Hannah Lerner
4/19/17

Biomarker Gene Analysis: The Future of Cancer Detection

Background:

When attempting to detect lung cancer, current procedures involve certain invasive tests such as a bronchoscopy screening, a lung biopsy, or even surgery. Each of these tests are very expensive to perform and incredibly inconvenient for the patient. There is a new method of detecting lung cancer that is currently under development. This new method would make the more invasive tests less crucial to the detection of the cancer[1]. The goal of this new method is to use a patient's gene expression sample (RNA) to find certain patterns that determine whether or not the patient has cancer. The gene analysis method requires a statistical analysis program to run through all of the gene combinations to try and find the most useful genes to focus on. Once the genes are selected, they are processed through an analysis pipeline which will take each of the factors into account and create an estimate on the likelihood of the patient having lung cancer. If the estimate proves to be sufficiently accurate, it can be moved into general use on patients to help diagnose lung cancer early in people with high-risk genes. The estimates are checked against data that has a strong impact on gene expression, such as a patient's smoking status. Once the analysis is sufficiently accurate, it can then be used to interpret much less certain data.

In order to make sure the analysis method works universally; the data is split into 10 cross-validation loops. Each cv loop contains approximately 20% of the entire data set. This ensures that the method currently being tested will work for more than one data set. It works by tricking the program into thinking that each new data set is completely different[2], and therefore cannot be looked at the same way. If the data is too similar, the program can sometimes run into the problem of only working for that one set of data. This scenario is known as overfitting a dataset. In this case, the results can initially look very promising, but only for the model data presented. If the model is then shifted to a new set of data, it can have a significantly lower accuracy rating. In the case of smoking status, the data leaves clear tracks in its gene expression. This helps guarantee that the cv loops are properly calibrated and will not return false data.

The analysis of these genes is processed in the programming language R using Rstudio. Rstudio is a free open source program that allows the user to edit and run scripts in R. Its syntax is very similar to Python, but it has significantly better tools to produce tables and graphs.

The data comes from a default "smoking status" analysis. The pipeline analysis program takes the raw score, which is calculated from a certain gene expression, and creates a prediction as to whether or not the patient is a current smoker A prediction of 0 indicates that the patient is a non-smoker, while a prediction of 1 would suggest that the patient is a smoker.

---

[1] The invasive tests will likely be necessary if the new tests return positively, however, they will no longer be necessary if the new tests are negative.

[2] The cross-validation process works by taking 80% of the data to develop a biomarker, then testing how well the biomarker does on the remaining 20% of samples.

The next stage of this analysis is to calculate the Area Under the Curve (AUC)[3] for each of the cross-validation loops (cv_loop). The sensitivity and specificity at each possible value for the biomarker must be calculated before the AUC can be determined. The sensitivity is calculated by calculating the number of true positives in the data divided by the sum of the true positives and the false negatives. A true positive would be when the program accurately predicts that a patient is a current smoker. A false negative would be if the program predicts a current smoker to not be smoking. In this table, the sensitivity is displayed under the column "asens".

The specificity is calculated by dividing the number of true negatives by the sum of true negatives and false positives. A true negative would be when the program accurately predicts that a patient is not currently a smoker. A false positive would be if the program falsely believes that a non-smoker is currently a smoker. From here, a Receiver Operating Characteristic Curve (ROC Curve)[4] is created by graphing the sensitivity and 1-specificity. In this case the prediction was very accurate as it was using the smoking status data. This resulted in a ROC curve with a very high AUC value.

Introduction:

People have been fighting lung cancer for thousands of years. While it had not been defined as a disease until 1761, lung cancer has been found in people from prehistoric times. The first reported "case" of lung cancer was from Egypt. Lung cancer was not well documented, however, until after World War I, when smoking became popularized. Lung cancer was not linked to smoking until 1929. Before then, miners were dying at a rate of 60-80% in certain mines. These mines were so hostile that they became known as "death pits." By 1924, the mystery in the mines had been discovered and radon was found to be the culprit of the deaths. Radon is a carcinogen, which means that it will increase the likelihood of cancer. In this case, radon is linked to causing lung cancer. By 1940, lung cancer had become the second most diagnosed case of cancer. This diagnosis rate of lung cancer was second only to that of stomach cancer.

Today, lung cancer is the second most diagnosed cancer in patients. In fact, about 18% of all cancer diagnosed is lung cancer. It is also responsible for 27% of all cancer deaths. It is most common in patients above the age of 65. There are three major forms of lung cancer: small cell cancer, non-small cell cancer, and lung carcinoid tumor. Non-small cell cancer is by far the most commonly diagnosed form of cancer. In fact, 85% of all diagnosed lung cancers tend to be non-small cell cancer. Small cell lung cancer, while not as common at a rate of 15% of all diagnosed lung cancers, is the fastest spreading lung cancer of the three. All three types of lung cancer appear to be exacerbated by smoking, but smoking appears to have a much more detrimental effect on patients diagnosed with small cell or non-small cell cancer.

Today, if someone is at risk of developing lung cancer, they must undergo different tests and scans. The three most common tests used in preliminary screening are a sputum cytology, a chest x-ray, or a computed tomography (CT) scan. The CT scan is the most detailed of the three

---

[3] The area under a ROC curve represents the accuracy of the prediction model. A high AUC value suggests that the True positive and true negative prediction rates are high and the model is successful overall in predicting accurate outcomes.

[4] Related to the AUC, the ROC curve is the actual function calculated by the relationship of the sensitivity and 1-specificity.

tests, but still often fails to distinguish cancerous abnormalities from a benign tumor. These scans can only indicate whether or not there is an abnormality in the body. Doctors must take a biopsy[5] of the tissue in order to determine the severity of the abnormality. This process can take up to 10 days to receive a definitive result and can be very invasive. Gene analysis, when tuned properly, can give a much more accurate, and more specific diagnosis for a patient than any scan or preliminary test will do given the same sample.

The gene analysis software works by running raw data through a biomarker[6] pipeline. These pipelines are designed to select certain genes from the sample and determine whether or not their expressions are significant for the given test. It is not yet known what each gene accounts for within the samples, but certain combinations appear to indicate certain conditions. The current problem researchers are running into with these genes is that individual gene expressions are not always significant, but when a series of genes are expressed in a certain way, the group can be significant, even when the individual genes are not.

The programs that create these biomarker pipelines are written in a programming language called R. R is a programming language used primarily for statistical computing. Like python and java, R is an object-oriented programming language. This means that those who work on the code control the data type of a data structure[7] as well as the functions [8]that can be applied to the data structure. The experiment was trying to test the effectiveness of four different methods of refining the pipeline algorithm: Feature Filter, Feature Selection, Biomarker Size, and the Classifier.

Methods:

This method section will help someone with little to no experience in R create a similar experiment. Last summer started out with a month-long lesson in R. Having no knowledge of the language before then, I had to read through a book for beginners on R. After that, I learned basic indexing of large databases[9]. R has a fairly steep learning curve as there are a lot of built-in functions that are vital to know in order to produce the desired output.

For this particular experiment, the program would need to sort through approximately 70,000 rows and 60 columns of data. When dealing with this amount of data, you need to be

---

[5] A biopsy is a small sample of tissue taken from a patient to conduct lab tests on.

[6] Biomarkers can be characteristic biological properties or molecules that can be detected and measured in parts of the body like the blood or tissue. They may indicate either normal or diseased processes in the body. Biomarkers can be specific cells, molecules, or genes, gene products, enzymes, or hormones.

[7] A data structure is a method people use to organize data into a certain arrangement that is convenient for whoever created it.

[8] A function is a tool created by a programmer designed to complete a simple or complex task.

[9] Ania Tassinari helped me with this. She spent a great deal of her time helping learn the language, and it is because of her that it only took a couple weeks to learn the necessary R for the program.

careful how you interpret it. If you take too many pieces in at once, the entire program will crash. Even worse so, it could run and give you inaccurate data[10].

The first program I wrote to try and sort through the data attempted to analyze the data as a whole. The original goal of this initial program was to create a template for all of the calculations. The first problem arose when I tried to run the data through the program all at once. Apparently Rstudio (the IDE I was using) can only handle so many inputs to a function before crashing. Unfortunately, because of this, I ended up not being able to properly test the entire data set using this program.

The second iteration of the biomarker pipeline output program was designed to split the data into 10 smaller groups before analysis. These groups were determined in the original pipeline output script (written by Joe Perez-Rogers). The 10 smaller groups were known as the cross-validation loops, or cv loops. With the data split up into smaller groups, this script took existing data files from my computer and organized them into a properly labeled table. This method still required the program to sort through all of the data, but it allowed the actual calculations to be made on significantly smaller sampling sizes. Once the data was properly summarized into tables that calculated the average AUC across cv loops for each model, the program goes on to try and graph the calculated data. Most of the graphing functions are built into R which made graphing these sets of data fairly trivial. The final output graphs of this function are boxplots of each of the desired features to be tested for accuracy as well as a ROC curve for the overall accuracy of the data.

The final iteration of this program simply cleaned up the second iteration as well as allow the user to customize their testing conditions more easily. It turns the script into a function, allowing the user to pick the file paths from their computer as well as the number of cross validation loops and models that the user wishes to test in this current iteration. To create this new arrangement, the program was modified to allow the user to declare the entire path that the program reads from while calling the function. Before this change, the user would need to change the hard-coded method each time they wanted to read in a new file. This new method is also ideal as it prevents the necessity of extraneous libraries (such as plyer) which were necessary when the program was using methods that were a little too specific for its intended use. Another benefit to this generalization process of the third iteration was the removal of specific commands as well as the addition of more overarching ones. A good example of this would be the difference between lines 61 in the second iteration versus lines 83 to 85 of the third iteration. The major difference between these two different approaches is that the one used in the second iteration will only work for the smoking status data. As the table as a whole has about 60 different attributes, the limitation to just one of these is very inefficient and inconvenient. The method used in the third iteration only accepts two inputs. Of the two inputs used, the positive input must be 1, but the second input can be anything. It was created in this way as to allow the user to have some flexibility when using binary inputs. In most of the attributes tested, the input numbers were either 0 and 1, or 1 and 2. While this new method sounds limiting in its own way, the program is designed to compare whether or not a piece of data is true or false. The necessity

[10] When in the lab, this happened once. The returned numbers were extremely scattered and simply wrong for the most part. This problem was fixed when the program separated the data into smaller categories to sort through. This method was extremely ineffectual. It was attempting to analyze about 4 million pieces of data all at once. It originally made some attempt to interpret all of the data, but eventually crashed under the weight of it all.

of the positive value being one was a personal choice made to simplify the input for the user. It would be just as possible to require the user to decide what the positive and negative values are when calling the function, but small choices like this could lead to an overcomplicated program.

The final changes made between the second and third iteration were mostly polishing the program in an attempt to make it more legible for users who wish to modify or use it. The most notable of these changes lies in lines 188 to 189 of the third iteration. These lines allow the user to make a visual display of the analyzed data in a PDF. This particular program focuses primarily on boxplot data, so the changes allow the user to add labels to the plots as well. The second iteration was able to provide the raw data with the plots, but it could get confusing at times as to which tables referred to which features.

In addition to some of the more stylistic choices implemented, other variations of the choices are more or less accessible in the third iteration in the commented code. The addition of these few lines makes debugging any possible problems significantly easier. The comments also explain some of the reasoning to help the user decide which method they would prefer.

Initially, a biomarker pipeline was created in order to analyze gene samples from patients in an attempt to determine whether or not they were smoking at the time. These samples are initially taken from intrathoracic (bronchial) and intrathoracic (buccal/nasal) epithelial cells[11]. Bronchial samples are taken by moving a type of brush through the airway. This process can potentially give far more information than the existing tests do today.

Once the sample was taken, it was sequenced and the data was uploaded onto a text document for analysis. This raw data was then passed through a biomarker pipeline. In this case, the pipeline used was created by Joe Perez-Rogers. The pipeline works by taking a series or select sets of genes from the raw data and creating a prediction for each possible issue with a person. The genes analyzed changes from pipeline to pipeline, but the general process is the same. The output data is given in a series of 10 cross validation loops. These loops are designed to keep the pipeline from cheating itself into methods designed to a specific set of data. This output data are the predictions the patients receive. If there are any tests that come up positive, the patient can then go through a more rigorous process for detection or treatment.

The data from the pipeline is then taken and run through a program created over the summer to check its overall accuracy. This program reads the analyzed data displays it as a series of ROC curves[12] as well as calculate the AUC[13] for each cross-validation loop. If the predictions using a certain method of analyzing the data appear to be accurate enough, the method eventually becomes a new method to test for various symptoms of lung cancer[14]. The last step of checking the accuracy will be unnecessary once the pipeline can consistently return accurate predictions for the desired data sets.

---

[11] Bronchial cells are coming from the airway between the trachea and the bronchioles. Buccal cells are coming from the lining of the mouth. Nasal cells refer to the nose.

[12] A ROC curve, also known as a receiver operating characteristic curve is a diagram that plots the true positive rate of a test against the false positive rate.

[13] The AUC, also known as the area under the curve, an area calculation of the area under the ROC curve. In an ideal scenario, The AUC will be 1.

[14] A past method from this lab to detect squamous cell lung cancer has been published in nature due to its accuracy in detection.

Results and Analysis:

Figures 1-4 display the results of four different testing parameters: Feature Filter, Feature Selection, Biomarker Size, and the Classifier. The feature filter is testing two different approaches to filtering the genes to remove genes that are unlikely to be expressed and therefore not useful in predicting the smoking phenotype. The first filtering method is variance. Variance is calculated by taking the sum of the square of the differences between the observations and the mean, then taking that result and dividing by the data size (see equation 1). The other filtering method is the mean. The mean of a sample is calculated by taking a sum of all the sample inputs, and dividing by the sample size (see equation 2). The next set step in the biomarker pipeline tests different feature selection methods. The two major feature selection methods being tested are the linear model t-statistic, and the linear model absolute t-statistic. The other feature selection method is a random forest test. The next step in the biomarker pipeline is choosing the biomarker size (the number of genes whose expression is combined to make the ultimate prediction). The size of the biomarker is decided by the user, and for this experiment a size of 25, 50, and 100 genes were tested. The final step in the pipeline is evaluating different methods for combining the genes to make a prediction. Due to the predictability of the data set used, the changes in the AUC were generally very small, however, certain methods of prediction appeared to have clear advantages over the others.

The feature filter graph was the least impressive of the four graphs created as the median of both the variance as well as the mean was the same. The variance appears to have a slight advantage over the mean due to more crowding. The AUC values for both the mean and the variance were approximately 0.85.

Within the Feature selection graph, the difference in the median AUC values only differed by approximately 0.015. The linear model sorting by t-statistic had a median AUC value of 0.860, while the linear model sorting by absolute t-statistic had a median AUC value of 0.845. That being said, the linear model sorting by t-statistic had consistently similar predictions[15] than the linear model sorting by absolute t-statistic. The linear model sorting by t-statistic also had more crowding at higher AUC values than the linear model sorting by absolute t-statistic.

The results of the biomarker appear to improve as the size goes up. The base score of the 25-gene size biomarker still has exceptional results at about 0.847. The biomarker with a size of 50 genes has an AUC value of approximately 0.853. This is a noticeable improvement on the biomarker with a size of 25 genes, but still only a difference of 0.006. The biomarker with a size of 100 had the best results with an AUC value of 0.856. The difference between the biomarker with a size of 50 and the biomarker with a size of 100 is smaller than the difference in AUC between a biomarker with size 25 and a biomarker with size 50, but the distribution of the biomarker with size 100 is significantly closer together than the other two sizes. The range of the biomarker with a size of 25 genes has a range of about 0.02 AUC value and an IQR of about 0.011, and the range of the biomarker with a size of 50 genes has a range of about 0.018 AUC value and an IQR of 0.015, but the range of the biomarker with a size of 100 genes has a range of 0.005 AUC value and an IQR of 0.003 AUC value. Even with such a small scale, the difference in the distribution between the three sample sizes indicates that a larger biomarker size is ideal for predicting the outcomes.

[15] The predictions were closer together than the other method.

The difference in the AUC values for the classifier were minute. The naive bayes algorithm had an AUC value of approximately 0.854, while the winter variation algorithm had an AUC value of about 0.853. While there is a difference in the AUC value of 0.001, the algorithms perform so similarly that either can be used interchangeably for this particular case.

Conclusion:

When a person smokes, the smoke leaves lasting impressions on the gene expression of certain cells. There are certain, extremely-predictable gene expressions that occur. Because of this, the biomarker analysis of the smoking status of a person is generally very accurate. This data shows no exception to that fact. All of the different methods at improving the predictability of the smoking status only had an effect of up to a 0.02 difference in the AUC value.

Before this exit protocol, the biomarker pipeline results were displayed in large scale text files which made it hard to parse through the information quickly. The code written over the summer was designed to summarize the pipeline output in such a way that it can work with any type of data input to it with as few as possible modifications. At the moment, it only intakes binary values. The predictions created in this data set were designed to determine whether or not the "smoking status" of a patient was positive or negative. This can be useful for various aspects of lung cancer predictions, but is not optimal in terms of space efficiency. For example, if someone wanted to predict how severe a cancer was in a patient, not knowing if they even have it would take multiple columns in a table asking binary questions which would need to be parsed together at the end. If the program was only dealing with a few hundred patients, the space efficiency of this method would not matter, but if the sample size is in the thousands, or even hundreds of thousands. Each additional column will add an enormous amount of data to the table which could have been prevented if the program could run more than just binary operations.

Acknowledgements:

I would like to thank Joe Perez-Rogers, Boston University and Ania Tassinari, Boston University for assistance with analyzing the biomarker pipeline, and Marc Lenburg, Boston University for a position in the lab as well as very helpful comments while editing.

Appendix:

Table 1: Results table averages

Model	Feature Filter	Feature Selection	Biomarker Size	Classifier	avgsens	avgspec	AUC
1	var	lm-t	25	wv	0.7880417	0.9092853	0.8486635
2	var	lm-t	25	nb	0.8484751	0.8685301	0.8585026

3	var	lm-t	50	wv	0.8202818	0.8989402	0.8596110
4	var	lm-t	50	nb	0.8490915	0.8691749	0.8591332
5	var	lm-t	100	wv	0.8314800	0.8818618	0.8566709
6	var	lm-t	100	nb	0.8398719	0.8672320	0.8535519
7	var	lm-abst	25	wv	0.7634793	0.9183675	0.8409234
8	var	lm-abst	25	nb	0.8038260	0.8848942	0.8443601
9	var	lm-abst	50	wv	0.7814753	0.9099887	0.8457320
10	var	lm-abst	50	nb	0.8091362	0.8769364	0.8430363
11	var	lm-abst	100	wv	0.8143891	0.9003777	0.8573834
12	var	lm-abst	100	nb	0.8356146	0.8729428	0.8542787
13	mean	lm-t	25	wv	0.7888051	0.9092853	0.8490452
14	mean	lm-t	25	nb	0.8500018	0.8685301	0.8592659
15	mean	lm-t	50	wv	0.8195184	0.8989402	0.8592293

16	mean	lm-t	50	nb	0.8513941	0.8704909	0.8609425
17	mean	lm-t	100	wv	0.8337701	0.8818618	0.8578159
18	mean	lm-t	100	nb	0.8398478	0.8651965	0.8525222
19	mean	lm-abst	25	wv	0.7634793	0.9183675	0.8409234
20	mean	lm-abst	25	nb	0.8038260	0.8848942	0.8443601
21	mean	lm-abst	50	wv	0.7814753	0.9093394	0.8454073
22	mean	lm-abst	50	nb	0.8098755	0.8762697	0.8430726
23	mean	lm-abst	100	wv	0.8158567	0.9004304	0.8581436
24	mean	lm-abst	100	nb	0.8386570	0.8716268	0.8551419

Appendix:

Equation 1: Variance of a sample

$$s^2 = \frac{\sum(X_i - \overline{X})^2}{n-1}$$

Equation 2: Mean of a sample

$$\overline{X} = \frac{\sum X_i}{n}$$

Boxplots of data:

Figure 1:

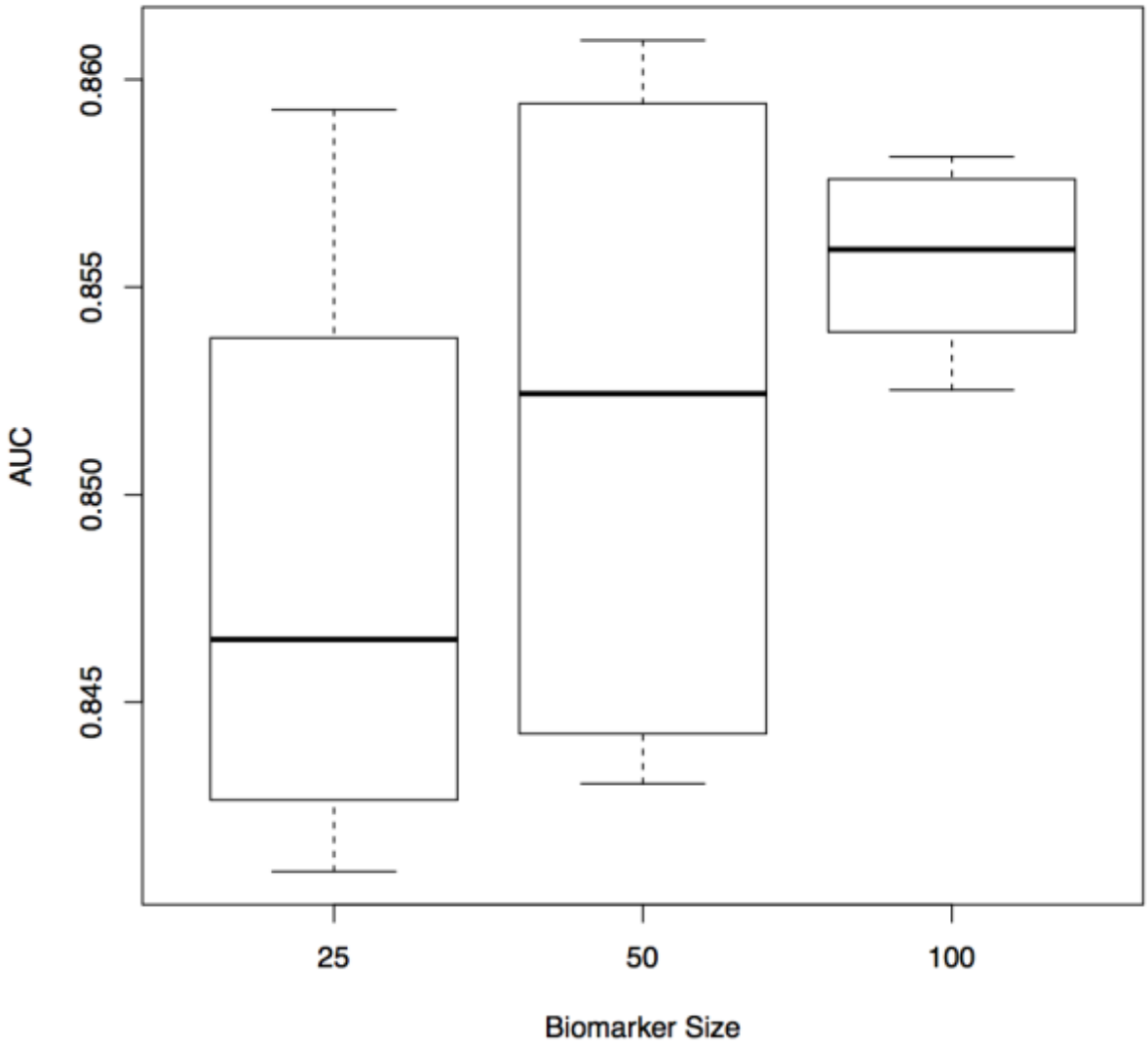


Figure 2:

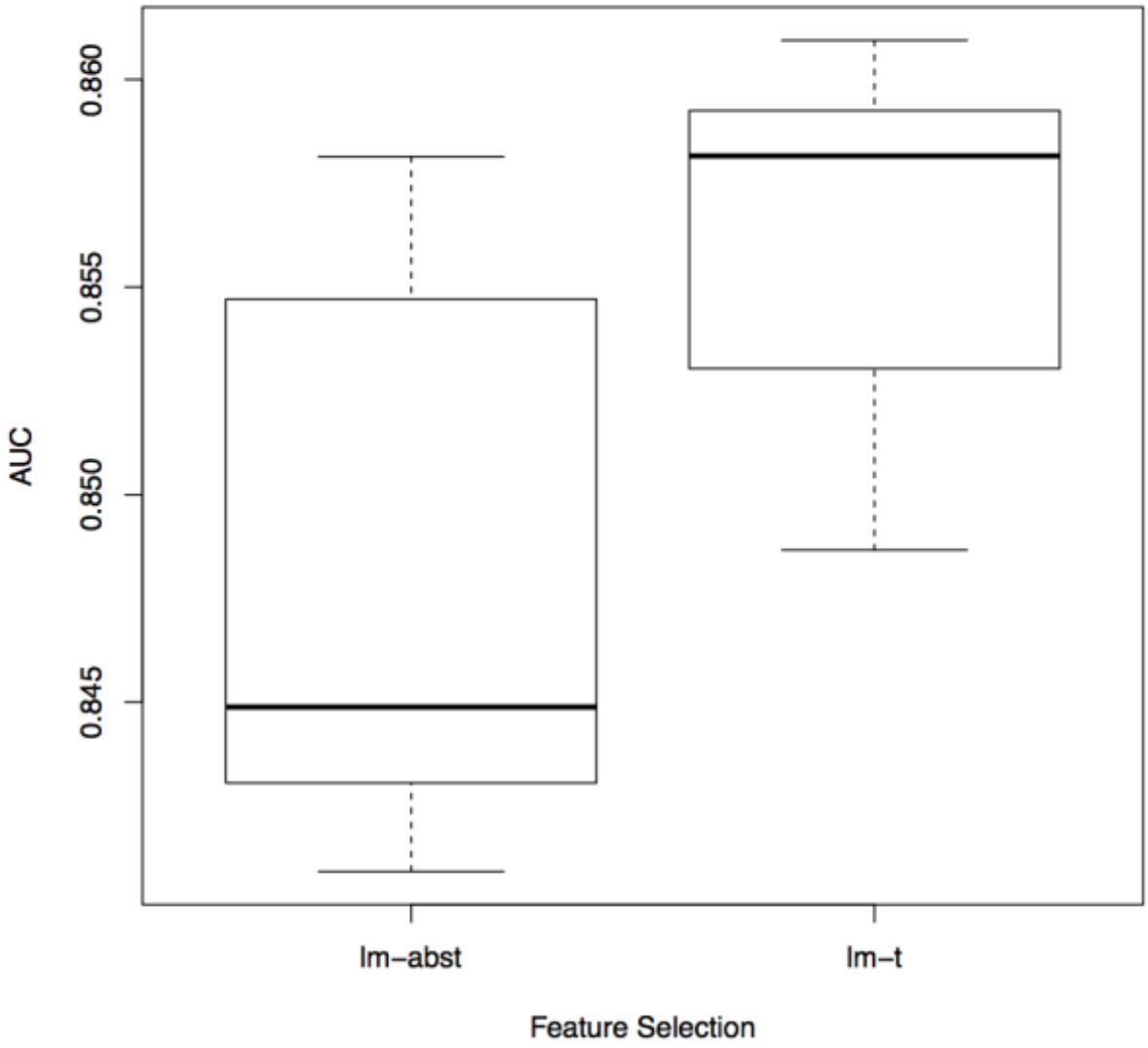


Figure 3:

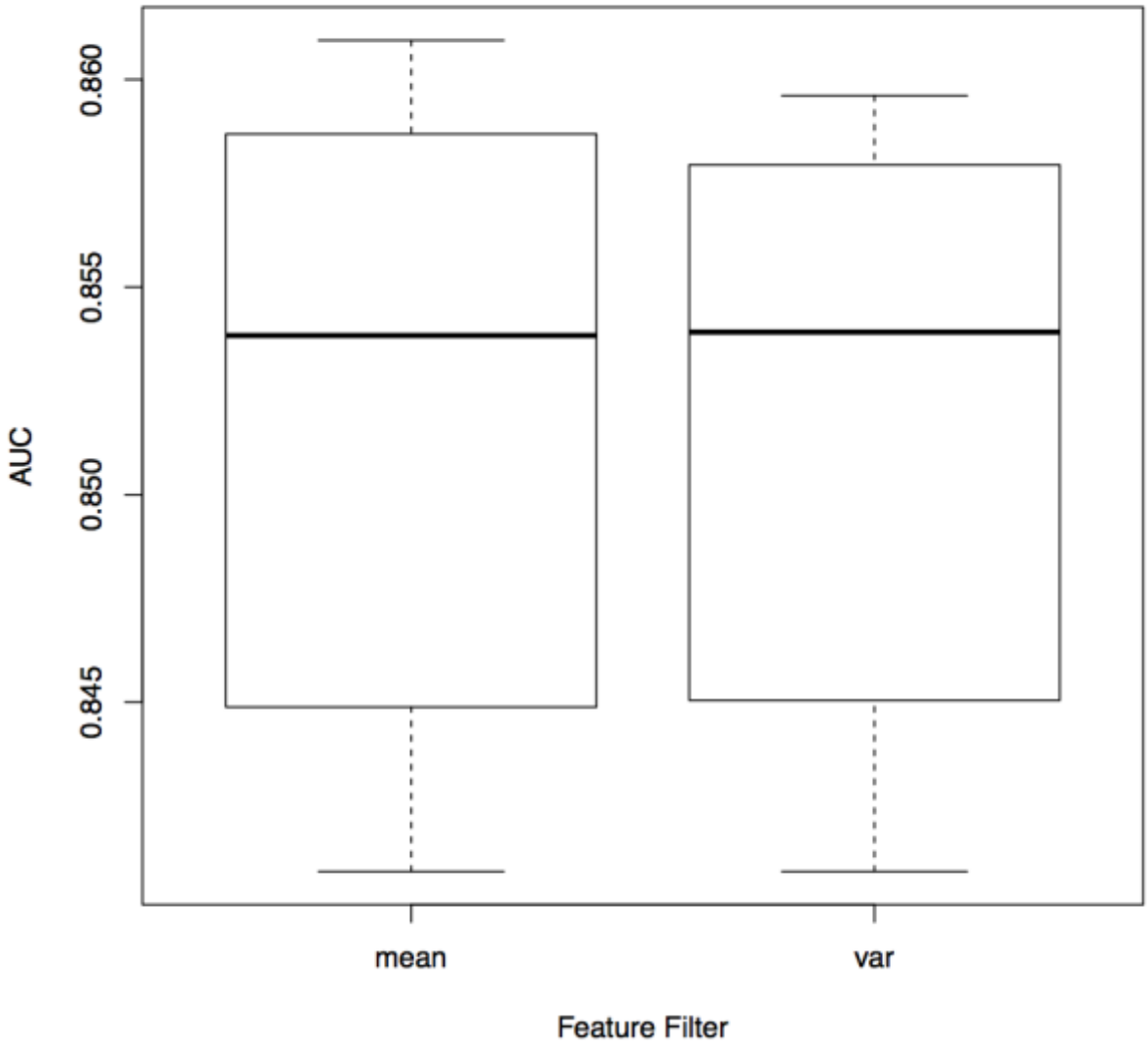


Figure 4:

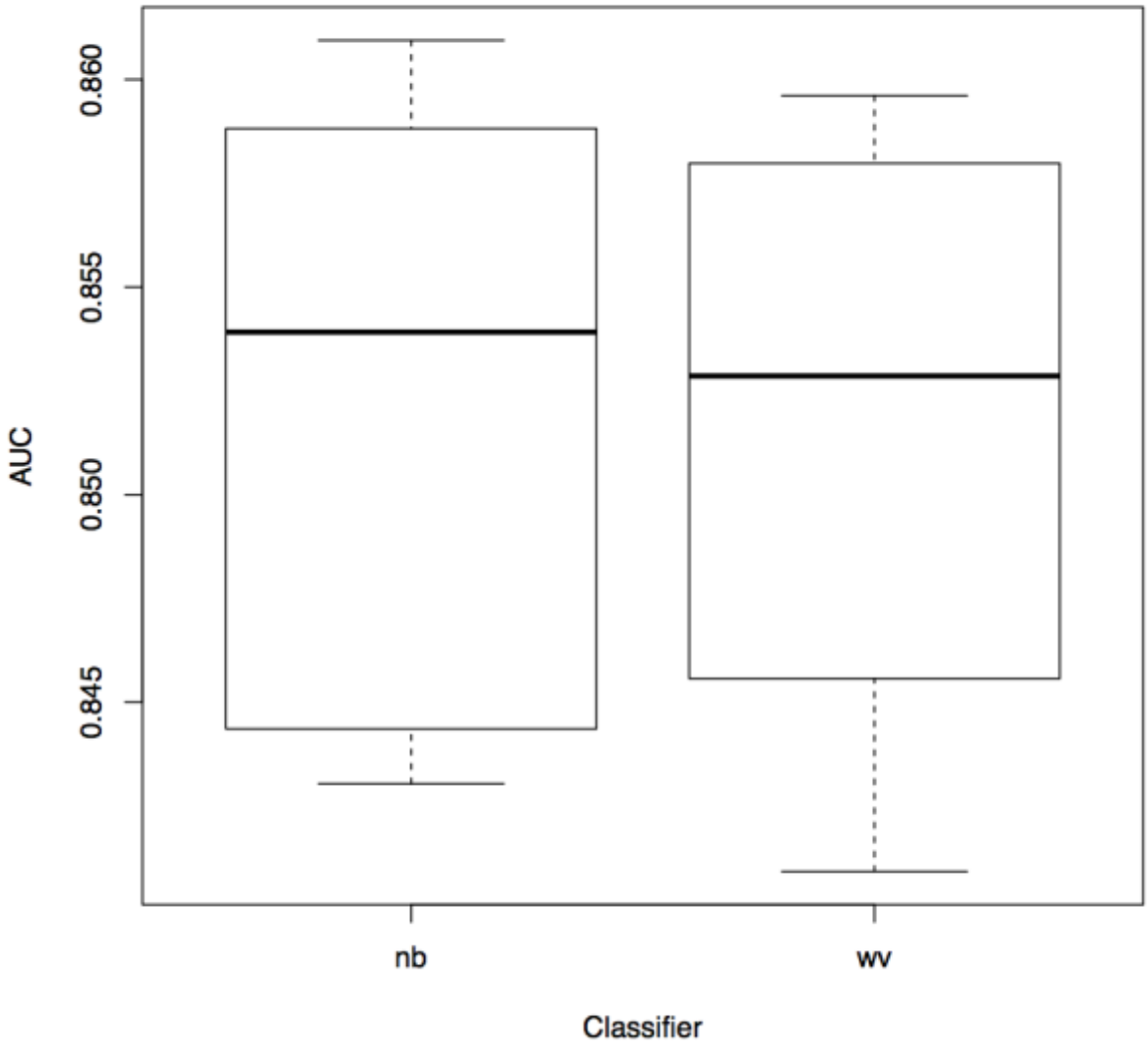
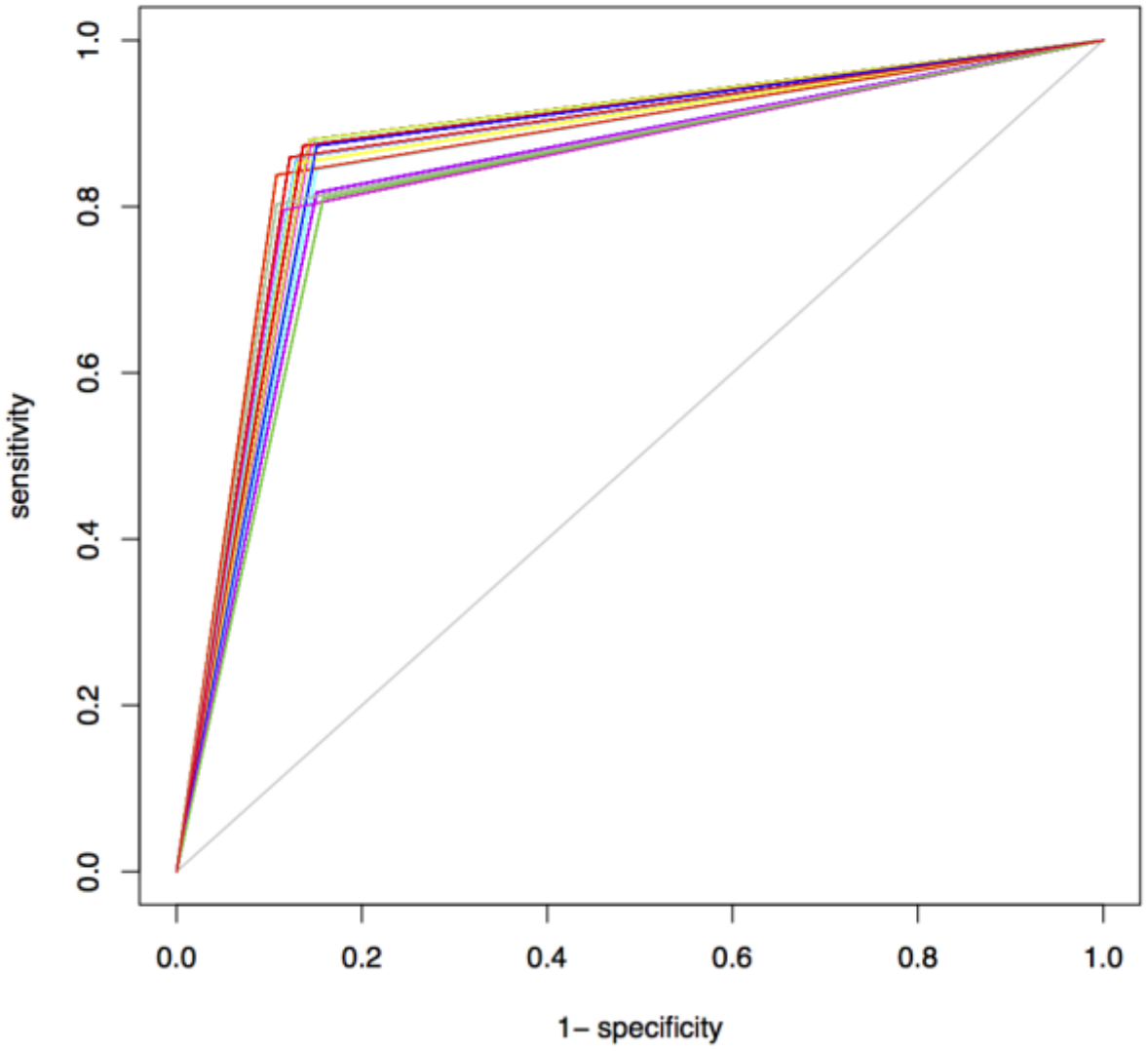


Figure 5:



Program used to check accuracy (final copy):

```
#Hannah Lerner
#7/13/16
#hlerner@bu.edu
#Pulmonomics Lab, Boston University


pipout<-function(testf , moda, folder , annotationCriteria , numberOfCvLoops ,
numberOfModels){

  #necessary libraries (kept for debugging):
```

```
#require(plyr)


#converting the function variables into variables used in the function
#if these are not named, the program will crash
modas<-moda
testfs<-testf
folders<-folder
cvs<-numberOfCvLoops
mods<-numberOfModels
ancrit <- annotationCriteria


#pulls annotations data and stores it to variable

#very specific
#annotations<-read.delim("~/Dropbox/Summer_2016/annotation.txt")

#more general (better)
annotations<-read.delim(testfs)
model_annotations<-read.delim(modas)

#create an empty data frame to store the information in
tests<-
data.frame(Sample_ID=factor(),Score=factor(),prediction=factor(),cv_loop=integer(),model=inte
ger())


#pipeline output
#folder<-tk_choose.dir(default = "", caption = "Select your Pipeline output folder")
#the outer loop is for each cv_loop (10 total)
for(cv in 1:cvs){
  cvloop<-paste0("/cv_loop_",cv)

  #the inner loop is for each model (24 in total)
  for(mymodel in 1:mods){

    #creating the pathway to look to read from
    modnum<-paste0("/model_",mymodel)
    modplace<-paste0(folders,cvloop)
    almostmod<-paste0(modplace,modnum)
    modfin<-paste0(almostmod,"/predictions.txt")

    #adds a column to differentiate the data in different cv_loops
    test<-cbind(read.table(modfin,header=TRUE,sep="\t"),cv)
    test<-cbind(test,model=mymodel)
```

```
    #puts the data into the table 1 model at a time
    tests<- rbind(tests,test)
  }
 }

 # make sure annotations is completely unique
 # tests is expected to not be unique, but is tested as well
 # View(tests)
 # table(tests$Sample_ID %in% annotations$ARRAY)
 # table(annotations$ARRAY %in% tests$Sample_ID)
 # table(duplicated(annotations$ARRAY))
 # table(duplicated(tests$Sample_ID))

 #tests <- na.omit(tests)

 #a new merged table with all the data merged
 ant<- merge(tests,annotations,by.x="Sample_ID",by.y="ARRAY", all.x = TRUE, all.y =
FALSE)



 #This needed to be changed when calculating the ROC curve/AUC for uniformity

 #original try (too specific)
 #ant$SmokingStatus<-revalue(as.factor(ant$SmokingStatus), c("2"="0"))

 #more general (better)
 #for this function to work, the annotation criteria needs to be 2 values with the
 #positive value equalling 1
 ant[[ancrit]] <- as.character(ant[[ancrit]])
 ant[[ancrit]][ant[[ancrit]] != "1" & ant[[ancrit]] != "NA"] <- "0"
 ant[[ancrit]] <- as.factor(ant[[ancrit]])



 #Test because ant$SmokingStatus was originally saved as an int
 class(ant[[ancrit]])


 afinaltable<-c()
 print("calculating...")
 #loop to count each cv_loop
 for(loop in 1:cvs){

   #loop to count each model
```

```
  for(mod in 1:mods){

    #calculate a true positive
    a<-sum(ant$Prediction==1&ant[[ancrit]]==1&ant$cv==loop&ant$model==mod)
    as<-paste0("true positive: ",a)



    #calculate a false negative
    b<-sum(ant$Prediction!=1&ant[[ancrit]]==1&ant$cv==loop&ant$model==mod)
    bs<-paste0("false negative: ",b)


    #calculate a false positive
    c<-sum(ant$Prediction==1&ant[[ancrit]]!=1&ant$cv==loop&ant$model==mod)
    cs<-paste0("false positive: ",c)
    #print(cs)

    #calculate a true negative
    d<-sum(ant$Prediction!=1&ant[[ancrit]]!=1&ant$cv==loop&ant$model==mod)
    ds<-paste0("true negative: ",d)


    #sensitivity
    sens<-a/(a+b)
    senstest<-paste0("sensitivity: ",sens)


    #specificity
    spec<-d/(c+d)
    spectest<-paste0("specificity: ", spec)


    #AUC
    funsub <- subset(ant,cv==loop&model==mod)
    myauc <- AUC::auc(AUC::roc(funsub$Prediction,as.factor(funsub[[ancrit]])))


    #Counts the number of NA's per model
    naCount<-sum(ant$Score=="NA"&ant[[ancrit]]=="NA"&ant$cv==loop&ant$model==mod)

    #saves the data from this iteration of the loop to a separate table
    afinaltable<-rbind(afinaltable,c(naPerModel = naCount , amodel=mod ,
asens=sens,aspec=spec,acv_loop=loop , AUC=myauc))

  }
```

```
  }
  #possibly unnecessary line
  afinaltable <- data.frame(afinaltable)

  #create new vector to house avgtable
  avgtable<-c()

  #averages the values from afinaltable to create a new table, avgtable
  for(aloop in 1:mods){

    #averages the values model by model (NOT cv_loop by cv_loop)
    tempsub <- subset(afinaltable, amodel==aloop)
    tempmeansens <- mean(tempsub$asens)
    tempmeanspec <- mean(tempsub$aspec)
    tempmeanAUC <- mean(tempsub$AUC)
    #tempmeannaPerModel <- mean(tempsub$naPerModel)


    #This should count the number of NA's in the score for each model
    tempmeannaPerModel <- sum(tempsub$Score==NaN)
      #tempsub$acv_loop==cvss&sum(tempmeannaPerModel)==sum(tempsub)



    # #makes sure the table is aware of a case of an
    # #insufficient amount of data for a given model
    # if(sum(tempmeannaPerModel)==sum(tempsub)){
    #   tempmeanAUC <- "Inconclusive"
    # }

    #clears the temp subset for reuse
    tempsub<-c()
    avgtable<-
rbind(avgtable,c(model=aloop,numNA=tempmeannaPerModel,avgsens=tempmeansens,avgspec
=tempmeanspec,AUC=tempmeanAUC))
  }
  #merges the model data to the analysis
  avgtable<-merge.data.frame(model_annotations,avgtable, by.x="model",by.y="model")

  #array to assign colors
  n<-
c(31,87,100,153,258,374,147,116,371,400,503,573,650,429,477,551,434,12,62,94,584,211,442)

  #Notifies user when the calculation phase ends and the graphing phase begins
  print("plotting...")
```

```
  #Creating a pdf to place the graphs in
  mypath = paste0(folders,"_Analysis.pdf")
  pdf(file = mypath)

  #graphs for each labeled classification
  boxplot(avgtable$AUC~avgtable$FeatureFilter, xlab = "Feature Filter" , ylab = "AUC")
  boxplot(avgtable$AUC~avgtable$FeatureSelection, xlab = "Feature Selection" , ylab = "AUC")
  boxplot(avgtable$AUC~avgtable$BiomarkerSize, xlab = "Biomarker Size" , ylab = "AUC")
  boxplot(avgtable$AUC~avgtable$Classifier, xlab = "Classifier" , ylab = "AUC")

  #ROC Curve
  for(cvl in 1:cvs){

    for(mod in 1:mods){

      notfunsubs <- subset(ant,cv==loop&model==mod)
      if(cvl==1&mod==1){
        plot(AUC::roc(notfunsubs$Prediction,as.factor(notfunsubs[[ancrit]])))
      }
      else{

plot(AUC::roc(notfunsubs$Prediction,as.factor(notfunsubs[[ancrit]])),add=TRUE,col=n[mod])
      }
    }
  }

  #stops the addition of data to PDF
  dev.off()

  #returns the final table produced by this function
  return(avgtable)

}
```

Second iteration of the accuracy checker program:

```
#Hannah Lerner
#7/13/16
#hlerner@bu.edu
#Pulmonomics Lab, Boston University

#pipouts<-function(testf , folder){
```

```
#require(plyr)
library(plyr)
#require(shiny)
#require(tcltk)
#testfs<-testf
#folders<-folder
folders<-"~/Dropbox/Summer_2016/Pipeline_Output"
#pulls annotations data and stores it to variable
annotations<-read.delim("~/Dropbox/Summer_2016/annotation.txt")
model_annotations<-read.delim("~/Dropbox/Summer_2016/model_annot.txt")
#testf<-tk_choose.files(default = "", caption = "Select your annotations file")
#annotations<-read.delim(testfs)

#create an empty data frame to store the information in
tests<-
data.frame(Sample_ID=factor(),Score=factor(),prediction=factor(),cv_loop=integer(),model=inte
ger())


#pipline output
#folder<-tk_choose.dir(default = "", caption = "Select your Pipeline output folder")
#the outer loop is for each cv_loop (10 total)
for(cv in 1:10){
  cvloop<-paste0("/cv_loop_",cv)

  #the inner loop is for each model (24 in total)
  for(mymodel in 1:24){

    #creating the pathway to look to read from
    modnum<-paste0("/model_",mymodel)
    modplace<-paste0(folders,cvloop)
    almostmod<-paste0(modplace,modnum)
    modfin<-paste0(almostmod,"/predictions.txt")

    #adds a column to differentiate the data in different cv_loops
    test<-cbind(read.table(modfin,header=TRUE,sep="\t"),cv)
    test<-cbind(test,model=mymodel)

    #puts the data into the table 1 model at a time
    tests<- rbind(tests,test)
  }
}

# make sure annotations is completely unique
# tests is expected to not be unique, but is tested as well
# View(tests)
```

```
# table(tests$Sample_ID %in% annotations$ARRAY)
# table(annotations$ARRAY %in% tests$Sample_ID)
# table(duplicated(annotations$ARRAY))
# table(duplicated(tests$Sample_ID))


#a new merged table with all the data merged
ant<- merge(tests,annotations,by.x="Sample_ID",by.y="ARRAY", all.x = TRUE, all.y =
FALSE)

#This needed to be changed when calculating the ROC curve/AUC for uniformity
ant$SmokingStatus<-revalue(as.factor(ant$SmokingStatus), c("2"="0"))

#Test because ant$SmokingStatus was originally saved as an int
class(ant$SmokingStatus)


afinaltable<-c()
print("calculating...")
#loop to count each cv_loop
for(loop in 1:10){

  #loop to count each model
  for(mod in 1:24){

    #calculate a true positive
    a<-sum(ant$Prediction==1&ant$SmokingStatus==1&ant$cv==loop&ant$model==mod)
    as<-paste0("true positive: ",a)



    #calculate a false negative
    b<-sum(ant$Prediction==0&ant$SmokingStatus==1&ant$cv==loop&ant$model==mod)
    bs<-paste0("false negative: ",b)


    #calculate a false positive
    c<-sum(ant$Prediction==1&ant$SmokingStatus==0&ant$cv==loop&ant$model==mod)
    cs<-paste0("false positive: ",c)
    #print(cs)

    #calculate a true negative
    d<-sum(ant$Prediction==0&ant$SmokingStatus==0&ant$cv==loop&ant$model==mod)
    ds<-paste0("true negative: ",d)
```

```
    #sensitivity
    sens<-a/(a+b)
    senstest<-paste0("sensitivity: ",sens)


    #specificity
    spec<-d/(c+d)
    spectest<-paste0("specificity: ", spec)


    #AUC
    funsub <- subset(ant,cv==loop&model==mod)
    myauc <- AUC::auc(AUC::roc(funsub$Prediction,as.factor(funsub$SmokingStatus)))


    #Counts the number of NA's per model
    naCount<-
sum(ant$Score=="NA"&ant$SmokingStatus=="NA"&ant$cv==loop&ant$model==mod)

    #saves the data from this iteration of the loop to a separate table
    afinaltable<-rbind(afinaltable,c(amodel=mod , asens=sens , aspec=spec , acv_loop=loop ,
AUC=myauc , naPerModel = naCount))

    }
}
#possibly unnecessary line
afinaltable <- data.frame(afinaltable)

#create new vector to house avgtable
avgtable<-c()

#averages the values from afinaltable to create a new table, avgtable
for(aloop in 1:24){

  #averages the values model by model (NOT cv_loop by cv_loop)
  tempsub <- subset(afinaltable, amodel==aloop)
  tempmeansens <- mean(tempsub$asens)
  tempmeanspec <- mean(tempsub$aspec)
  tempmeanAUC <- mean(tempsub$AUC)
  tempmeannaPerModel <- sum(tempsub$Score==NaN)

  #clears the temp subset for reuse
  tempsub<-c()
  avgtable<-
rbind(avgtable,c(model=aloop,numNA=tempmeannaPerModel,avgsens=tempmeansens,avgspec
=tempmeanspec,AUC=tempmeanAUC))
```

```
}
avgtable<-merge.data.frame(model_annotations,avgtable, by.x="model",by.y="model")
boxplot(avgtable$AUC~avgtable$FeatureFilter)
boxplot(avgtable$AUC~avgtable$FeatureSelection)
boxplot(avgtable$AUC~avgtable$BiomarkerSize)
boxplot(avgtable$AUC~avgtable$Classifier)
#array to assign colors
n<-
c(31,87,100,153,258,374,147,116,371,400,503,573,650,429,477,551,434,12,62,94,584,211,442)

#plots the ROC curve
print("plotting...")
for(cvl in 1:10){

  for(mod in 1:24){

    notfunsubs <- subset(ant,cv==loop&model==mod)
    if(cvl==1&mod==1){
    plot(AUC::roc(notfunsubs$Prediction,as.factor(notfunsubs$SmokingStatus)))
    }
    else{

plot(AUC::roc(notfunsubs$Prediction,as.factor(notfunsubs$SmokingStatus)),add=TRUE,col=n[
mod])
    }
  }
}
#return(avgtable)
#used to distribute the avgtable
#write.table(avgtable,file="avgtable.txt",sep="\t",col.names=TRUE,row.names=FALSE,quote=F
ALSE)

#}
```

Table 2: Final table used for analysis:

	model	sensitivity	specificity	cv_loop	AUC
1	1	0.8000000	0.9178082	1	0.8589041
2	2	0.8370370	0.8835616	1	0.8602993
3	3	0.8148148	0.9109589	1	0.8628869
4	4	0.8444444	0.8835616	1	0.8640030
5	5	0.8296296	0.9041096	1	0.8668696

6	6	0.8444444	0.8904110	1	0.8674277
7	7	0.7703704	0.9246575	1	0.8475140
8	8	0.7777778	0.9041096	1	0.8409437
9	9	0.7703704	0.9178082	1	0.8440893
10	10	0.8074074	0.8972603	1	0.8523338
11	11	0.8000000	0.9178082	1	0.8589041
12	12	0.8296296	0.8972603	1	0.8634450
13	13	0.8000000	0.9178082	1	0.8589041
14	14	0.8370370	0.8835616	1	0.8602993
15	15	0.8148148	0.9109589	1	0.8628869
16	16	0.8370370	0.8835616	1	0.8602993
17	17	0.8296296	0.9041096	1	0.8668696
18	18	0.8444444	0.8904110	1	0.8674277
19	19	0.7703704	0.9246575	1	0.8475140
20	20	0.7777778	0.9041096	1	0.8409437
21	21	0.7703704	0.9178082	1	0.8440893
22	22	0.8074074	0.8972603	1	0.8523338
23	23	0.8000000	0.9178082	1	0.8589041
24	24	0.8296296	0.8972603	1	0.8634450
25	1	0.8461538	0.8902439	2	0.8681989
26	2	0.9059829	0.8353659	2	0.8706744
27	3	0.8632479	0.8780488	2	0.8706483
28	4	0.9059829	0.8231707	2	0.8645768
29	5	0.8803419	0.8414634	2	0.8609026
30	6	0.9059829	0.8170732	2	0.8615280
31	7	0.8205128	0.9024390	2	0.8614759
32	8	0.8632479	0.8597561	2	0.8615020
33	9	0.8632479	0.8841463	2	0.8736971

34	10	0.8717949	0.8475610	2	0.8596779
35	11	0.8547009	0.8780488	2	0.8663748
36	12	0.9059829	0.8414634	2	0.8737232
37	13	0.8461538	0.8902439	2	0.8681989
38	14	0.9059829	0.8353659	2	0.8706744
39	15	0.8632479	0.8780488	2	0.8706483
40	16	0.9059829	0.8231707	2	0.8645768
41	17	0.8803419	0.8414634	2	0.8609026
42	18	0.9059829	0.8170732	2	0.8615280
43	19	0.8205128	0.9024390	2	0.8614759
44	20	0.8632479	0.8597561	2	0.8615020
45	21	0.8632479	0.8841463	2	0.8736971
46	22	0.8717949	0.8475610	2	0.8596779
47	23	0.8547009	0.8780488	2	0.8663748
48	24	0.9059829	0.8414634	2	0.8737232
49	1	0.8029197	0.9166667	3	0.8597932
50	2	0.8613139	0.8750000	3	0.8681569
51	3	0.8102190	0.8958333	3	0.8530262
52	4	0.8613139	0.8888889	3	0.8751014
53	5	0.8321168	0.8888889	3	0.8605028
54	6	0.8467153	0.8888889	3	0.8678021
55	7	0.7737226	0.9236111	3	0.8486669
56	8	0.8029197	0.8888889	3	0.8459043
57	9	0.7737226	0.9166667	3	0.8451946
58	10	0.8029197	0.9027778	3	0.8528487
59	11	0.8102190	0.9027778	3	0.8564984
60	12	0.8467153	0.8958333	3	0.8712743
61	13	0.8029197	0.9166667	3	0.8597932

62	14	0.8613139	0.8750000	3	0.8681569
63	15	0.8102190	0.8958333	3	0.8530262
64	16	0.8613139	0.8888889	3	0.8751014
65	17	0.8321168	0.8888889	3	0.8605028
66	18	0.8467153	0.8888889	3	0.8678021
67	19	0.7737226	0.9236111	3	0.8486669
68	20	0.8029197	0.8888889	3	0.8459043
69	21	0.7737226	0.9166667	3	0.8451946
70	22	0.8029197	0.9027778	3	0.8528487
71	23	0.8102190	0.9027778	3	0.8564984
72	24	0.8467153	0.8958333	3	0.8712743
73	1	0.7099237	0.9400000	4	0.8249618
74	2	0.8015267	0.8800000	4	0.8407634
75	3	0.7786260	0.9400000	4	0.8593130
76	4	0.8015267	0.8733333	4	0.8374300
77	5	0.7786260	0.8933333	4	0.8359796
78	6	0.8015267	0.8733333	4	0.8374300
79	7	0.7022901	0.9400000	4	0.8211450
80	8	0.7404580	0.9200000	4	0.8302290
81	9	0.7328244	0.9400000	4	0.8364122
82	10	0.7404580	0.9000000	4	0.8202290
83	11	0.7709924	0.9266667	4	0.8488295
84	12	0.8015267	0.8733333	4	0.8374300
85	13	0.7175573	0.9400000	4	0.8287786
86	14	0.8167939	0.8800000	4	0.8483969
87	15	0.7709924	0.9400000	4	0.8554962
88	16	0.8091603	0.8800000	4	0.8445802
89	17	0.8015267	0.8933333	4	0.8474300

90	18	0.8091603	0.8666667	4	0.8379135
91	19	0.7022901	0.9400000	4	0.8211450
92	20	0.7404580	0.9200000	4	0.8302290
93	21	0.7328244	0.9400000	4	0.8364122
94	22	0.7557252	0.8933333	4	0.8245293
95	23	0.7786260	0.9200000	4	0.8493130
96	24	0.8091603	0.8666667	4	0.8379135
97	1	0.7967480	0.9050633	5	0.8509056
98	2	0.8455285	0.8670886	5	0.8563085
99	3	0.8048780	0.9050633	5	0.8549707
100	4	0.8455285	0.8734177	5	0.8594731
101	5	0.8373984	0.8860759	5	0.8617372
102	6	0.8292683	0.8734177	5	0.8513430
103	7	0.7642276	0.9177215	5	0.8409746
104	8	0.8048780	0.8987342	5	0.8518061
105	9	0.7967480	0.9113924	5	0.8540702
106	10	0.8048780	0.8860759	5	0.8454770
107	11	0.8130081	0.9050633	5	0.8590357
108	12	0.8292683	0.8797468	5	0.8545076
109	13	0.7967480	0.9050633	5	0.8509056
110	14	0.8455285	0.8670886	5	0.8563085
111	15	0.8048780	0.9050633	5	0.8549707
112	16	0.8455285	0.8734177	5	0.8594731
113	17	0.8373984	0.8860759	5	0.8617372
114	18	0.8292683	0.8734177	5	0.8513430
115	19	0.7642276	0.9177215	5	0.8409746
116	20	0.8048780	0.8987342	5	0.8518061
117	21	0.7967480	0.9113924	5	0.8540702

118	22	0.8048780	0.8860759	5	0.8454770
119	23	0.8130081	0.9050633	5	0.8590357
120	24	0.8292683	0.8797468	5	0.8545076
121	1	0.8188976	0.9025974	6	0.8607475
122	2	0.8503937	0.8701299	6	0.8602618
123	3	0.8346457	0.9025974	6	0.8686215
124	4	0.8503937	0.8831169	6	0.8667553
125	5	0.8110236	0.8896104	6	0.8503170
126	6	0.8110236	0.8636364	6	0.8373300
127	7	0.7637795	0.9285714	6	0.8461755
128	8	0.8188976	0.8896104	6	0.8542540
129	9	0.7874016	0.9090909	6	0.8482462
130	10	0.8110236	0.8766234	6	0.8438235
131	11	0.8346457	0.8961039	6	0.8653748
132	12	0.8267717	0.8831169	6	0.8549443
133	13	0.8188976	0.9025974	6	0.8607475
134	14	0.8503937	0.8701299	6	0.8602618
135	15	0.8346457	0.9025974	6	0.8686215
136	16	0.8503937	0.8831169	6	0.8667553
137	17	0.8110236	0.8896104	6	0.8503170
138	18	0.8110236	0.8636364	6	0.8373300
139	19	0.7637795	0.9285714	6	0.8461755
140	20	0.8188976	0.8896104	6	0.8542540
141	21	0.7874016	0.9090909	6	0.8482462
142	22	0.8110236	0.8766234	6	0.8438235
143	23	0.8346457	0.8961039	6	0.8653748
144	24	0.8267717	0.8831169	6	0.8549443
145	1	0.7042254	0.9352518	7	0.8197386

146	2	0.8309859	0.8920863	7	0.8615361
147	3	0.7816901	0.9136691	7	0.8476796
148	4	0.8380282	0.8705036	7	0.8542659
149	5	0.8309859	0.8920863	7	0.8615361
150	6	0.8380282	0.8633094	7	0.8506688
151	7	0.7112676	0.9352518	7	0.8232597
152	8	0.7957746	0.8992806	7	0.8475276
153	9	0.7394366	0.9352518	7	0.8373442
154	10	0.8028169	0.8776978	7	0.8402574
155	11	0.7746479	0.9136691	7	0.8441585
156	12	0.8380282	0.8633094	7	0.8506688
157	13	0.7042254	0.9352518	7	0.8197386
158	14	0.8309859	0.8920863	7	0.8615361
159	15	0.7816901	0.9136691	7	0.8476796
160	16	0.8380282	0.8705036	7	0.8542659
161	17	0.8309859	0.8920863	7	0.8615361
162	18	0.8380282	0.8633094	7	0.8506688
163	19	0.7112676	0.9352518	7	0.8232597
164	20	0.7957746	0.8992806	7	0.8475276
165	21	0.7394366	0.9352518	7	0.8373442
166	22	0.8028169	0.8776978	7	0.8402574
167	23	0.7746479	0.9136691	7	0.8441585
168	24	0.8380282	0.8633094	7	0.8506688
169	1	0.7761194	0.8775510	8	0.8268352
170	2	0.8134328	0.8435374	8	0.8284851
171	3	0.8208955	0.8571429	8	0.8390192
172	4	0.8208955	0.8435374	8	0.8322165
173	5	0.7910448	0.8571429	8	0.8240938

174	6	0.8059701	0.8435374	8	0.8247538
175	7	0.7611940	0.8979592	8	0.8295766
176	8	0.7985075	0.8435374	8	0.8210224
177	9	0.7686567	0.8843537	8	0.8265052
178	10	0.8134328	0.8435374	8	0.8284851
179	11	0.7910448	0.8775510	8	0.8342979
180	12	0.7985075	0.8503401	8	0.8244238
181	13	0.7761194	0.8775510	8	0.8268352
182	14	0.8134328	0.8435374	8	0.8284851
183	15	0.8208955	0.8571429	8	0.8390192
184	16	0.8358209	0.8435374	8	0.8396792
185	17	0.7910448	0.8571429	8	0.8240938
186	18	0.8059701	0.8435374	8	0.8247538
187	19	0.7611940	0.8979592	8	0.8295766
188	20	0.7985075	0.8435374	8	0.8210224
189	21	0.7686567	0.8843537	8	0.8265052
190	22	0.8134328	0.8435374	8	0.8284851
191	23	0.7910448	0.8775510	8	0.8342979
192	24	0.7985075	0.8503401	8	0.8244238
193	1	0.7874016	0.9155844	9	0.8514930
194	2	0.8582677	0.8896104	9	0.8739391
195	3	0.8346457	0.9155844	9	0.8751150
196	4	0.8425197	0.8961039	9	0.8693118
197	5	0.8503937	0.9025974	9	0.8764956
198	6	0.8425197	0.9025974	9	0.8725585
199	7	0.7716535	0.9285714	9	0.8501125
200	8	0.8188976	0.8961039	9	0.8575008
201	9	0.7795276	0.9090909	9	0.8443092

202	10	0.8267717	0.8961039	9	0.8614378
203	11	0.8425197	0.9155844	9	0.8790521
204	12	0.8346457	0.8961039	9	0.8653748
205	13	0.7874016	0.9155844	9	0.8514930
206	14	0.8582677	0.8896104	9	0.8739391
207	15	0.8346457	0.9155844	9	0.8751150
208	16	0.8503937	0.9025974	9	0.8764956
209	17	0.8503937	0.9025974	9	0.8764956
210	18	0.8346457	0.8961039	9	0.8653748
211	19	0.7716535	0.9285714	9	0.8501125
212	20	0.8188976	0.8961039	9	0.8575008
213	21	0.7795276	0.9025974	9	0.8410625
214	22	0.8188976	0.8961039	9	0.8575008
215	23	0.8425197	0.9155844	9	0.8790521
216	24	0.8503937	0.8896104	9	0.8700020
217	1	0.8380282	0.8920863	10	0.8650572
218	2	0.8802817	0.8489209	10	0.8646013
219	3	0.8591549	0.8705036	10	0.8648293
220	4	0.8802817	0.8561151	10	0.8681984
221	5	0.8732394	0.8633094	10	0.8682744
222	6	0.8732394	0.8561151	10	0.8646773
223	7	0.7957746	0.8848921	10	0.8403334
224	8	0.8169014	0.8489209	10	0.8329111
225	9	0.8028169	0.8920863	10	0.8474516
226	10	0.8098592	0.8417266	10	0.8257929
227	11	0.8521127	0.8705036	10	0.8613081
228	12	0.8450704	0.8489209	10	0.8469956
229	13	0.8380282	0.8920863	10	0.8650572

230	14	0.8802817	0.8489209	10	0.8646013
231	15	0.8591549	0.8705036	10	0.8648293
232	16	0.8802817	0.8561151	10	0.8681984
233	17	0.8732394	0.8633094	10	0.8682744
234	18	0.8732394	0.8489209	10	0.8610801
235	19	0.7957746	0.8848921	10	0.8403334
236	20	0.8169014	0.8489209	10	0.8329111
237	21	0.8028169	0.8920863	10	0.8474516
238	22	0.8098592	0.8417266	10	0.8257929
239	23	0.8591549	0.8776978	10	0.8684264
240	24	0.8521127	0.8489209	10	0.8505168

Bibliography:

Beal, Vangie. "OOP – Object Oriented Programming." *What Is Object-Oriented Programming? Webopedia Definition*. N.p., n.d. Web. 30 Jan. 2017.

Beane, Jennifer. "A Prediction Model for Lung Cancer Diagnosis That Integrates Genomic and Clinical Features." *Cancer Prevention Research* 1.1 (2008): n. pag. 31 Mar. 2008. Web. 20 Aug. 2016.

"Biomarker (medicine)." *Wikipedia*. Wikimedia Foundation, n.d. Web. 30 Jan. 2017.

BERRIDGE, VIRGINIA. "THE POLICY RESPONSE TO THE SMOKING AND LUNG CANCER CONNECTION IN THE 1950s AND 1960s." *Historical Journal (Cambridge, England)* 49.4 (2006): 1185–1209. *PMC*. Web. 30 Jan. 2017.

By Adding : Colour in Ggplot Function , I Could Achieve the Lines with Different Colors Related to the Group Present in the Plot. "Plot Multiple Lines (data Series) Each with Unique Color in R." - *Stack Overflow*. N.p., 2013. Web. 19 July 2016.

By. "How to Calculate Area Under the Curve (AUC), or the C-statistic, by Hand." *Regression*. N.p., 2015. Web. 10 Oct. 2016.

By Posting Your Answer, You Agree to the Privacy Policy and Terms of Service. "Conditional Sum in R." - *Stack Overflow*. N.p., 2012. Web. 10 Oct. 2016.

By Posting Your Answer, You Agree to the Privacy Policy and Terms of Service. "Plot ROC Curve and Calculate AUC in R at Specific Cutoff Info." - *Stack Overflow*. N.p., 2014. Web. 29 Aug. 2016.

"Exams and Tests That Look for Lung Cancer." *American Cancer Society*. N.p., n.d. Web. 22 Jan. 2017.

Lemmens, Paul. "[R] Reading Multiple Txt Files into One Data Frame." *[R] Reading Multiple Txt Files into One Data Frame*. N.p., 31 July 2006. Web. 13 Aug. 2016.

"Lung Cancer: History and Hope." *Top Masters in Healthcare Administration*. N.p., n.d. Web. 29 Jan. 2017.

N.p., n.d. Web. 29 Jan. 2017.

Silvestri, Gerald A., and Anil Vachani. "A Bronchial Genomic Classifier for the Diagnostic Evaluation of Lung Cancer — NEJM." *New England Journal of Medicine*. N.p., 17 May 2015. Web. 10 Oct. 2016.

Spira, Avrum, Jennifer E. Beane, and Vishal Shah. "Airway Epithelial Gene Expression in the Diagnostic Evaluation of Smokers with Suspect Lung Cancer." *Nature Medicine*. N.p., 4 Mar. 2007. Web. 20 Aug. 2016.

Whitney, Duncan H., Michael R. Elashoff, and Kate Porta-Smith. "Derivation of a Bronchial Genomic Classifier for Lung Cancer in a Prospective Study of Patients Undergoing Diagnostic Bronchoscopy." *BioMed Central*. N.p., 6 May 2015. Web. 28 July 2016.

Wily, Joshua. "[R] Import Multiple Csv Files and Merge into One Master File." *[R] Import Multiple Csv Files and Merge into One Master File*. N.p., 8 Oct. 2010. Web. 16 Oct. 2016.