Alzheimer’s Project

Dustin Miller

4/18/2018

knitr::opts\_chunk$set(eval = FALSE)

Function will remove columns and transpose data to be used for Machine Learning. The input file for this function is the output file from annotatedVCFModifyer.py.

makeAnnotatedVcfTidy <- function(inputFile, outputName) {  
 library(tidyverse)  
 #import file  
 gettinrTidy <- read\_tsv(inputFile)  
 print("File read complete. Transforming data to be used for Machine Learning...")  
 #remove particular columns  
 gettinrTidy <- select(gettinrTidy, -c(CHROM, POS, REF, ALT, QUAL, FILTER, INFO, FORMAT, ID))  
 #transpose  
 gettinrTidy <- t(gettinrTidy)  
 #set row name as a new column  
 gettinrTidy <- cbind(ID = rownames(gettinrTidy), gettinrTidy)  
 #write file  
 gettinrTidy <- as\_data\_frame(gettinrTidy)  
 print("Transformation complete, you are so close. Writing file...")  
 write\_tsv(gettinrTidy, outputName)  
 print("Complete! Happy Tidy-ing! Next step, add 'Class' to file in second column.")  
}

Divide data into a test and validation set. Modify variable names as needed. The input for this was the output from the addClassToTidyTsv.py script.

library(tidyverse)  
#Read file  
classFile <- read\_tsv(Path to File)  
#Filter for controls only  
controls <- filter(classFile, Class == "CONTROL")  
#Randomly sample .75 of controls to a test set  
controlsTest <- sample\_frac(controls, .75)  
#Remaining .25 of controls assigned to val set  
controlsVal <- anti\_join(controls, controlsTest)  
#Filter for cases only  
cases <- filter(classFile, Class == "CASE")  
#Randomly sample .75 of cases to a test set  
casesTest <- sample\_frac(cases, .75)  
#Remaining .25 of controls assigned to val set  
casesVal <- anti\_join(cases, casesTest)  
#Combine case and control test sets into a combined test set, and randomize rows  
testSet <- bind\_rows(casesTest, controlsTest) %>% sample\_frac(1)  
#Combine case and control val sets into a combined val set, and randomize rows  
valSet <- bind\_rows(casesVal, controlsVal) %>% sample\_frac(1)  
#write files  
write\_tsv(testSet, output file name and path)  
write\_tsv(valSet, output file name and path)

This function will import the ShinyLearner metric file and average accuracy.

importSlMetricThenAverageAccuracy <- function(metricFile) {  
 library(tidyverse)  
 #Load in algorithm metric files and parse for rows with "Accuracy" and average  
 metricFile <- read\_tsv(metricFile)  
 metricFile <- filter(metricFile, Metric == "Accuracy")  
 metricFile <- round(mean(metricFile$Value), 4)  
 return(metricFile)  
}

This function will import the ShinyLearner elapsed time file and sum time across iterations.

importSlTimeThenSumIterations <- function(elapsedTimeFile, unit = 1) {  
 library(tidyverse)  
 #Load in algorithm elapsed time files and sum elapsed time in seconds unless otherwise specified  
 timeFile <- read\_tsv(elapsedTimeFile)  
 timeFile <- (sum(timeFile$ElapsedSeconds) / unit)  
 return(timeFile)  
}

Function will create a data frame with averaged prediction rate and completion time for each algorithm

dataFrameMlData <- function(algorithm = c("mlr: svm", "scikit-learn: random forest", "weka: naive Bayes", "scikit-learn: k-NN", "scikit-learn: multilayer perceptron", "scikit-learn: logistic regression", "mlr: xgboost"), avgAccuracy = c(), time =c()) {  
   
 mlDf <- data\_frame(algorithm, avgAccuracy, time)  
 return(mlDf)  
}

This code was used to make dataframes and plot data

#averageAccuracy is a vector of all the files imported with importSlMetricThenAverageAccuracy  
#time is a vector of all the files imported with importSlTimeThenSumIterations  
newDataFrame <- dataFrameMlData(avgAccuracy = c(), time = c())  
#Make a scatterplot  
ggplot(data = newDataFrame) + geom\_point(mapping = aes(x = time, y = avgAccuracy, fill= algorithm, colour = algorithm)) + labs(x = "Elapsed Time", y = "Prediction Accuracy") + ylim(0.485, 0.515) + theme\_bw(base\_size = 16) + #theme(plot.title = element\_text(hjust = 0.5)) +   
 geom\_hline(yintercept=0.5109, linetype="dashed", color = "red")  
ggsave("name of output.png", plot = last\_plot(), width = 8, height = 4, dpi = 500)  
  
  
install.packages(MikTeX)