Barbara Schweitzer

IST 707 – Final Project Report

Breast cancer, in the United States, is one of the most common types of cancer among women. In the US, between the years 1999 and 2017, approximately 41,000 people a year have died from breast cancer. Though the rate of breast cancer deaths has been decreasing since 1999, the number of cases of breast cancer has been rising, thus the number of deaths has remained relatively constant. Based on the attributes of a person and their specific cancer, it is important to understand the likelihood of survival of patients. Using different analysis techniques, a meaningful prediction for cancer recurrence based on observed attributes of patients and their disease will hopefully be found.

The data set chosen to be studied is a Breast Cancer Data Set, from Kaggle. This data set contains 2,509 entries, with 34 attributes. The attributes include numerous different categorizations and elements of the specific patient’s cancer, different treatments the patient has received, as well as relevant information regarding the patient and their health. Among these variables, there are different possibilities for classification in this data set. Based on further exploration into the data and brief experimental tests, the patient’s relapse status was the best variable to use as the target variable.

Using this data set, the most useful form of analysis will likely be Naïve Bayes, K-Nearest Neighbor, and Decision Tree analysis. I believe that Naïve Bayes will provide useful results because the data has many categorical variables, and the class attribute is independent for each patient. K-Nearest Neighbor will also be a helpful form of classification for this data, due to the type of data provided. Since the data is medical in nature, it is common for prognoses of diseases to be given based on patients having symptoms or disease characteristics similar to another person suffering from the illness. Finally, Decision Tree analysis will likely be an effective form of analysis because the data on the disease leads to a certain binary outcome, being if the disease recurs or not, which the decision tree will assist in providing.

To begin the data analysis, the data was first processed. In the raw data, there were approximately five hundred data entries that were almost entirely blank, with only one or two of the recorded attribute values available. To simplify the data and handle the missing values, the entries were removed, which left 1986 data points for analysis. To further simplify the data, the relapse status attribute was changed from the values of Recurred and Not Recurred to the dummy variables 1 and 0 respectively. This allowed the data to be more easily analyzed. Because there were thirty-four variables, it was necessary to find the variables that were most significantly related to the relapse status of patients. Thus, information gain was used on the variables to see their importance, and the top ten variables, excluding the patient identification number, were used in the analysis. The variables chosen were Pam50 + Claudin-low Subtype, Integrative Cluster, Lymph Nodes Examined Positive, Nottingham Prognostic Index, Overall Survival in Months, Overall Survival Status, Relapse Free Status in Months, Tumor size, Tumor Stage, and Patient’s Vital Status. The attribute importance values are shown in the image below.

Text

Description automatically generated

To complete the overall data preprocessing, the relapse status variable was changed from a numeric value to a factor, so that it could be used as the class in the classification algorithms. For the kNN algorithm, there will be further data processing done, which will be completed after the Decision Tree and Naïve Bayes algorithms.

To begin the data analysis, it was important to create training and testing data sets, so that the algorithms can be tested on sample data to check the accuracy of the classification. To make the different sets of data, a seventy/thirty split was used. Thus, seventy percent of the data was used for the training set, and thirty percent of the data was used as the testing set. This left 1390 data points for training and 595 data points for testing.

The first algorithm utilized was a decision tree model. To create this model, the rpart and rpart.plot packages were used. With the rpart model, the training data was utilized, with the Relapse Free Status variable used as the class. The tree diagram produced is shown below.

Diagram, timeline

Description automatically generated

With the predict function, the accuracy of the model was measured. Using the testing data, the model was 96.30252% accurate. Though the algorithm was very accurate, the results may have been slightly skewed. As shown in the tree above, the classification is entirely based on attributes that are directly related to the outcome a patient’s disease, whether it be the number of months without a relapse, the patient’s vital status, or their overall survival time. Because these variables are closely related to the patient’s relapse status, there may be a strong correlation between the attributes that could cause the model to be inaccurate. A confusion matrix with the relapse status of the patients from the data and the predicted classes was created and is shown below.

Text

Description automatically generated

There were twenty-one false negatives and one false positive. With medical data, the false negative results are much more detrimental. If a care provider utilized this test to predict if a patient had a recurrence of cancer and a false positive result was given, the patient may not be able to get the necessary treatments to try to improve their condition. Therefore, though the algorithm had a high accuracy rate, the false negative results were potentially concerning.

The next algorithm utilized was Naïve Bayes. Using the e1071 package, two Naïve Bayes algorithms were created, one without Laplace smoothing, and one with. The first model, without the smoothing, resulted in an 88.06723% accuracy after running it on the testing data set. A confusion matrix with the relapse status of the patients from the data and the predicted classes was created and is shown below.

Graphical user interface, text

Description automatically generated

From this model, there were thirty-eight false negatives and thirty-three false positives. As seem in the decision tree model, there are more data entries incorrectly classified as false negative, which is much more dangerous in medical cases. Even with the relatively high accuracy rate for this model, the false negative results should be taken into consideration.

Next, the Naïve Bayes model was run with Laplace smoothing. Laplace smoothing can be utilized to correct zero probability problems in categorical data. In this case, especially because the data was preprocessed, the Laplace smoothing did not affect the accuracy of the model, and the accuracy remained at 88.06723%, with thirty-eight false negative classifications and 33 false positive classifications.

The next and final algorithm run was k-Nearest Neighbor. To run this algorithm, extra preprocessing was necessary. Utilizing the class package, the kNN algorithm would not run with the categorical variables in the data, and thus they were changed to dummy variables. The attributes changed were Pam 50 + Claudin-low Subtype, Integrative cluster, Overall Survival Status, and Patient’s Vital Status. The Pam 50 + Claudin-low Subtype attribute was changed to the numbers one through seven to represent the different values. The values were transformed from Basal, Claudin-low, Her2, LumA, LumB, NC, and Normal, to one through seven, respectively. For integrative cluster, the values in the data consisted of the numbers one through ten, 4ER- , and 4ER+. The values in the data represented the molecular subtype of the cancer, based on certain gene expression, thus they were not numeric in nature. Therefore, the 4ER- and 4ER+ entries were changed to the numbers eleven and twelve, to represent these molecular subtypes. The Overall Survival Status attribute consisted of Deceased and Living values, which were changed to zero and one, respectively. Finally for the Patient’s Vital Status, the three values were Died of Disease, Died of Other Causes, and Living, which were changed to the values one, two, and three, respectively. With this new data representation, the data was reloaded into R. The newly changed attributes were changed to factor variables, rather than numeric, and testing and training data sets were created. For the kNN algorithm to run, the missing data values needed to be removed using the na.omit function. To complete the preprocessing, the class attribute, Relapse Free Status, was removed from the training and testing data sets, and moved to new label data tables. The final training data set had 985 data entries, and the testing data set had 414 data entries.

For the first kNN model, the k value was set at fifteen. The model had an 84.54106% accuracy when run on the testing data. There were forty-three false negatives and twenty-one false positives, which was the lowest scoring model run thus far. The classifications from the data compared with the model predictions are shown below.

Text

Description automatically generated with low confidence

To try to achieve a better accuracy and misclassification rate, other k-values were tested.

The next kNN model utilized a k value of thirty. This model resulted in an 81.15942% accuracy, with fifty-two data entries misclassified as false negative and twenty-six data entries as false positive. The classifications from the data compared with the model predictions are shown below.

A picture containing calendar

Description automatically generated

This model ended up being the least accurate kNN model for this data, making this model not useful. Thus, one more k-value was tested.

The last kNN model utilized a k value of five. This model resulted in an 86.23188% accuracy, with forty data entries misclassified as false negative and seventeen data entries as false positive. The classifications from the data compared with the model predictions are shown below.

A picture containing text

Description automatically generated

This model ended up being the most accurate kNN model for this data, however the accuracy and misclassified values made this model not as successful as the other algorithms. Overall, the k-Nearest Neighbor algorithm was the least successful at classifying the data. The k value of five had the highest accuracy, however it did not compare with the Decision Tree or Naïve Bayes models, especially when taking into consideration the types of misclassifications present.

After the models were all built, it was clear that there was not an obvious choice of which model was best. The Decision Tree model had the highest accuracy, but the model was built using data that may skew the results, especially when a doctor will not necessarily have the survival status, etc., from patients that have a chance of disease relapse. The Naïve Bayes model had the second highest accuracy and seemed to have a sounder basis for the classifications, however the accuracy was below ninety percent, and there were many false negative misclassifications. The k-Nearest Neighbor models were clearly the worst in both the accuracy and misclassifications. Due to the data being medical in origin, it is very important that the models are highly accurate and also have a low frequency of dangerous false negative results. Because of this concern, none of the algorithms run on the subset of data chosen would be likely candidates for a real-world application to be used in medical diagnoses. However, the best algorithm from the ones run above would likely be the Naïve Bayes model, because it had a relatively high accuracy and was likely created on a sounder basis than the Decision Tree model.

For further analysis of this data, there are many different options for changes that could be made. One possible route could be to remove the attributes that are closely related to the relapse status, such as the Relapse Status in Months and the Patient’s Vital Status. After removing these specific attributes, the algorithms could also be executed on the entire data set, including the variables with lower information gain values. Also, the classification algorithms could use a different variable as the target, which may provide more useful results. One final possibility could be to run other algorithms, including, but not limited to, Neural Networks, Support Vector Machines, or Ensemble Learning techniques.

**Appendix**

**Data Source :** <https://www.kaggle.com/gunesevitan/breast-cancer-metabric>

**Data /Attribute Description :** <https://www.kaggle.com/raghadalharbi/breast-cancer-gene-expression-profiles-metabric>

**Code :**

# Looking at the information gain to choose all relevant variables

originalData <- read.csv("MS - Quarter II/Data Analytics/Breast Cancer METABRIC-Original.csv")

originalData$`Relapse Free Status` <- as.factor(originalData$`Relapse Free Status`)

# install.packages("FSelector")

library("FSelector")

infoGainOriginal <- information.gain(`Relapse Free Status`~., originalData)

infoGainOriginal

# Loading data set with only relevant variables and missing data removed

library(readr)

bcData <- read\_csv("MS - Quarter II/Data Analytics/Breast Cancer METABRIC-Final(words).csv")

bcData$`Relapse Free Status` <- as.factor(bcData$`Relapse Free Status`)

str(bcData)

# Looking at the information gain for these variables

# install.packages("FSelector")

library("FSelector")

infoGain <- information.gain(`Relapse Free Status`~., bcData)

infoGain

# Create training and testing data sets

# install.packages("caret")

library("caret")

set.seed(3456)

trainIndex <- createDataPartition(bcData$`Relapse Free Status`, p = 0.7, list = FALSE, times = 1)

trainData <- bcData[trainIndex, ]

testData <- bcData[-trainIndex, ]

write.csv(trainData, 'bcDataTrainSet.csv')

write.csv(testData, "bcDataTestSet.csv")

# Removing the Patient ID

trainData <- trainData[ , -1]

testData <- testData[ , -1]

**# Decision tree classification**

library(rpart)

library(rpart.plot)

model <- rpart(`Relapse Free Status`~., data = trainData, method = 'class')

rpart.plot(model)

prediction <-predict(model, testData, type = 'class')

table\_mat <- table(testData$`Relapse Free Status`, prediction)

table\_mat

treeAccuracy <- ((table\_mat[1,1] + table\_mat[2,2])/nrow(testData))\*100

treeAccuracy

# 96.30252% accuracy -- more false negatives (21) vs 1 false positive

**# Naive Bayes**

library(e1071)

NBmodel <- naiveBayes(`Relapse Free Status`~ ., data = trainData, na.action = na.pass)

predict(NBmodel, testData)

pred <- predict(NBmodel, testData)

table(testData$`Relapse Free Status`, pred)

NBaccuracy <- ((321+203)/595)\*100

NBaccuracy

#88.06723% accurate -- 38 false negatives and 33 false positives

# Using Laplace Smoothing

NBmodelLS <- naiveBayes(`Relapse Free Status`~ ., data = trainData, laplace = 1, na.action = na.pass)

predLS <- predict(NBmodelLS, testData)

table(testData$`Relapse Free Status`, predLS)

# Accuracy was the same as without Laplace Smoothing

**# K-Nearest Neighbor classification**

# Using the data set with dummy variables for all of the categorical variables

bcDataN <- read\_csv("MS - Quarter II/Data Analytics/Breast Cancer METABRIC-Final(numeric).csv")

bcDataN$`Pam50 + 2 subtype` <- as.factor(bcDataN$`Pam50 + 2 subtype`)

bcDataN$`Integrative Cluster` <- as.factor(bcDataN$`Integrative Cluster`)

bcDataN$`Overall Survival Status` <- as.factor(bcDataN$`Overall Survival Status`)

bcDataN$`Relapse Free Status` <- as.factor(bcDataN$`Relapse Free Status`)

bcDataN$`Patient's Vital Status` <- as.factor(bcDataN$`Patient's Vital Status`)

str(bcDataN)

# Create training and testing data sets

# install.packages("caret")

library("caret")

set.seed(3456)

trainDataN <- bcDataN[trainIndex, ]

testDataN <- bcDataN[-trainIndex, ]

write.csv(trainDataN, 'bcDataTrainSetN.csv')

write.csv(testDataN, "bcDataTestSetN.csv")

# Removing the Patient ID

trainDataN <- trainDataN[ , -1]

testDataN <- testDataN[ , -1]

# Remove Relapse Status from the data and remove na's

noNATrainDataN <- na.omit(trainDataN)

noNATestDataN <- na.omit(testDataN)

trainDataKnn <- noNATrainDataN[, -8]

testDataKnn <- noNATestDataN[, -8]

relapseTrainLabels <- noNATrainDataN[, 8]

relapseTestLabels <- noNATestDataN[, 8]

# k = 15

classPredictionK15 <- knn(train = trainDataKnn, test = testDataKnn, cl = relapseTrainLabels$`Relapse Free Status`, k=15)

CrossTable(x = relapseTestLabels$`Relapse Free Status`, y = classPredictionK15, prop.chisq=FALSE)

k15NNAccuracy <- ((223+127)/414)\*100

k15NNAccuracy

# 84.54106% accuracy -- more false negatives (43) and false positives (21)

# k = 30

classPredictionK30 <- knn(train = trainDataKnn, test = testDataKnn, cl = relapseTrainLabels$`Relapse Free Status`, k=30)

CrossTable(x = relapseTestLabels$`Relapse Free Status`, y = classPredictionK30, prop.chisq=FALSE)

k30NNAccuracy <- ((218+118)/414)\*100

k30NNAccuracy

# 81.15942% accuracy -- more false negatives (52) and false positives (26)

#k = 5

classPredictionK5 <- knn(train = trainDataKnn, test = testDataKnn, cl = relapseTrainLabels$`Relapse Free Status`, k=5)

CrossTable(x = relapseTestLabels$`Relapse Free Status`, y = classPredictionK5, prop.chisq=FALSE)

k5NNAccuracy <- ((227+130)/414)\*100

k5NNAccuracy

# 86.23188% accuracy -- fewer false positives (17), lower number of false negatives (40)