



Classification of cancer status based on genetic markers using various unsupervised learning, classification and regression methods

Duyen Ho (duyenho2), Michael Garbus (mgarbus2), Team Lead: Christina Hu (ch35)

12/6/2021

Contents

Project Description and Summary	2
Literature Review	2
Summary Statistics and Data Processing	3
Data Cleaning	4
Use correlation matrix	5
Modeling PR.Status	5
Modeling with KNN (predict PR.Status)	5
Modeling with Lasso (predict PR.Status)	6
Modelling histological.type	8
Modelling histological.type with SVM	8
Modelling histological.type with Random Forests	9
50-Variable Selection	10

Project Description and Summary

The goal of this project is to find the most accurate model using a variety of statistical model building techniques in order to predict medical outcomes relating to breast cancer. We will use the BRCA Multi-Omics (TCGA) data from Kaggle, and predict outcomes for the `PR.Status`, `ER.Status`, `HER2.Final.Status`, and `histological.type`.

For our approach, we will first perform the necessary data cleaning operations, then we will use **KNN** and **Lasso** methods to build classification models for `PR.Status` and `SVM` and **Random Forests** for `histological.type`, using classification error and AUC respectively as the evaluation criteria. Next, we will have **Linear** and **Radial SVM** along with **XGBoost** to build a model with the goal of accurately predicting **all four** outcomes narrowing the predictors down to 50 variables.

Finally, we have concluded that Lasso was a better method to fit the model when predicting `PR.Status`. On the other hand, Random Forest gives us a higher AUC value, so this would best predict our `histological.type` variable. And for our last part of choosing 50 variables to predict all 4 outcomes, XGBoost helps us estimate the importance of features for a predictive modeling problem using the gradient boosting algorithm. With this method, we are confident that our 50 predictors influence our outcomes the most. This way, our model doesn't have noise from unimportant/irrelevant variables that we have filtered out.

Some challenges we faced were that the categorical variables gave us many errors when we were trying to fit some of the models, so we excluded them while modeling those potential fits first.

Literature Review

We consulted some research involving the subject to guide us in our modelling. The Wisconsin Breast Cancer dataset was used, which involves a digitized image of a fine needle aspirate (FNA) of a breast mass, which describe the characteristics of the cell nuclei present in the image. Unfortunately, this is rather different from our current data, which contains a sample of genetic data instead of a picture of cell data. However, we still feel that their approaches can be useful for our data, as they are predicting binary outcomes on a similar subject matter.

The first study considered was “Breast Cancer Prediction: A Comparative Study Using Machine Learning Techniques”, published in 2020 by Islam, M.M. et al. in SN Computer Science. The paper compares 5 different supervised machine learning techniques (support vector machine, K-nearest neighbors, random forests, artificial neural networks, and finally, logistic regression). They performed little preprocessing on the data, as they only removed 16 NA variables. They divided the data into test and train sets, and used a 10-fold cross validation, with nine-fold used for training and the remaining fold used for testing. The results revealed that ANNs return the highest accuracy (98.57%), KNN and SVM came second (97.1%), and random forests and logistic regression came third, clocking in with an accuracy of 95.71% for the two models. ANNs also achieved the highest specificity (96%), whereas SVM achieved the lowest, at around 92.3%. KNN and random forest had a specificity of 98.53%, whereas logistic regression had a specificity of 95.65%. ANN and SVM achieved the highest sensitivity, clocking in at exactly 100%. This made SVM and ANN very attractive to us when considering our model selection. KNN had a sensitivity of 97.82%, Random Forest had a sensitivity of 95.65%, and logistic regression had a sensitivity of 95.74%.

However, the highest values for AUC were KNN and a random forest model, both achieving an AUC of 99%. As a result, we decided to use these models for the project.

In the discussion of related works, the authors mentioned another study which received a 98.83% accuracy for a boosted trees random forest model, which inspired us to test gradient boosted trees for the final part of the project, as they will be able to determine which features are important, and hopefully cut down on features as well.

A similar study, “Using Machine Learning Algorithms for Breast Cancer Risk Prediction and Diagnosis”, published in 2016 by Asri, H. et al in *Procedia Computer Science* 83 tested SVM, decision tree (C 4.5, a technique available in the open source WEKA data analysis tool), Naive Bayes, as well as KNN on the same UCI Breast Cancer dataset, with a focus on accuracy. Asri et al. discovered that although SVM took the longest time to create a model, it resulted in the highest accuracy of 97.54%, which is a different result to the previous model we studied.

While we do not believe that time constraints will be important in regards to our model as we are trying to achieve a high level of classification, it is interesting that these values are different. We imagine that this may be due to a difference in tools used, or perhaps, in the kernel used for the SVM classification. Additionally, the study also found that SVM had the highest AUC value on the ROC curve (99%). For these reason, we decided to use SVM for our model.

In the paper “Supervised Risk Predictor of Breast Cancer Based on Intrinsic Subtypes” published in 2009 by many authors from multiple institutions, the authors focus on the approaches to improve on current standards for breast cancer prognosis and prediction of chemotherapy benefit by developing a risk model using gene expressions. Various methods were used such as Samples and Clinical Data, Microarray, Identification of Prototypical Intrinsic Subtype Samples and Genes, Sample Subtype Prediction. A standardized method of classification using a statistically derived gene and sample set validated across multiple cohorts was developed. They split training and testing data for patient cohorts. Their method is also to use microarray and quantitative reverse transcriptase polymerase chain reaction data to develop a 50-gene subtype predictor using data from prototype samples. In the end, they compared four different models for prediction of relapse and used C-index to compare the strength of the models. C-index is calculated for each test set to form the estimate of each model with two sample t test. They found that all the intrinsic subtypes are present and significant for outcome predictions in cohorts of patients diagnosed with ER tumors. Stratification of the subtypes within HER2clin-positive samples did not show significance in outcome predictions. And also, approximately 10% of breast cancers were classified as normal-like and can be ER tumors and have an intermediate prognosis.

In another paper “Comprehensive Molecular Portraits of Invasive Lobular Breast Cancer” published in 2015, invasive lobular carcinoma (ILC) was focused on as the second most common histologic subtype of invasive breast cancer along with some potential clinical options. Fisher’s test and binary of copy number alterations and mutations were used to analyze selected events. The authors also used the microarray approach. Limmavoom package helps them perform differential expression analysis on RNA-seq data. It is to estimate the mean-variance relationship in the data to compute an appropriate precision weight for each observation, this could possibly be used in our project. Expression levels and proteins levels data was used for protein differential expression analysis and was assessed by t test as well, like the previous paper. Another procedure that they used is Consensus Cluster Plus Analysis to determine ILC subtypes based on 1000 most differentially expressed genes and a classifier built by C4.5. They have found that lobular tumors were predominantly classified as LumA and are ER positive tumors. ER status was clinically determined by immunohistochemistry on 120 of 127 ILC cases, with 94% ($n = 113$) scoring positively. Their studies also showed that mixed histology tumors really tend to resemble ILC and IDC. Moreover, IDC and ILC discriminant molecular features, in particular CDH1 status, could be used to stratify mixed tumors into ILC-like and IDC-like tumor subgroups. The two papers use microarray, t-test, and stratified samplings. Each has some unique methods of classification that we could also try out for our project.

For the creation of our final model, we consulted some literature on predicting in medical data. We found “Predicting Missing Values in Medical Data via XGBoost Regression”, from Zhang et al., which found that XGBoost was able to improve prediction of missing values by 20% for over 8200 observations. Because of this model, as well as discussion in Islam et al., we decided to consider XGBoost.

Summary Statistics and Data Processing

- 705 observations (breast cancer samples)

- 1936 variables/predictors (4 different omics data types)
- [1:604] are rn - gene expressions 604 <- categorical data
- [605:1464] are cn - copy number variations 860
- [1465:1713] are mu - somatic mutations 249 <- categorical data
- [1714:1936] are pp - protein levels 223
- 4 outcomes [1937:1940]

```
## [1] 705 1941
```

Data Cleaning

First, we must clean the data. We will clean the data by only keeping observations that satisfy the criteria of having values equal to the levels/outcomes that we are looking for (Positive, Negative, infiltrating ductal carcinoma and infiltrating lobular carcinoma). If they are not satisfied or null, we will remove them.

```
# remove the 'vital.status'
brca_org <- brca_org[, -1937]
# organize the 4 outcome data
brca_org <- brca_org[((brca_org$histological.type == "infiltrating lobular carcinoma") |
  (brca_org$histological.type == "infiltrating ductal carcinoma")) &
  ((brca_org$HER2.Final.Status == "Positive") | (brca_org$HER2.Final.Status ==
    "Negative")) & ((brca_org$ER.Status == "Positive") |
  (brca_org$ER.Status == "Negative")) & ((brca_org$PR.Status ==
    "Positive") | (brca_org$PR.Status == "Negative")), ]
```

Dimension and sample of our cleaned dataset (only the outcomes):

```
## [1] 507 1940
```

```
##   PR.Status ER.Status HER2.Final.Status histological.type
## 1 Positive  Positive          Negative infiltrating ductal carcinoma
## 2 Positive  Negative          Negative infiltrating ductal carcinoma
## 3 Positive  Positive          Negative infiltrating ductal carcinoma
## 4 Positive  Positive          Negative infiltrating ductal carcinoma
## 5 Positive  Positive          Negative infiltrating ductal carcinoma
```

- There was no missing value in the entire dataset, so there should be no missing values in the continuous predictors as well

```
sum(colSums(is.na(brca_org)))
```

```
## [1] 0
```

- There were some outliers in the continuous predictors, we just did a quick test on one of the continuous variables. But we decided to keep them as they are since there are only a few, and we will filter out the dataset later on with the correlation matrix.

```
outfun <- function(x) {
  abs(x - mean(x, na.rm = TRUE)) > 3 * sd(x, na.rm = TRUE)
}
sum(outfun(brca_org[, 605]))
```

```
## [1] 5
```

```
sum(outfun(brca_org[, 1000]))
```

```
## [1] 1
```

- We want focus on filtering out the insignificant continuous variables first, so we used the correlation matrix method. We then dropped the highly correlated columns.

Use correlation matrix

We will next split our data into outcomes and predictors, then use the correlation method with a cutoff of 0.7 to filter our highly correlated columns:

```
# only use variances using the corr mat Only continuous
# predictors
non_categorical_brca = cbind(brca_data[1:604], brca_data[1714:1936])
# use correlation matrix
cor_mat = cor(non_categorical_brca)
# returns vector of indices to remove
cor_list = findCorrelation(cor_mat, cutoff = 0.7)
# drop highly correlated columns
non_categorical_brca = non_categorical_brca[, -c(cor_list)]
```

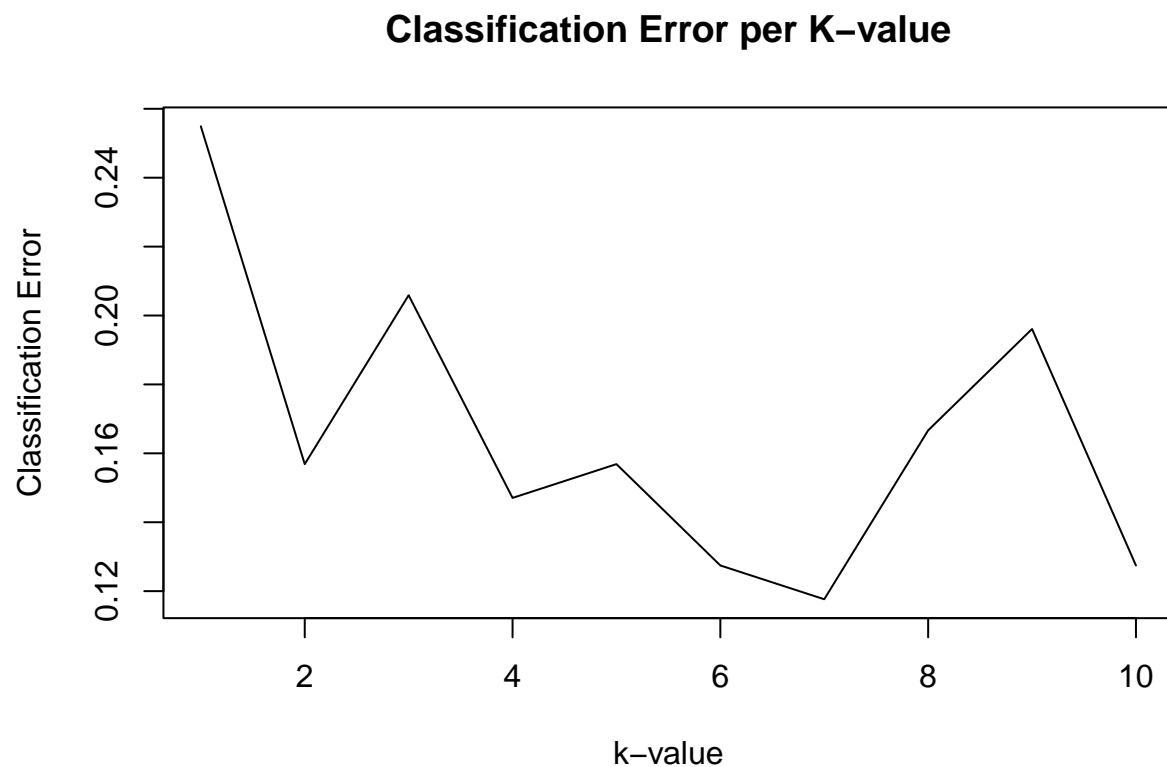
Modeling PR.Status

Modeling with KNN (predict PR.Status)

First, we will use KNN model to predict PR.Status. We will use a test train split of 80/20 on the data. We will only be using the non-categorical variables to build the KNN model since we ran into a lot of errors trying to incorporate the categorical variables

We will find the optimal k-value to use, ranging from k=1-10.

Plot the graph of the classification errors for each k-value and see if we can use the result to determine the best k:



As shown from the plot, the k-value with the lowest classification error is k=7, so we will use that for our model.

Next, use k=7 to create our KNN model and display the confusion matrix:

```
##
## pred      Negative Positive
## Negative    20         5
## Positive    10        67
```

Displaying the accuracy:

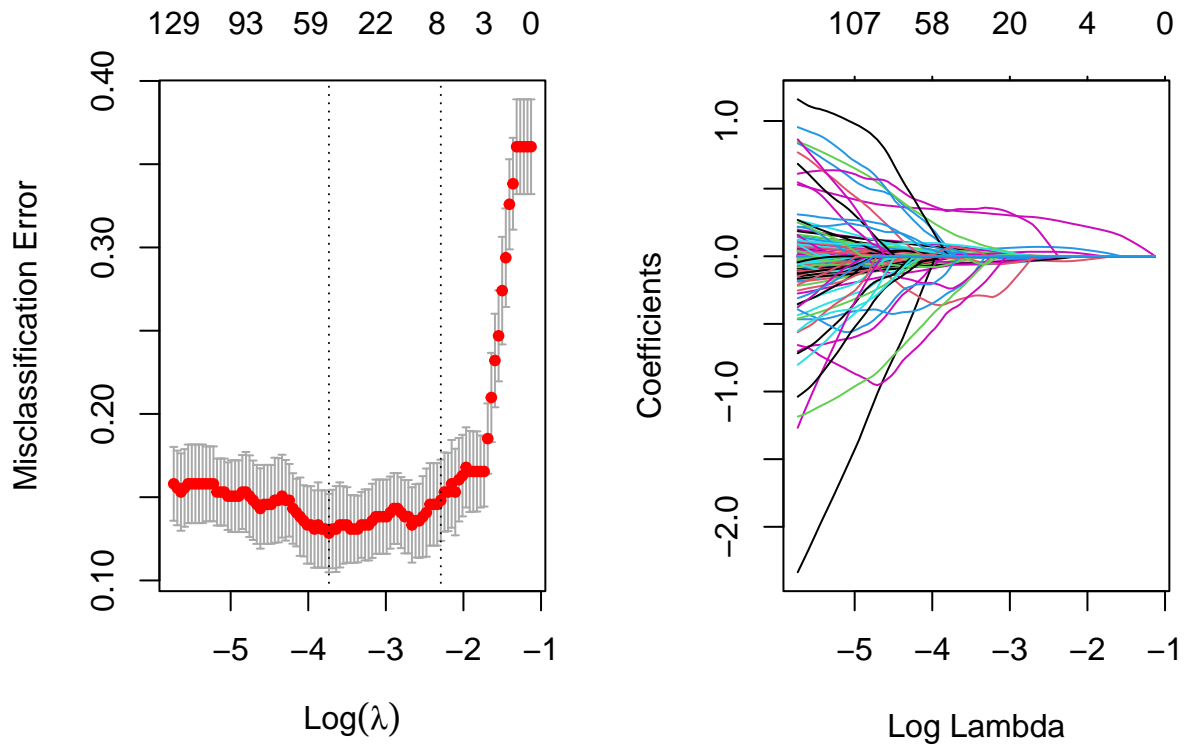
```
## [1] 0.8529412
```

The classification error for a model with k=7 is $1 - 0.8529412 = 0.1470588$.

Modeling with Lasso (predict PR.Status)

Next, we will create a Lasso model with 10-fold cross-validation to help predict `PR.Status`.

Plotting the fit of our Lasso model:



We will use the `best_lambda` to use for the prediction on the test data:

Displaying the confusion matrix of the results:

```
##          test_predict
##          Negative Positive
## Negative         22      8
## Positive         6      66
```

Displaying the accuracy:

```
## [1] 0.8627451
```

The classification error is $1 - 0.8627451 = 0.1372549$.

PR.Status Summary:

To summarize the model predictions for `PR.Status`, we used KNN and Lasso models to predict the outcome. We only used non-categorical data for both instances since the categorical variables gave many errors while we were in the process of fitting the models. However, even without the categorical predictors, both models yielded relatively low classification errors, with the KNN model (using $k=7$) having a classification error of 0.1470588 and the Lasso model having a classification error of 0.1372549. Looking at these results, the Lasso model would be better suited to predict `PR.Status`.

Modelling histological.type

Next, let us model `histological.type`, using AUC as the evaluation criterion. For both models that we chose to model `histological.type` with, we decided to use ALL of the categorical and non-categorical predictors (we still used the correlation matrix with `cutoff=0.7` to filter out highly correlated columns first) to see if we could get a more accurate result.

Modelling histological.type with SVM

We decided to test both radial and linear SVM to see which SVM model would perform better.

Let us display the SVM results for radial and linear models respectively:

```
svm.radial
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 405 samples
## 979 predictors
## 2 classes: 'infiltrating ductal carcinoma', 'infiltrating lobular carcinoma'
##
## Pre-processing: centered (979), scaled (979)
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 324, 325, 324, 323, 324
## Resampling results across tuning parameters:
##
##  C      sigma  Accuracy  Kappa
##  0.01  1      0.9037285  0
##  0.01  2      0.9037285  0
##  0.01  3      0.9037285  0
##  0.10  1      0.9037285  0
##  0.10  2      0.9037285  0
##  0.10  3      0.9037285  0
##  0.50  1      0.9037285  0
##  0.50  2      0.9037285  0
##  0.50  3      0.9037285  0
##  1.00  1      0.9037285  0
##  1.00  2      0.9037285  0
##  1.00  3      0.9037285  0
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 3 and C = 0.01.
```

```
svm.linear
```

```
## Support Vector Machines with Linear Kernel
##
## 405 samples
## 979 predictors
## 2 classes: 'infiltrating ductal carcinoma', 'infiltrating lobular carcinoma'
##
## Pre-processing: centered (979), scaled (979)
## Resampling: Cross-Validated (5 fold)
```



```
## Summary of sample sizes: 325, 323, 324, 324, 324
## Resampling results across tuning parameters:
##
##      C      Accuracy   Kappa
##  0.01 0.9234831 0.5484948
##  0.10 0.8840334 0.3695863
##  0.50 0.8840334 0.3695863
##  1.00 0.8840334 0.3695863
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was C = 0.01.
```

Next, we will use each model to predict on the test data:

```
linear_svm_prediction <- predict(svm.linear, test_data[, -1])
radial_svm_prediction <- predict(svm.radial, test_data[, -1])
# Radial --> unable to classify any infiltrating lobular
# carcinoma. Linear is the better model
suppressMessages(library(ROCR))
lin.svm <- prediction(as.numeric(as.factor(linear_svm_prediction)),
  as.factor(test_data[, 1]))
rad.svm <- prediction(as.numeric(as.factor(radial_svm_prediction)),
  as.factor(test_data[, 1]))
```

Finally, let us display the results of the Linear and Radial SVM AUC:

```
## [1] 0.6598402
```

```
## [1] 0.5
```

Looking at the results above, Linear SVM has $AUC = 0.6598402$ and Radial SVM has $AUC = 0.5$. We can conclude that Linear SVM is probably the better model due to this significant difference.

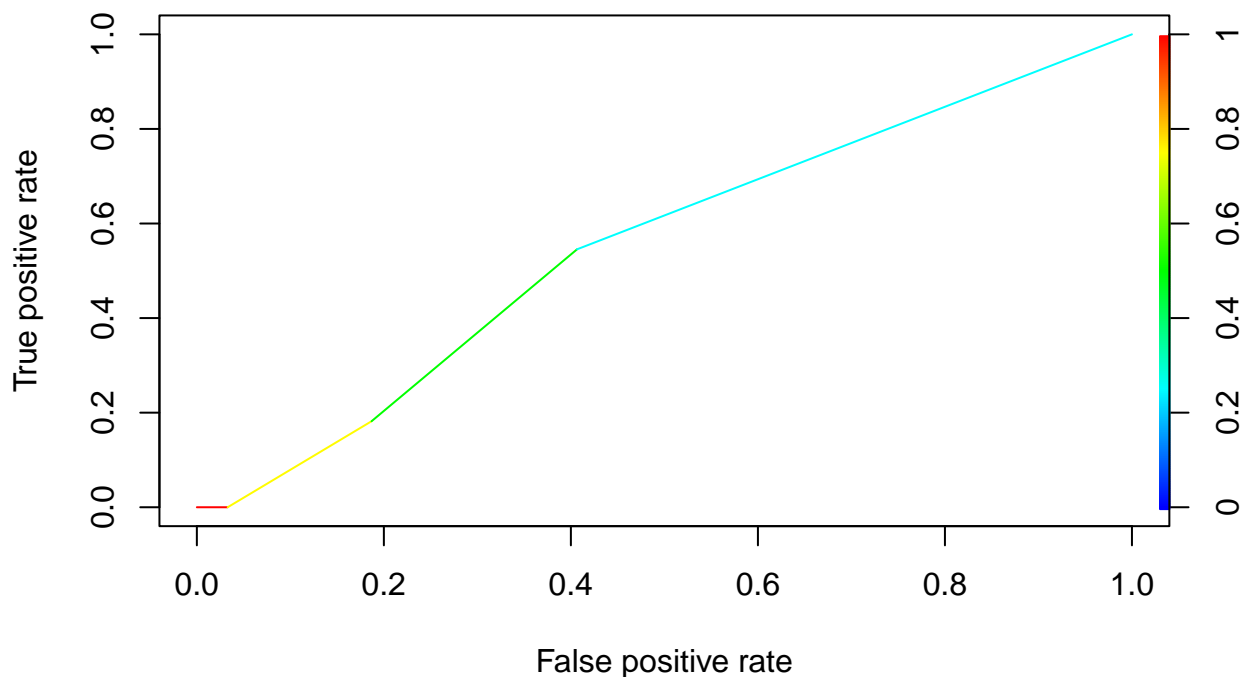
Modelling `histological.type` with Random Forests

Next, let us use a Random Forest model.

Let us fit the model with `nodesize=8` and `sampszie=50`. Through multiple rounds of experimentation with values, we found those parameters to be most optimal:

```
suppressMessages(library(randomForest))
set.seed(432)
rf.fit = randomForest(data.matrix(brca_train_2[, -c(640)]), y = as.factor(brca_train_2[,
  640]), ntree = 4, mtry = 4, nodesize = 8, sampsize = 50)
```

Next, use this model to predict on the test data and display the AUC plot:



The AUC:

```
## [1] 0.5524476
```

The AUC for the random forest model is 0.5524476.

histological.type Summary:

To summarize our models, we used Linear and Radial SVM, which had AUC values of 0.6598402 and 0.5 respectively. For random forest, the AUC value resulted in 0.5524476. Therefore, the Linear SVM model can be determined as the better model for predicting `histological.type`, although its AUC score is still a bit low.

50-Variable Selection

Starting with the cleaned data, we selected the 50 variables which had the highest correlation, using the correlation matrix approach as we did in earlier modelling attempts. We used 3-fold cross-validation, taking 2 folds for test and 1 fold for train each time.

```
suppressMessages(library(utiml))
suppressMessages(library(mldr))
suppressMessages(library(kableExtra))
suppressMessages(library(xgboost))
```

```

set.seed(1)
# Add Folds
brca_fold = read.csv("https://raw.githubusercontent.com/mgarbvs/STAT-432-final-project/mgarbus2/brca_data.csv")
brca_fold <- brca_fold[, -1937]
brca_fold$foldId <- sample(1:3, 705, replace = TRUE)
# organize the 4 outcome data
brca_fold <- brca_fold[((brca_fold$histological.type == "infiltrating lobular carcinoma") |
  (brca_fold$histological.type == "infiltrating ductal carcinoma")) &
  ((brca_fold$HER2.Final.Status == "Positive") | (brca_fold$HER2.Final.Status ==
    "Negative")) & ((brca_fold$ER.Status == "Positive") |
  (brca_fold$ER.Status == "Negative")) & ((brca_fold$PR.Status ==
    "Positive") | (brca_fold$PR.Status == "Negative")), ]

non_response_brca <- brca_fold[, -c(which(colnames(brca_fold) %in%
  c("PR.Status", "ER.Status", "HER2.Final.Status", "histological.type")))]
cor_mat = cor(non_response_brca[, -1937])
# returns vector of indices to remove
cor_list = findCorrelation(cor_mat, cutoff = 0.7)
non_response_brca <- non_response_brca[, c(cor_list, 1937)]
outcomes = brca_fold %>%
  select(c("PR.Status", "ER.Status", "HER2.Final.Status", "histological.type"))
response_data <- outcomes
response_data_n <- data.frame(sapply(response_data, function(x) as.numeric(as.factor(x)) -
  1))

# 50 highest corr vars with_factors
factor_data <- cbind(non_response_brca[c(1:50, 980)], response_data_n)

```

For the 50-variable selection model, we decided to consider a multi-label machine learning tool, `utiml`, so we would be able to predict all four labels from our 50 predictors. Serendipitously, this library is very similar to the WEKA tool used in Asri et al. which led us to consider using SVM for this application as well. Fortunately, the `utiml` package uses the package we used for SVM in this class, `e1071`. XGBoost is also used in this package, and is able to use AUC values as its objective function. We ultimately decided to use XGBoost, as it had slightly higher outcomes for our mean AUC values after cross validation when compared to the outcomes from SVM.

```

# Create two partitions (train and test) of toyml
# multi-label dataset
my_mldr <- mldr_from_dataframe(factor_data[, -51], labelIndices = c(52:54))
mldr_12 <- mldr_from_dataframe(factor_data[factor_data$foldId %in%
  c(1, 2), ], -51, labelIndices = c(51:54))
mldr_23 <- mldr_from_dataframe(factor_data[factor_data$foldId %in%
  c(2, 3), ], -51, labelIndices = c(51:54))
mldr_13 <- mldr_from_dataframe(factor_data[factor_data$foldId %in%
  c(1, 3), ], -51, labelIndices = c(51:54))

mldr_1_test <- mldr_from_dataframe(factor_data[factor_data$foldId %in%
  c(1), ], -51, labelIndices = c(51:54))
mldr_2_test <- mldr_from_dataframe(factor_data[factor_data$foldId %in%
  c(2), ], -51, labelIndices = c(51:54))
mldr_3_test <- mldr_from_dataframe(factor_data[factor_data$foldId %in%
  c(3), ], -51, labelIndices = c(51:54))

```

Finally, let us display our 3-fold cross-validated and averaged AUC:

```
# XGB MODEL
xgb_12_pred <- br(mldr_12, base.algorith = "XGB", seed = 1, cores = 1,
  eta = 0.3575, nrounds = 50, eval_metric = "auc", subsample = 1)
xgb_13_pred <- br(mldr_13, base.algorith = "XGB", seed = 1, cores = 1,
  eta = 0.3575, nrounds = 50, eval_metric = "auc")
xgb_23_pred <- br(mldr_23, base.algorith = "XGB", seed = 1, cores = 1,
  eta = 0.3575, nrounds = 50, eval_metric = "auc")

prediction_12_3_xgb <- predict(xgb_12_pred, mldr_3_test, type = "class")
prediction_13_2_xgb <- predict(xgb_13_pred, mldr_2_test, type = "class")
prediction_23_1_xgb <- predict(xgb_23_pred, mldr_1_test, type = "class")

xgb_results <- rbind(multilabel_evaluate(mldr_3_test, prediction_12_3_xgb,
  measures = c("macro-based", "micro-based"))[c(1, 5)], multilabel_evaluate(mldr_2_test,
  prediction_13_2_xgb, measures = c("macro-based", "micro-based"))[c(1,
  5)], multilabel_evaluate(mldr_1_test, prediction_23_1_xgb,
  measures = c("macro-based", "micro-based"))[c(1, 5)])

xgb_output <- rbind(xgb_results, colMeans(xgb_results))
rownames(xgb_output) <- c("folds 1 and 2 compared to 3", "folds 1 and 3 compared to 2",
  "folds 2 and 3 compared to 1", "mean AUC value")

kable(xgb_output)
```

	macro-AUC	micro-AUC
folds 1 and 2 compared to 3	0.8193707	0.9301318
folds 1 and 3 compared to 2	0.7733339	0.9015268
folds 2 and 3 compared to 1	0.8041382	0.9129050
mean AUC value	0.7989476	0.9148545

The first 3 rows correspond to the micro and macro AUC for the different fold comparisons. The macro AUC and micro AUC are the average AUC for each of the 4 outcomes at that CV level. Finally, the last row is the averaged out AUC value for all folds.

As shown on the table, the micro-AUC is significantly higher than the macro-AUC. Using the 50-variable model, we were able to result in an average micro-AUC of 0.9148545.

We believe that the micro-AUC is the preferred metric as we the data has a class imbalance and the micro-AUC accounts for that.