Mammographic Mass Data Set

Sergio Castillo, 1513228@rgu.ac.uk

22nd November 2019

Machine Learning and Big Data Analytics (CM3111)
School of Computing Science & Digital Media
Robert Gordon University

Contents

1 Overview

The purpose of this report is to analyze a given dataset, in order to reach certain conclusions and be able to predict an outcome with the highest accuracy possible, thus showing competence in understanding, identifying, applying and evaluating machine learning algorithms. In order to do that, I will be using a data set focused on the prediction of malicious mammographic masses based on the result of BI-RAD (Breast Imaging-Reporting and Data System) assessments.

2 Data Exploration

2.1 The Dataset

I've chosen this dataset because I am interested in learning how data analysis and machine learning can be used in order to save people's lives. Additionally, the number of attributes and instances seemed adequate for analysis.

2.1.1 Dataset Name

Mammographic Mass

2.1.2 Dataset Source

The dataset was obtained from the UCI Machine Learning Repository[1]. The origin of the dataset is a paper published by M.Elter, R.Schulz-Wendtland and T.Wittenberg, titled **”The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process”**[2].

2.1.3 Dataset Acquirement Date

The dataset was obtained on the 4th of October 2019
The original paper was published in 2007, and the dataset was donated on the 29th of October, 2007.

2.2 Problem Statement & Data Exploration

This dataset contains information of 961 patients' BI-RAD assessment, having different parameters of the BI-RAD attributes, patient's age and the result of such assessment Mammography is the most effective method for breast cancer screening available today. However, the low positive predictive value of breast biopsy resulting from mammogram interpretation leads to approximately 70% unnecessary biopsies with benign outcomes. To reduce the high number of unnecessary breast biopsies, several computer-aided diagnosis (CAD) systems have been proposed in the last years.These systems help physicians in their decision to perform a breast biopsy on a suspicious lesion seen in a mammogram or to perform a short term follow-up examination instead.

2.2.1 Objective

This data set can be used to predict the severity (benign or malignant) of a mammographic mass lesion from BI-RADS (Breast Imaging-Reporting and Data System) attributes and the patient's age.

```
# Dataset dimensions
nrow(df)

## [1] 961

ncol(df)

## [1] 6

dim(df)

## [1] 961   6

# Show some rows
df[1:10,]

##    Assessment Age Shape Margin Density Severity
## 1           5  67     3      5       3        1
## 2           4  43     1      1    <NA>        1
## 3           5  58     4      5       3        1
## 4           4  28     1      1       3        0
## 5           5  74     1      5    <NA>        1
## 6           4  65     1   <NA>       3        0
## 7           4  70  <NA>   <NA>       3        0
## 8           5  42     1   <NA>       3        0
## 9           5  57     1      5       3        1
## 10          5  60  <NA>      5       1        1
```

2.2.2 Attributes

The attributes provide information about the patients' age, mammograpic mass results and the severity of the mass. There are 6 attributes in total, from which 4 are predictive attributes, 1 is non-predictive and 1 is the goal field.

1. **Assessment**: Results of the BI-RADS, an indication of how well a CAD system performs compared to the radiologists
   - 0: Definitely benign - 6: Highly suggestive of malignacy
   - Ordinal Categorical Variable
   - Non-Predictive Attribute
2. **Age**: Patient's age in years
   - Integer Variable
   - Predictive Attribute
3. **Shape**: Mass shape
   - 1: Round, 2: Oval, 3: Lobular, 4: Irregular
   - Nominal Categorical Variable
   - Predictive Attribute
4. **Margin**: Mass margin
   - 1: Circumscribed, 2: Microlobulated, 3: Obscured, 4: Ill-defined, 5: Spiculated
   - Nominal Categorical Variable
   - Predictive Attribute
5. **Density**: Mass density
   - 1: High, 2: Iso, 3: Low, 4: Fat-containing
   - Ordinal Categorical Variable
   - Predictive Attribute
6. **Severity**: Severity of the mammographic mass
   - 0: Bening, Malignant: 1
   - Binomial Categorical Variable
   - Goal Attribute

```
# Show Column names
names(df)

## [1] "Assessment" "Age"        "Shape"      "Margin"     "Density"
## [6] "Severity"
```

2.2.3 Class Distribution

The goal field on the dataset is the severity of the mammographic mass, which can either be benign (i.e Non-Harmful) or malicious.

```
# Finding how many unique fields there are, and the unique values
length(unique(df$Severity))

## [1] 2

unique(df$Severity)

## [1] 1 0
## Levels: 0 1
```

As shown in the bar plot below, the dataset contains 961 entries, for 516 benign and 445 malignant masses.

```
# Class distribution
severityFreq <- table(df$Severity)
barplot(
  severityFreq,
  col = c('steelblue4','coral1'),
  main="Severity Frequency",
  names.arg=c("Benign","Malicious"),
  ylab="Number of patients"
)
```

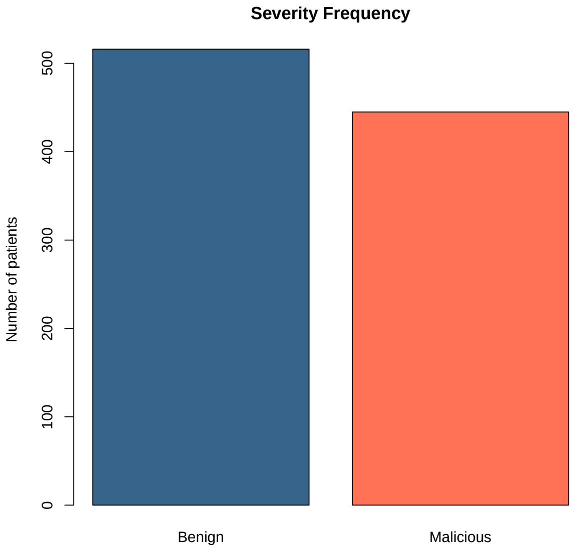

Figure 1: Class distribution of the severity

2.2.4 Dataset Summary

The summary displays a briefing of every column, showing statistical iformation about the data in every entry. The summary can either be numerical (Which shows information such as the maximum and minimum values, median and quartiles) or categorical, which shows the frequency of each one of the categorical values.
In this dataset, only the Patient's age is numerical. The rest of the attributes only show the frequency.

Age Summary

The patient Age attribute is a numerical attribute, so the summary shows the minimum, first quartile, median, mean, third quartile and maximum values of the data

```
# Age    Summary
summary(df$Age)
```

Min	1st Qu	Median	Mean	3rd Qu	Max	Missing
18.00	45.00	57.00	55.49	66.00	96.00	5

```
Age <- df$Age
hist(
  Age,
  xlab="Patient Age",
  col="turquoise3",
)
```

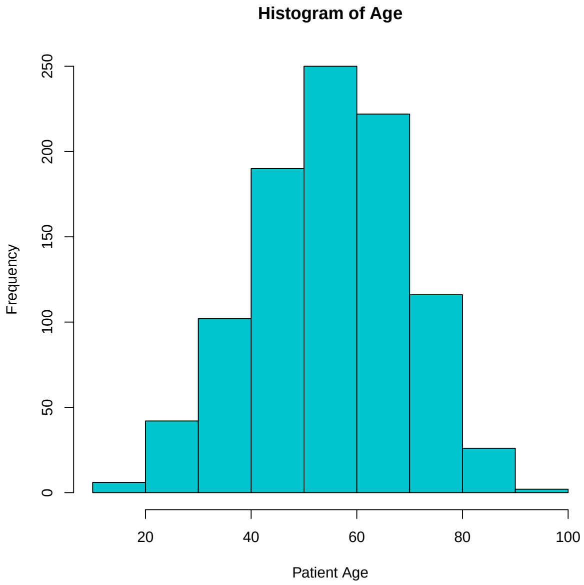

Figure 2: Histogram of the Patients' Age

Shape Summary

The mass Shape attribute is a categorical attribute, so the summary shows the value frequency

```
# Shape Summary
summary(df$Shape)
```

Round	Oval	Lobular	Irregular	Missing
224	211	95	400	31

Margin Summary

The mass Margin attribute is a categorical attribute, so the summary shows the value frequency

```
# Margin Summary
summary(df$Margin)
```

Circumscribed	Microlobulated	Obscured	Ill-defined	Spiculated	Missing
357	24	116	280	136	48

Density Summary

The mass Density attribute is a categorical attribute, so the summary shows the value frequency

```
# Density Summary
summary(df$Density)
```

High	Iso	Low	Fat-Containing	Missing
16	59	798	12	76

2.2.5 Attribute Correlation

Correlation measures the relationship between two measurements. This dataset only contains one numerical attribute, so there's no chance to create a numerical correlation between the different values. However, a relationship can be established between the age of the patient and the severity of the mammographic mass.This relation is known, but the other attributes also contribute to it.

```r
# Finding how attributes compare to goal
plot(
  data = df,
  Age~Severity,
  col = c('steelblue4','coral1'),
  names.arg=c("Benign","Malicious")

)
```

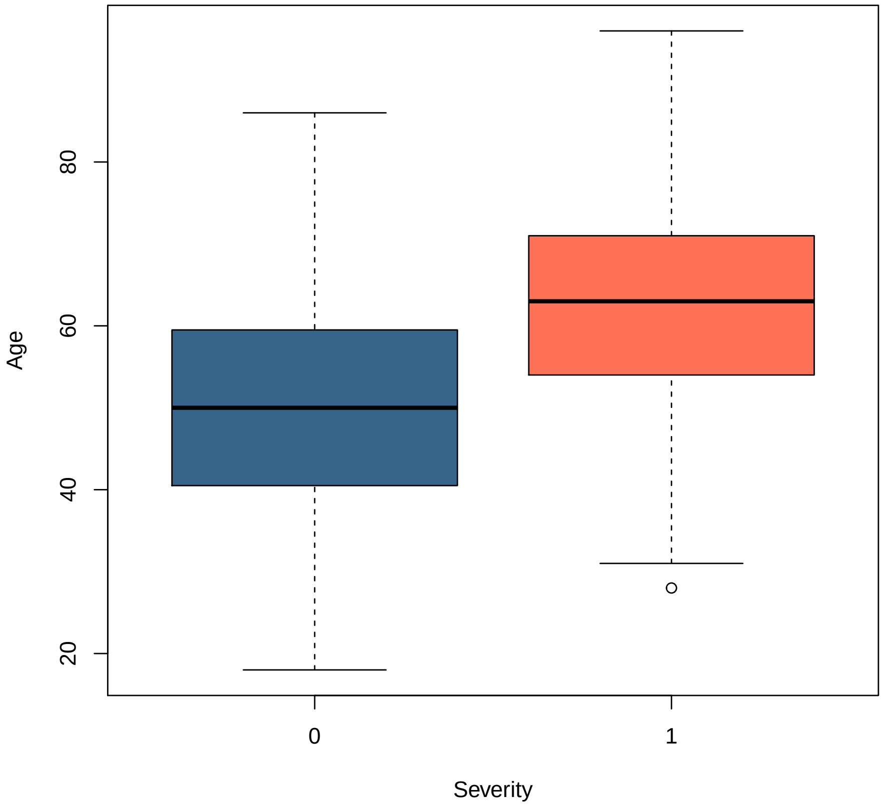

Figure 3: Correlation between the Age and the severity of the Mammographic mass

In the following page, correlations 3 Age and different Mammographic attributes can be found [4], [5], [6]. As mentioned, there is a relation between the age and the type of mass, but it's not enough to reach a conclusion.

```r
library(datasets)
data <- transform(df, Shape)
boxplot(Age~Shape, data, xlab="Shape", ylab="Age")
```

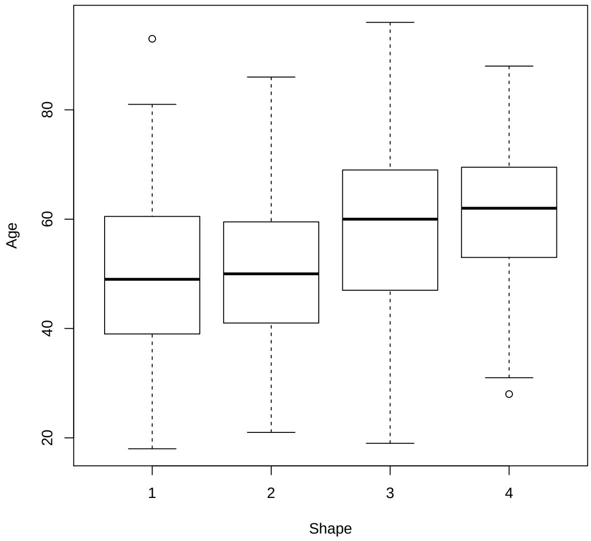

Figure 4: Relation between Age and Mass Shape (Round, Oval, Lobular, Irregular)

```
data <- transform(df, Margin)
boxplot(Age~Margin, data,xlab="Margin",ylab="Age")
```

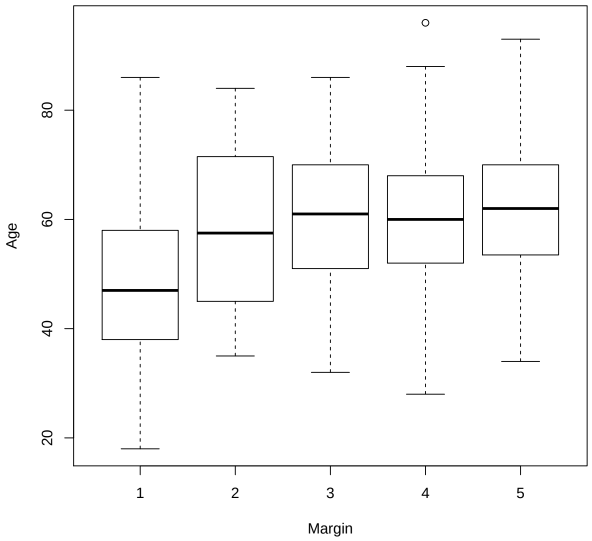

Figure 5: Relation between Age and Mass Margin (Circumscribed, Microlobulated, Obscured, Ill-defined, Spiculated)

```
data <- transform(df, Density)
boxplot(Age~Density, data,xlab="Density",ylab="Age")
```

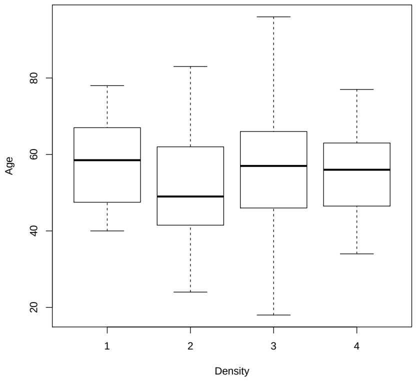

Figure 6: Relation between Age and the Mass Density (High, Iso, Low, Fat-containing)

3 Data Pre-processing

In order to analyze the data, the attributes must first be adequated for study. This is known as pre-processing

3.1 Irrelevant Columns

Before going any further, it has been previously mentioned that the Assessment column is a non-predictive attribute, since it measures the effectiveness of CAD Systems. Therefore, the column is not relevant for data mining, and it should be dropped.

```
# Backing up the data frame
df_full <- df

# Dropping the irrelevant column
df$Assessment <- NULL

# Showing the data frame columns without the dropped one
names(df)

## [1] "Age"        "Shape"      "Margin"     "Density"    "Severity"
```

3.2 Missing Values

One of the main issues with the dataset is the number of missing values. When taking a look at the raw data, it can be appreciated that the way missing values were handled was by writing a question mark "?" in place of the value.
The data frame will need to have those missing values recognized, so when parsing the file as a data frame, missing values must be specified

```
# Importing the data
dataFrame <- read.table('data/mammographic_masses.data',
                        header=FALSE, sep=",", na.strings="?")

# Showing some rows
df[1:5,]

##   Age Shape Margin Density Severity
## 1  67     3      5       3        1
## 2  43     1      1    <NA>        1
## 3  58     4      5       3        1
## 4  28     1      1       3        0
## 5  74     1      5    <NA>        1
```

Now that missing values have been identified, it is necessary to know which values are missing:

```
sum(is.na(df))

## [1] 160

sum(is.na(df$Age))

## [1] 5

sum(is.na(df$Shape))

## [1] 31

sum(is.na(df$Margin))

## [1] 48

sum(is.na(df$Density))

## [1] 76

sum(is.na(df$Severity))

## [1] 0
```

Age	Shape	Margin	Density	Severity	Total
5	31	48	76	48	162

3.2.1 Filling missing values

With numerical values, a way to deal with missing values is by replacing the blank attribute with the mean of such attribute. Since the age is the only numerical attribute, we can only use this method here.

```r
# Defining the function
fillMissings <- function(x) {
  replace(
    x,
    is.na(x),
    mean(x[!is.na(x)]))
}

# Applying the function to the Age column
df$Age <- fillMissings(df$Age)

# Verifying the number of missing values on the age column
sum(is.na(df$Age))

## [1] 0
```

3.2.2 Deleting rows

Since we cannot fill the missing values, due to the categorical nature of the attributes, we have no other option but to drop the rows that contain them.

```r
# Deleting rows with misssing factor attributes
df <- df[!is.na(df$Shape),]
df <- df[!is.na(df$Margin),]
df <- df[!is.na(df$Density),]
```

We can now observe that there are no missing values in our data frame

```r
sum(is.na(df))

## [1] 0
```

3.3 Missing Header

Another issue with the dataset is the lack of headers. We may know the meaning of every column after reading about the dataset, but we still need a way to identify each one of the columns.

```
# Showing the data frame header
names(dataFrame)

## [1] "V1" "V2" "V3" "V4" "V5" "V6"

# Changing the column names
names(dataFrame) <- c("Assessment","Age","Shape","Margin","Density", "Severity")

# Showing the column names
names(dataFrame)

## [1] "Assessment" "Age"        "Shape"      "Margin"     "Density"
## [6] "Severity"
```

3.4 Categorical Values

As previously mentioned, most of the values are categorical, rather than numerical. This means that it can't just be measured as a number, but as propertie of the entries. After dropping unnecessary columns, the categorical attributes are the following:

1. **Shape**: Shape of the mammographic mass. (Nominal)
2. **Margin**: Margin of the mammographic mass. (Nominal)
3. **Density**: Density of the mammographic mass. (Ordinal)
4. **Severity**: Severity of the mammographic mass. (Binominal)

Fortunately enough, the categorical attributes already come as numerical values. If this was not the case and they were represented as, for instance, strings, such strings would need to be converted into numerical values.
Now, the attributes will be converted into categorical values.

```
# Turning the attributes into categofical
df$Shape      <- factor(df$Shape)
df$Margin     <- factor(df$Margin)
df$Density    <- factor(df$Density, order=TRUE, levels=c("1","2","3","4"))
df$Severity   <- factor(df$Severity)

# Checking whether the fields are factors
is.factor(df$Age)
```

```
## [1] FALSE

is.factor(df$Shape)

## [1] TRUE

is.factor(df$Margin)

## [1] TRUE

is.factor(df$Density)

## [1] TRUE

is.factor(df$Severity)

## [1] TRUE
```

3.5 Data Normalization

Since there are no noticeable irregularities in the dataset, such as outliers or skewed values, there is no need to normalize the dataset.

4 Modelling

Now that the dataset has been explored and there is a better idea of what needs to be predicted, we can start building and testing models.

```
# Downloading necessary libraries
library(ISLR)
library(dplyr)
library(ROCR)
```

4.1 Creating Subsets

Before creating predictive models, the dataset will be split into a training and a testing set, in order to avoid future problems with overfitting or underfitting.

```
# Dividing a dataset into train and test
createSubsets <- function (dataset, percentage) {

  # Getting the training dataset
  train<-sample_frac( dataset , percentage)
  sid<-as.numeric(rownames(train))

  # Getting the test dataset
  validation <- dataset[-sid,]

  # Returning the 2 subsets
  return(list("train"=train, "validation"=validation))
}

# Retrieving the training and validation subsets
subset <- createSubsets(df, 0.7)

# Retrieving the training subset
train <- subset$train
nrow(train)

## [1] 585

# Retrieving the test subset
validation <- subset$validation
nrow(validation)

## [1] 251
```

```
# Verify that the dataset was split correctly
testSize <- function(dataset, train, test) {
  return(
    ifelse(nrow(train) + nrow(validation) == nrow(dataset),
    "Properly split",
    "Not properly split")
  )
}

print( testSize(df, train, validation) )

## [1] "Properly split"
```

- **Training**: The training subset represents a bigger portion of the initial set, and it will be used to create the model.
- **Testing**: Also known as the Validation subset, it will be used to test the performance of the model with new data.

4.2 Implementing a Model

Now that the dataset has been split, the training subset can be used to start designing the model. For this case, Logistic Regression will be used. Logistic regression is used to model the probability that an ‘event’ will occur. In other words, the dependent variable, now p(X), assumes a value between 0 and 1. This is suitable for the current dataset, since what we are trying to predict is whether a mammographic mass can be benign of malicious (0 or 1).

```
# Implementing the Logistic Regression model

createRegressionModel <- function (dataset) {
  return( glm(
    Severity ~ Age+Shape+Margin+Density, # Severity against parameters
    data=dataset,                        # Using training data
    family=binomial                      # Goal is a binomial value
  ))
}

# Spitting the summary of the model
regressionModel <- createRegressionModel(train)
summary(regressionModel)

##
## Call:
## glm(formula = Severity ~ Age + Shape + Margin + Density, family = binomial,
##     data = dataset)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5276  -0.5524  -0.2079   0.6493   2.6392
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.956289   0.685872  -7.226 4.96e-13 ***
## Age          0.053218   0.009676   5.500 3.80e-08 ***
## Shape2      -0.424488   0.399448  -1.063 0.287924
## Shape3       0.491117   0.440369   1.115 0.264747
## Shape4       1.380617   0.406291   3.398 0.000679 ***
## Margin2      1.775312   0.681475   2.605 0.009185 **
## Margin3      1.485908   0.442906   3.355 0.000794 ***
## Margin4      1.654457   0.368175   4.494 7.00e-06 ***
## Margin5      2.137907   0.454706   4.702 2.58e-06 ***
## Density.L   -1.657063   0.863658  -1.919 0.055028 .
## Density.Q   -0.338661   0.679045  -0.499 0.617968
## Density.C   -0.478684   0.441448  -1.084 0.278210
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 810.69  on 584  degrees of freedom
## Residual deviance: 501.39  on 573  degrees of freedom
## AIC: 525.39
##
## Number of Fisher Scoring iterations: 5
```

As we have learnt during the exploration dataset, the properties of the mammographic mass and the age of the patient will be used to predict whether the mass is malicious or benign.

4.3 Model Evaluation

After applying certain model to a dataset it is necessary to evaluate its performance. In order to dot that, a confusion matrix will be used, in which the prediction column that was produced during the model needs to be tested against the actual class values that already existed on the dataset. This will produce the following possible results:

- **True Positive (TP)**: correctly classified as the class of interest
- **True Negative(TN)**: Correctly classified as not the class of interest
- **False Positive(FP)**: Incorrectly classified as the class of interest
- **False Negative(FN)**: Incorrectly classified as not the class of interest

In order to perform a thorough analysis of the performance, it is advisable to predict the severity of the mammographic masses against the validation subset. In order to do that, a prediction is created by using the model and the testing dataset, with which the accuracy can be calculated.

```
# Test the model's performance
getModelAccuracy <- function (dataset, model) {
  # Generating a prediction
  probabilities <- predict(
    model,                                       # Used model
    newdata=subset(dataset,select=c(1,2,3,4)), # Test dataset
    type="response"                              # Type of prediction
  )

  # Store the predictions as 1 or 0
  predictions <- ifelse(probabilities>0.5, 1, 0)

  # Store accuracy
  accuracy = mean(predictions==validation$Severity)

  # Return Accuracy
  return( list("probabilities"=probabilities,
               "predictions"=predictions,
               "accuracy"=accuracy))
}

# Get the performance parameters
accuracyList = getModelAccuracy(validation,regressionModel)

# Output the accuracy
accuracyList$accuracy

## [1] 0.8207171
```

Finally, a ROC curve can be used to measure the performance of the model:

- The true positive rate is the sensitivity: the fraction of defaulters that are correctly identified, using a given threshold value.
- The false positive rate is 1-specificity: the fraction of non-defaulters that we classify incorrectly as defaulters, using that same threshold value.

```r
# ROC Model
pr <- prediction(accuracyList$probabilities, validation$Severity)
prf <- performance( pr, measure = "tpr", x.measure = "fpr")
plot(prf,frame=FALSE)
```

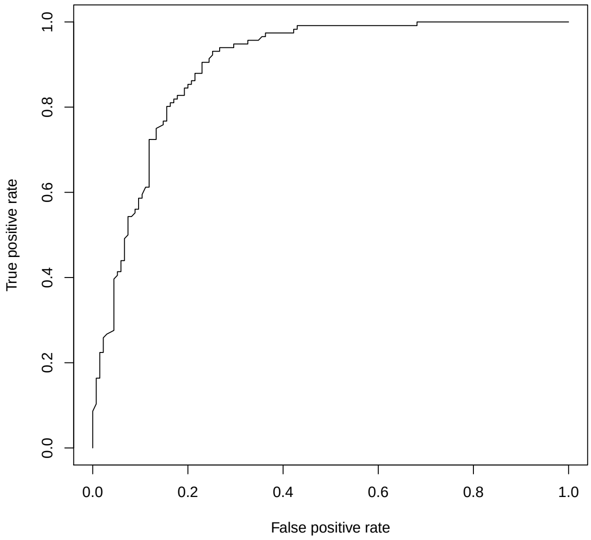

Figure 7: ROC curve of the regression model

5 Improving Performance

Even though the model has achieved a fairly decent performance, it can definitely be improved. Small tweaks in the dataset, model and validation might have a positive impact on the accuracy, which would lead to a more trustworthy prediction of a potential malignant lesion, and therefore saving lives. I've chosen the following methods to improve my dataset's model:

- **Filling missing values**: By dealing with missing values in a better way, the dataset size may increase, which leads to more accurate models.
- **Cross Validation**: When using training and testing datasets, there must be a tradeoff between maximizing training and validation dataset size, which can lead to inaccurate models. By using different validation methods, the performance of a given model may increase.
- **Different Models**: Using different modelling algorithms may vary the prediction accuracy for any given dataset, so it is a good idea to use more complex models.

```r
# Downloading necessary libraries
library(caret)
library(mice)
```

5.1 Filling Missing Values

Having a large enough dataset is a very relevant factor when it comes to perfecting a machine learning algorithm. The bigger the training data, the better the adjustments, and the more accurate the system may become. On any given dataset, there may be missing values, and while sometimes there is no way to recreate such missing values and entire rows must be dropped, an effort must be made in order to save as much information as possible.

Earlier in this coursework, it was assumed that all missing categorical values were supposed to be deleted, without trying to handle them. However, this was a mistake, and there are sevetral algorithms that deal with missing categorical values [3]. In this case, the package of choice has been MICE (Multivariate Imputation via Chained Equations), one of the commonly used package by R users. Creating multiple imputations as compared to a single imputation takes care of uncertainty in missing values. MICE assumes that the missing data are Missing at Random (MAR), which means that the probability that a value is missing depends only on observed value and can be predicted using them. It imputes data on a variable by variable basis by specifying an imputation model per variable.

```
# Getting our new dataframe ready
new_df <- df_full
new_df$Assessment <- NULL
new_df$Age <- fillMissings(new_df$Age)

fillMissing <- function(dataset, method) {
  # Using MICE to fill missing categorical values
  dataset <- mice(
    dataset,          # Using the newDF dataframe
    m=1,              # Generating only 1 dataframe
    maxit=20,         # Number of Iterations
    method=method     # Imputation Method
  )
  # Returning the filled dataframe
  return(complete(dataset,1))
}

# Fill missing categofical values using the polyreg method
new_df <- fillMissing(new_df, 'polyreg')

##
##  iter imp variable
##   1   1  Shape  Margin  Density
##   2   1  Shape  Margin  Density
##   3   1  Shape  Margin  Density
##   4   1  Shape  Margin  Density
##   5   1  Shape  Margin  Density
##   6   1  Shape  Margin  Density
##   7   1  Shape  Margin  Density
##   8   1  Shape  Margin  Density
##   9   1  Shape  Margin  Density
##   10   1  Shape  Margin  Density
##   11   1  Shape  Margin  Density
##   12   1  Shape  Margin  Density
##   13   1  Shape  Margin  Density
##   14   1  Shape  Margin  Density
##   15   1  Shape  Margin  Density
##   16   1  Shape  Margin  Density
##   17   1  Shape  Margin  Density
##   18   1  Shape  Margin  Density
##   19   1  Shape  Margin  Density
##   20   1  Shape  Margin  Density
```

```r
# Checking the number of missing values
sum(is.na(new_df))
```

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

Now that the missing values have been dealt with, there is a larger dataset to model, which may lead to more consistent results when testing the performance of a model.

5.2 Cross Validation

As mentioned earlier, dividing datasets into training and a validation subsets is an easy way to train a model, but it can lead to tradeoffs during the different stages of the model creation. There are many ways to validate models, but in this case the best approach is to use Cross Validation method. The k-fold cross validation method involves splitting the dataset into k-subsets. During the training phase each subset is held out while the model is trained on all other subsets. This process is completed until accuracy is determine for each instance in the dataset, and an overall accuracy estimate is provided.

```
# Create a partition, to reserve a validation subset for later testing
inTrain <- createDataPartition(
  y=new_df$Severity, # Have Severity as the classifying value
  p=0.85,            # Assign 85% of the data to training
  list=FALSE
)

# Generating the training subset
new_training <- new_df[inTrain,]
nrow(new_training)

## [1] 818

# Generating the validating subset
new_validating <- new_df [-inTrain,]
nrow(new_validating)

## [1] 143

# Make sure the dataset has been split correctly
print( testSize(new_df, new_training, new_validating) )

## [1] "Not properly split"

# Cross Validation
train_control<- trainControl(
  method="cv", # Use the Cross Validation Method
  number=10,   # Divide into 7 subsets
)
```

5.3 Different Models

On this particular dataset, having an accurate prediction can drastically influence a patient, since a false negative means that a mammographic mass could develop into a potentially untreated cancerous condition.

The previously used logistic regression algorithm may have given decent results, but it still might be a good idea to test a different model. In this case, the chosen predictive model will be ”Random Forest”, which consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model’s prediction [4].

```
# Random Forest
randomForestModel<- train(
  Severity˜.,                  # Severity is the value to predict
  data=new_training,           # Use the training data
  trControl=train_control,     # Use Cross Validation
  method="rf",                 # Use Random Forest
  family=binomial()            # The value to predict is binomial
)

# Print the Random Forest results
print(randomForestModel)
```

```
## Random Forest
##
## 818 samples
##   4 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 736, 736, 736, 736, 736, 736, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.7835061  0.5659470
##    6    0.7676524  0.5358972
##   11    0.7505793  0.4985203
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 2.
```

5.4 Conclusion

Now that we have improved different parts of our dataset and model, it is time to test its performance. Just like during the modelling stage, we will test the model against the validation subset, and verify the model accuracy.

```
# Make Predictions
predictions<- predict(
  randomForestModel,
  new_validating[,-ncol(new_validating)]
)

# Append predictions
new_validating<- cbind(
  new_validating,
  predictions
)

# Summarize results
results<- confusionMatrix(
  new_validating$predictions,
  new_validating$Severity
)

cat("Accuracy is: ", sum(diag(results$table))/nrow(new_validating))

## Accuracy is:  0.8251748
```

Interestingly enough, even though more refine methods were using to create models, there was no massive difference in the accuracy outcome, with small variations on accuracy. There is no clear evidence that Random Trees consistently outperforms Logistic Regression, having some cases in which Logistic Regression gets higher accuracy than Random Forest.

6 References

[1] Matthias Elter. *Mammographic Mass Data Set*. 2007. URL: https://archive.ics.uci.edu/ml/datasets/MammographicMass.

[2] *The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process*. 2007. URL: https://www.researchgate.net/publication/5776853_The_prediction_of_breast_cancer_biopsy_outcomes_using_two_CAD_approaches_that_both_emphasize_an_intelligible_decision_process.

[3] Harshitha Mekala. *Dealing with Missing Data using R*. 2019. URL: https://medium.com/coinmonks/dealing-with-missing-data-using-r-3ae428da2d17.

[4] Tony Yiu. *Understanding Random Forest*. 2019. URL: https://towardsdatascience.com/understanding-random-forest-58381e0602d2.