,data= train)

print(forest$finalModel)

model <- forest

#importance(model)

#varImpPlot(model)

#pred\_test <- predict(model, newdata = test, type= "class")

#pred\_test

str(test)

#confusionMatrix(table(pred\_test,test$class))

pred\_train\_prob <- predict(model, newdata = train, type= "prob")

rf\_train\_df <- as.data.frame(cbind(train,pred\_train\_prob))

pred\_test\_prob <- predict(model, newdata = test, type= "prob")

rf\_test\_df <- as.data.frame(cbind(test,pred\_test\_prob))

data\_scaled\_df <- as.data.frame(data\_scaled)

predict\_RFall <- predict(model, newdata = data\_scaled\_df, type= "prob")

predict\_RFall\_df <- as.data.frame(predict\_RFall)

print('testingBEFORE')

rf\_cv <- rfcv(train\_noClass, train$class, cv.fold=3, step=0.5)

print('testingAFTER')

no\_na\_df$RFpred <- predict\_RFall\_df$P

data\_scaledMAVEonly\_df <- as.data.frame(data\_scaled\_MAVEonly)

predict\_RFall2 <- predict(model, newdata = data\_scaledMAVEonly\_df, type= "prob")

predict\_RFall2\_df <- as.data.frame(predict\_RFall2)

str(predict\_RFall2\_df)

no\_na\_MAVEonly\_df$RFpred <- predict\_RFall2\_df$P

print('testing6') # troubleshooting

#######################

#######################

#######################

# write loop to generate ROC curve for each individual MAVE

## and also the integrated dataset

# make a new subset df with added PC1, GFMM?, naive bayes, and random forest

## not sure how to make probabilites with GFMM, similar problem to kmeans clustering

### so leave off here, but can include output or plots using GFMM / kmeans

df\_subset2 <- df\_subset

df\_subset2$PC1 <- df\_wPC1$PC1

df\_subset2$NB <- p\_df$P

df\_subset2$RF <- predict\_RFall\_df$P

#run on first column

ROC\_firstCol <- roc(as.factor(df1$binary\_clinvar\_class), df\_subset2[[1]])

my.coords <- coords(ROC\_firstCol, "best", ret = "all", transpose = FALSE)

# setup empty dataframe to accept data within loop # add 25th column for AUC

big.coords\_df <- data.frame(matrix(ncol=25,nrow=length(colnames(df\_subset2))))

colnames(big.coords\_df) <- c(colnames(my.coords),'ROC\_AUC') # add AUC column

rownames(big.coords\_df) <- colnames(df\_subset2)

#str(big.coords\_df)

# can i use a list or vector of plots to make ROC curves?

plot\_vector <- vector(mode = "list", length = length(colnames(df\_subset2)))

# loop through each MAVE and calculate ROC curve

for (i in colnames(df\_subset2)){

ROC\_loop <- roc(as.factor(df1$binary\_clinvar\_class), df\_subset2[[i]])

AUC\_loop <- auc(ROC\_loop)

# the next two lines work to produce ROC plots locally in Rstudio, but stumped as to how to make this work with Shiny

#plot\_vector[i] <- plot(ROC\_loop, col="purple", legacy.axes=T, main=rownames(big.coords\_df[i,])) # comment this out

#plot\_vector[i] # comment this out

my.coords\_loop <- coords(ROC\_loop, "best", ret = "all", transpose = FALSE)

big.coords\_df[i,] <- my.coords\_loop

big.coords\_df[i,25] <- AUC\_loop

#print(big.coords\_df[i,])

#counter<-counter+1

#print(counter)

}

# lets develop model metrics specific for train and test splits

# store NB threshold

NBthreshold <- big.coords\_df["NB",1]

ROC\_NB\_train <- roc(as.factor(train$class), as.numeric(train\_noClass\_df2$NBpred))

#plot(ROC\_NB\_train, col="purple", legacy.axes=T, main="ROC\_NB\_train") #troubleshooting

my.coords\_NB\_train <- coords(ROC\_NB\_train, as.numeric(NBthreshold), ret = "all", transpose = FALSE)

print(my.coords\_NB\_train) #troubleshooting

ROC\_NB\_test <- roc(as.factor(test$class), as.numeric(test\_noClass\_df2$NBpred))

my.coords\_NB\_test <- coords(ROC\_NB\_test, as.numeric(NBthreshold), ret = "all", transpose = FALSE)

print(my.coords\_NB\_test) #troubleshooting

# store RF threshold

RFthreshold <- big.coords\_df["RF",1]

print(str(rf\_train\_df))

ROC\_RF\_train <- roc(as.factor(train$class), as.numeric(rf\_train\_df$P))

#plot(ROC\_NB\_train, col="purple", legacy.axes=T, main="ROC\_NB\_train") #troubleshooting

my.coords\_RF\_train <- coords(ROC\_RF\_train, as.numeric(RFthreshold), ret = "all", transpose = FALSE)

print(my.coords\_RF\_train) #troubleshooting

ROC\_RF\_test <- roc(as.factor(test$class), as.numeric(rf\_test\_df$P))

my.coords\_RF\_test <- coords(ROC\_RF\_test, as.numeric(RFthreshold), ret = "all", transpose = FALSE)

print(my.coords\_RF\_test) #troubleshooting

# make a dataframe

test\_train\_split\_metrics <- as.data.frame(rbind(my.coords\_NB\_train,my.coords\_NB\_test,my.coords\_RF\_train,my.coords\_RF\_test))

rownames(test\_train\_split\_metrics) <- c("NB\_train","NB\_test","RF\_train","RF\_test")

# how to add ROC curves to Shiny app

## wish I could overlay like code below, but probably going to have to display separately

#plot(ROC\_DN\_reporter, col="purple", legacy.axes=T,

# main="Individual assays")

#lines(ROC\_WT\_nutlin, col = "red")

#lines(ROC\_Etoposide, col = "green")

#lines(ROC\_null\_nutlin, col = "blue")

#print(model$results$Kappa) # troubleshooting

print('testing7') # troubleshooting

#######################

#######################

####################### calculate OddsPath

# add correct pseudocount adjustments

## thank you Malvika!!

# Pseudocount adjustments

# if assay\_abnormal\_num == total\_assay\_true\_path:

# assay\_abnormal\_num += 1

# if total\_assay\_true\_ben == 0 and benign\_controls > 1:

# total\_assay\_true\_ben += 1

# OddsPath = [P2 \* (1 - P1)] / [(1 - P2) \* P1]

# calculate OddsPath

# pre-loop

# count number of path and number of benign

path\_count <- length(which(df1$binary\_clinvar\_class == "P"))

benign\_count <- length(which(df1$binary\_clinvar\_class == "B"))

## set variables not specific to MAVE or integration method

total\_controls <- path\_count + benign\_count

## calculate prior probability

prior\_prob\_path <- path\_count / total\_controls

prior\_prob\_benign <- benign\_count / total\_controls

# loop

# setup empty dataframe to accept data within loop

oddsPath\_path\_vec <- c()

oddsPath\_benign\_vec <- c()

counter=1 # not sure if I need a counter

# need to fix this

for (i in colnames(df\_subset2)){

# set MAVE-specific or integration method-specific variables

TotalAssayAbnormal <- big.coords\_df$tp[counter] + big.coords\_df$fp[counter]

TotalAssayNormal <- (big.coords\_df$tn[counter]+big.coords\_df$fn[counter])

TruePathInAbnormal <- big.coords\_df$tp[counter]

TruePathInNormal <- big.coords\_df$fn[counter]

#AssayFalseNegative <- big.coords\_df$fn[counter]

# Pseudocount adjustments

# if assay\_abnormal\_num == total\_assay\_true\_path:

# assay\_abnormal\_num += 1

if (TotalAssayAbnormal == TruePathInAbnormal) {

TotalAssayAbnormal = TotalAssayAbnormal + 1

}

# if total\_assay\_true\_ben == 0 and benign\_controls > 1:

# total\_assay\_true\_ben += 1

if (TruePathInNormal == 0 & benign\_count>1) {

TruePathInNormal = 1

}

ProportionPathInAbnormal <- TruePathInAbnormal / TotalAssayAbnormal

ProportionPathInNormal <- TruePathInNormal / TotalAssayNormal

# calculate posterior probability

# pull from ROC

#post\_prob\_path <- TruePathInAbnormal / TotalAssayAbnormal

#post\_prob\_path <- big.coords\_df$tp[counter] / (big.coords\_df$tp[counter] + big.coords\_df$fp[counter] + 1)

#post\_prob\_path <- big.coords\_df$tp[counter] / (big.coords\_df$tp[counter] + big.coords\_df$fp[counter])

#post\_prob\_benign <- big.coords\_df$fn[counter] / (big.coords\_df$tn[counter]+big.coords\_df$fn[counter])

#post\_prob\_benign <- AssayFalseNegative / (big.coords\_df$tn[counter]+big.coords\_df$fn[counter])

# troubleshooting

#print(post\_prob\_path)

#print(post\_prob\_benign)

#oddsPath\_path\_vec[counter] <- (post\_prob\_path \* (1 - prior\_prob\_path)) / ((1 - post\_prob\_path) \* prior\_prob\_path)

oddsPath\_path\_vec[counter] <- (ProportionPathInAbnormal \* (1 - prior\_prob\_path)) / ((1 - ProportionPathInAbnormal) \* prior\_prob\_path)

# troubleshooting

#print(oddsPath\_path\_vec[counter])

oddsPath\_benign\_vec[counter] <- (ProportionPathInNormal \* (1 - prior\_prob\_path)) / ((1 - ProportionPathInNormal) \* prior\_prob\_path)

counter=counter+1

}

# add OddsPath to output csv

big.coords\_df$OddsPath\_path <- oddsPath\_path\_vec

big.coords\_df$OddsPath\_benign <- oddsPath\_benign\_vec

#

OddsPath\_PLOT1 <- ggplot(big.coords\_df, aes(x=rownames(big.coords\_df),y=OddsPath\_path, fill=as.factor(rownames(big.coords\_df)))) +

geom\_col() + theme\_bw() + xlab("") +

theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),legend.position="none")

output$plot3 <- renderPlot({

OddsPath\_PLOT1

})

OddsPath\_PLOT2 <- ggplot(big.coords\_df, aes(x=rownames(big.coords\_df),y=OddsPath\_benign, fill=as.factor(rownames(big.coords\_df)))) +

geom\_col() + theme\_bw() + xlab("") +

theme(axis.text.x = element\_text(angle = 90, vjust = 0.5, hjust=1),legend.position="none")

output$plot4 <- renderPlot({

OddsPath\_PLOT2

})

# this works but plots too small, can make bigger???

## lets only use for output svg

#svglite(filename = "/Users/jeffreycalhoun/Downloads/Rplots.svg", width = 10, height = 8,

# bg = "white", pointsize = 12, scaling = 1)

#grid.arrange(k\_meansPLOT1, k\_meansPLOT1, OddsPath\_PLOT1,OddsPath\_PLOT2,nrow = 2)

#dev.off()

#

#output$NB\_intro <- renderText({ "Naive Bayes Kappa with 3-fold CV:" })

#output$NB\_cv <- renderText({ model$results$Kappa[1] })

output$RF\_intro <- renderText({ "Random Forest 3 fold CV error:" })

output$RF\_cv <- renderText({ rf\_cv$error.cv[1] })

print('testing8') # troubleshooting

#######################

#######################

#######################

output$downloadData <- downloadHandler(

filename = function() {

paste("data-Metrics-", Sys.Date(), ".csv", sep="")

},

content = function(file) {

write.csv(big.coords\_df, file)})

output$downloadData2 <- downloadHandler(

filename = function() {

paste("data-VariantLevelTS-", Sys.Date(), ".csv", sep="")

},

content = function(file) {

write.csv(no\_na\_df, file)})

output$downloadData3 <- downloadHandler(

filename = function() {

paste("data-VariantLevelALL-", Sys.Date(), ".csv", sep="")

},

content = function(file) {

write.csv(no\_na\_MAVEonly\_df, file)})

output$downloadData4 <- downloadHandler(

filename = function() {

paste("data-MetricsTrainingTestSplits-", Sys.Date(), ".csv", sep="")

},

content = function(file) {

write.csv(test\_train\_split\_metrics, file)})

output$downloadPlot <- downloadHandler(

#svglite(filename = "test.svg", width = 10, height = 8,

# bg = "white", pointsize = 12, scaling = 1)

filename = function(){paste("outputPlotsPC",Sys.Date(),'.svg',sep='')},

content = function(file){

ggsave(file,plot=grid.arrange(k\_meansPLOT1, k\_meansPLOT2,nrow = 2,heights=c(4,4),widths=c(4)),width=12,height=10,dpi=300,units=c("in"))

}

)

output$downloadPlot2 <- downloadHandler(

#svglite(filename = "test.svg", width = 10, height = 8,

# bg = "white", pointsize = 12, scaling = 1)

filename = function(){paste("outputPlotsOddsPath",Sys.Date(),'.svg',sep='')},

content = function(file){

ggsave(file,plot=grid.arrange(OddsPath\_PLOT1,OddsPath\_PLOT2,nrow = 2,heights=c(2,2)),width=8,height=10,dpi=300,units=c("in"))

}

)

output$downloadPlot3 <- downloadHandler(

#svglite(filename = "test.svg", width = 10, height = 8,

# bg = "white", pointsize = 12, scaling = 1)

filename = function(){paste("outputPlotsROCcurve",Sys.Date(),'.svg',sep='')},

content = function(file){

ggsave(file,plot=ROCall,width=8,height=8,dpi=300,units=c("in"))

}

)

})

}

# Create Shiny app ----

shinyApp(ui, server)