log

Tim Swarts

06/10/2021

Research questions

Which attributes best predict breast cancer malignancy?

Is it possible to create a machine learning model that can predict breast cancer malignancy with an accuracy of 80% or higher?

Load libraries and read data

```
library(tibble)
```

```
## Warning: package 'tibble' was built under R version 4.0.5
```

```
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 4.0.5
```

```
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.0.5
```

```
library(scales)
```

```
## Warning: package 'scales' was built under R version 4.0.5
```

This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg.

The data provided does have clear column names, but a description of each attribute has been given in the *breast-cancer-wisconsin.names* file. This allows us to assign representative names to every column. The code to do this is shown below.

```r
# Read data
data <- read.csv("breast-cancer-wisconsin.data", na.strings = '?')
# Convert to tibble for readability later on
data <- as_tibble(data)
# Set column names
colnames(data) <- c("id", "Clump_Thickness", "Cell_Size_Uniformity", "Cell_Shape_Uniformity",
                    "Bland_Chromatin", "Normal_Nucleoli", "Mitoses", "Class")
```

Basic look at the data

```r
# Look at head of data
data
```

```
## # A tibble: 698 x 11
##         id Clump_Thickness Cell_Size_Uniform~ Cell_Shape_Unifo~ Marginal_Adhesi~
##      <int>           <int>              <int>             <int>            <int>
##  1 1002945               5                  4                 4                5
##  2 1015425               3                  1                 1                1
##  3 1016277               6                  8                 8                1
##  4 1017023               4                  1                 1                3
##  5 1017122               8                 10                10                8
##  6 1018099               1                  1                 1                1
##  7 1018561               2                  1                 2                1
##  8 1033078               2                  1                 1                1
##  9 1033078               4                  2                 1                1
## 10 1035283               1                  1                 1                1
## # ... with 688 more rows, and 6 more variables:
## #   Single_Epithelial_Cell_Size <int>, Bare_Nuclei <int>,
## #   Bland_Chromatin <int>, Normal_Nucleoli <int>, Mitoses <int>, Class <int>
```

```r
# Summarize data
summary(data)
```

```
##        id           Clump_Thickness  Cell_Size_Uniformity Cell_Shape_Uniformity
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000       Min.   : 1.000
##  1st Qu.:  870258   1st Qu.: 2.000   1st Qu.: 1.000       1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000       Median : 1.000
##  Mean   : 1071807   Mean   : 4.417   Mean   : 3.138       Mean   : 3.211
##  3rd Qu.: 1238354   3rd Qu.: 6.000   3rd Qu.: 5.000       3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000       Max.   :10.000
##
##  Marginal_Adhesion Single_Epithelial_Cell_Size  Bare_Nuclei
##  Min.   : 1.000    Min.   : 1.000              Min.   : 1.000
##  1st Qu.: 1.000    1st Qu.: 2.000              1st Qu.: 1.000
##  Median : 1.000    Median : 2.000              Median : 1.000
##  Mean   : 2.809    Mean   : 3.218              Mean   : 3.548
##  3rd Qu.: 4.000    3rd Qu.: 4.000              3rd Qu.: 6.000
##  Max.   :10.000    Max.   :10.000              Max.   :10.000
##                                                NA's   :16
##  Bland_Chromatin  Normal_Nucleoli    Mitoses          Class
##  Min.   : 1.000   Min.   : 1.00   Min.   : 1.00   Min.   :2.000
```

```
##  1st Qu.: 2.000   1st Qu.: 1.00   1st Qu.: 1.00   1st Qu.:2.000
##  Median : 3.000   Median : 1.00   Median : 1.00   Median :2.000
##  Mean   : 3.438   Mean   : 2.87   Mean   : 1.59   Mean   :2.691
##  3rd Qu.: 5.000   3rd Qu.: 4.00   3rd Qu.: 1.00   3rd Qu.:4.000
##  Max.   :10.000   Max.   :10.00   Max.   :10.00   Max.   :4.000
##
```

All data has been normalized to fit a grading systems that grades the severity of each attribute. A detailed description of what each grade value means for each attribute can be found in the Breast Cancer Diagnosis Web User Interface.

Correlation

To get a sense of which attributes are best suited to predict the class, it is helpful to take a look at the correlation. Below is a piece of code that prints the column names of the columns with a correlation of 0.8 or higher to the Class attribute.

```
# Get correlation
data.cor <- cor(data[,-1], use = "complete.obs")
data.cor <- as.data.frame(data.cor)
# Keep all correlations that are higher than 0.8
cor.names <- names(data.cor)[which(data.cor$Class > 0.8)]
# Print
cat(cor.names)
```

```
## Cell_Size_Uniformity Cell_Shape_Uniformity Bare_Nuclei Class
```

This gives us just three attributes of interest besides the Class attribute which obviously fully correlates to itself. This is good, because this means the model we will train later will likely only need these three attributes to accurately predict the Malignancy of a cancer clump. A simple model is always better, because less data needs to be acquired in order to use it. However, to make sure we do not miss any value information, it is useful to make a heat map of the correlation table. The heat map and corresponding code is shown below.

```
# Make correlation matrix
cormat <- cor(na.omit(data[, -1]))
# Melt to correct format
melted.cormat <- melt(cormat)
# head(melted.cormat) # <- uncomment this to look at the new matrix

# Make heatmap with ggplot
ggplot(data = melted.cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() + # geom_tile makes the heatmap tiles
  scale_fill_gradient(low = rgb(0, 0, 0),
                      high = rgb(0, 1, 1),
                      guide = "colorbar") + # sets color the gradient for tiles
  ylab('') +
  xlab('') + # the x and y axes have no useful names, and are thus left blank
  labs(caption =
      "Figure 1 ~ Correlation Heatmap of breast cancer data set.
      The lighter the color, the better the correlation between the two columns.") + # adds caption
  theme_minimal() + # set theme to minimal
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1), # adjust x labels to be diagonal
        plot.caption = element_text(hjust = 0.7)) # adjust caption position
```

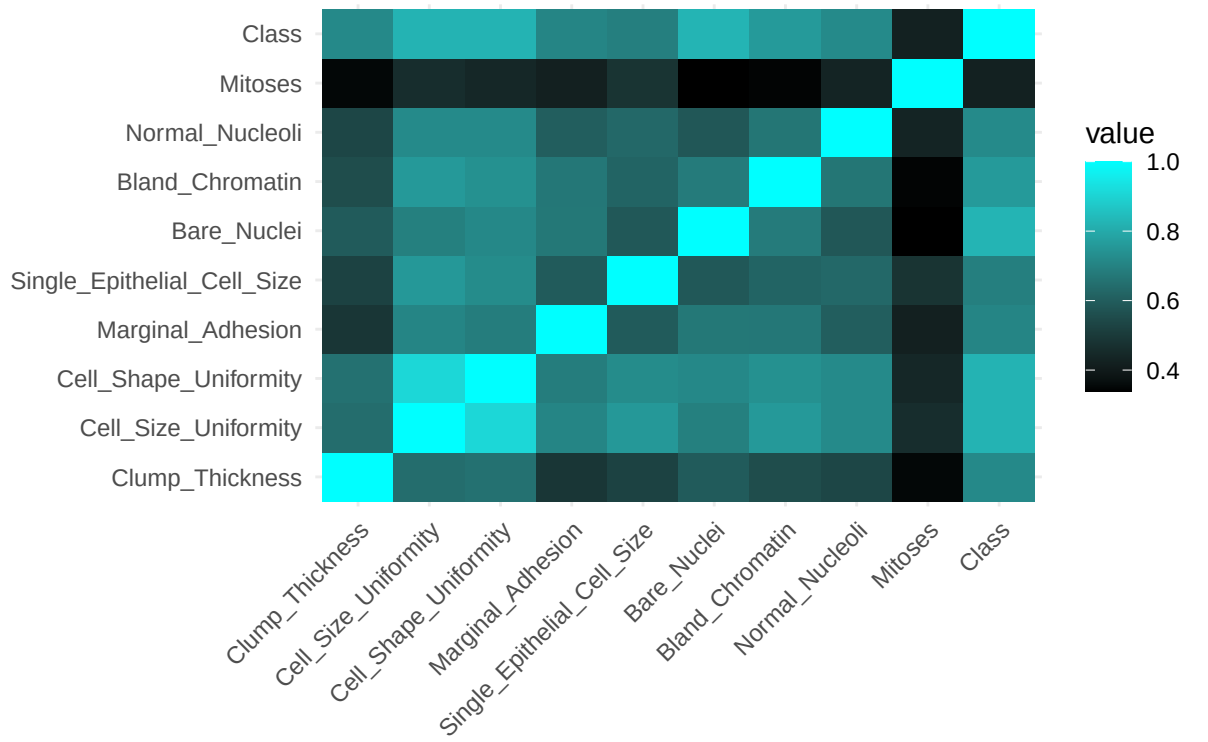


Figure 1 ~ Correlation Heatmap of breast cancer data set.
The lighter the color, the better the correlation between the two columns.

Looking at 'Class' column of the heat map in figure 1, the Bland_Chromatin correlation is the lightest color visible besides the three attributes found earlier (Cell_Size_Uniformity, Cell_Shape_Uniformity Bare_Nuclei). The following code takes a look at the exact correlation of this attribute.

```
# Print correlation of Bland_Chromatin to Class
cormat["Bland_Chromatin", "Class"]
```

```
## [1] 0.7583497
```

The correlation is roughly 0.76, which is close to the 0.8 threshold set earlier. Bland_Chromatin might be another attribute of interest.

Another thing that stands out in figure 1 is that Cell_Shape_Uniformity and Cell_Size_Uniformity show a high correlation as depicted by the light 4x4 square at the bottom left of the figure. This will be discussed further later on.

Class Distribution

Change Class to be factor

The code shown below substitutes the previously numeric data in the Class column of the data set to a factor with two labels: B for benign and M for malignant. This makes following visualizations easier and is also needed when eventually classifying the data with our model.

```r
# Change class values to B for benign and M for Malignant instead of 2 and 4
data$Class[data$Class == 2] <- 'B'
data$Class[data$Class == 4] <- 'M'
data["Class"]
```

```
## # A tibble: 698 x 1
##    Class
##    <chr>
##  1 B
##  2 B
##  3 B
##  4 B
##  5 M
##  6 B
##  7 B
##  8 B
##  9 B
## 10 B
## # ... with 688 more rows
```

Pie Chart

To visualize the distribution of instances over the two classes a pie chart is made.

```r
# Get class counts
m.count <- length(data$Class[data$Class == 'M'])
b.count <- length(data$Class[data$Class == 'B'])
# Combine in data frame
count.data <- data.frame(Class = c('Malignant', 'Benign'), Value = c(m.count, b.count))

count.data %>%
arrange(desc(Value)) %>%
mutate(prop = percent(Value / sum(Value),  accuracy = 0.1)) -> count.data

# Make pie chart
ggplot(count.data, aes(x = "", y = Value, fill = Class)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) +
  geom_label_repel(aes(label = prop), size=5, show.legend = F, nudge_x = -1, segment.colour= NA) +
  labs(caption="Figure 2 ~ A pie chart of the distrubtion of the classes in the dataset.") +
  theme_void() +
  theme(plot.caption = element_text(hjust = 1))
```

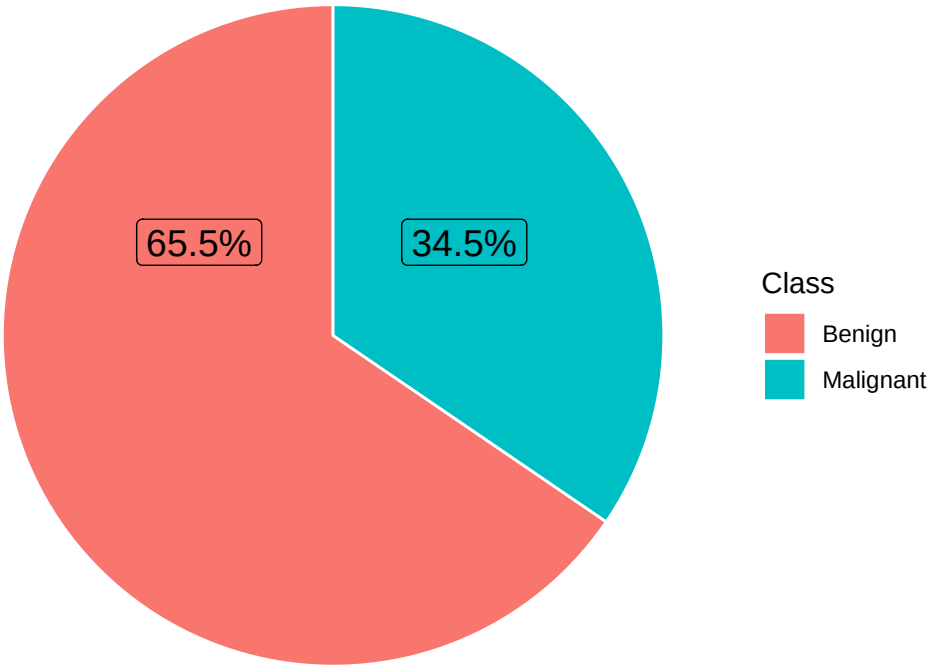


Figure 2 ~ A pie chart of the distrubtion of the classes in the dataset.

```
# Calculate exact percentage
m.count / (m.count + b.count) * 100
```

```
## [1] 34.52722
```

Figure 2 shows that roughly two thirds of the data consists of Benign instances. When calculated, we find that only 34.5% of the instances are malignant. This should be kept in mind when testing the model. A ZeroR algorithm would for instance already classify 65.5% of the instances correctly by pure chance. Although this seems accurate, this would obviously not be a good model, since ZeroR ignores all predictors and simply picks the majority category.

Further Visualization

Correlation Cell Size, Cell Shape

The following graph shows the relation between Cell Size and Cell Shape as discussed earlier when looking at figure 1.

```
# Make plot
# coef(linear.model <- lm(Cell_Shape_Uniformity ~ Cell_Size_Uniformity, data = data))

ggplot(data = data, mapping = aes(x = Cell_Size_Uniformity, y = Cell_Shape_Uniformity)) +
  # Add points
  geom_jitter(aes(col = Class), size = 0.7, alpha = 0.5, width = 0.36, height = 0.36) +
```

```
  # Add linear regression line
  geom_smooth(method = "lm", se = T, col='darkgrey') +
  # Set axis labels
  xlab("Cell Size Uniformity (grade 1-10)") +
  ylab("Cell Shape Uniformity (grade 1-10)") +
  # Scale x and y axis to represent the 1 - 10 grading
  ylim(0, 10) +
  scale_y_continuous(breaks = seq(0, 10, by = 2)) +
  xlim(0, 10) +
  scale_x_continuous(breaks = seq(0, 10, by = 2)) +
  labs(caption="Figure 3 ~ Cell Shape as a function of Cell Size.
       The color of the points show their class, red being benign, blue being malignant.") +
  # Make theme minimal
  theme_minimal() +
  theme(plot.caption = element_text(hjust = 0.5))
```

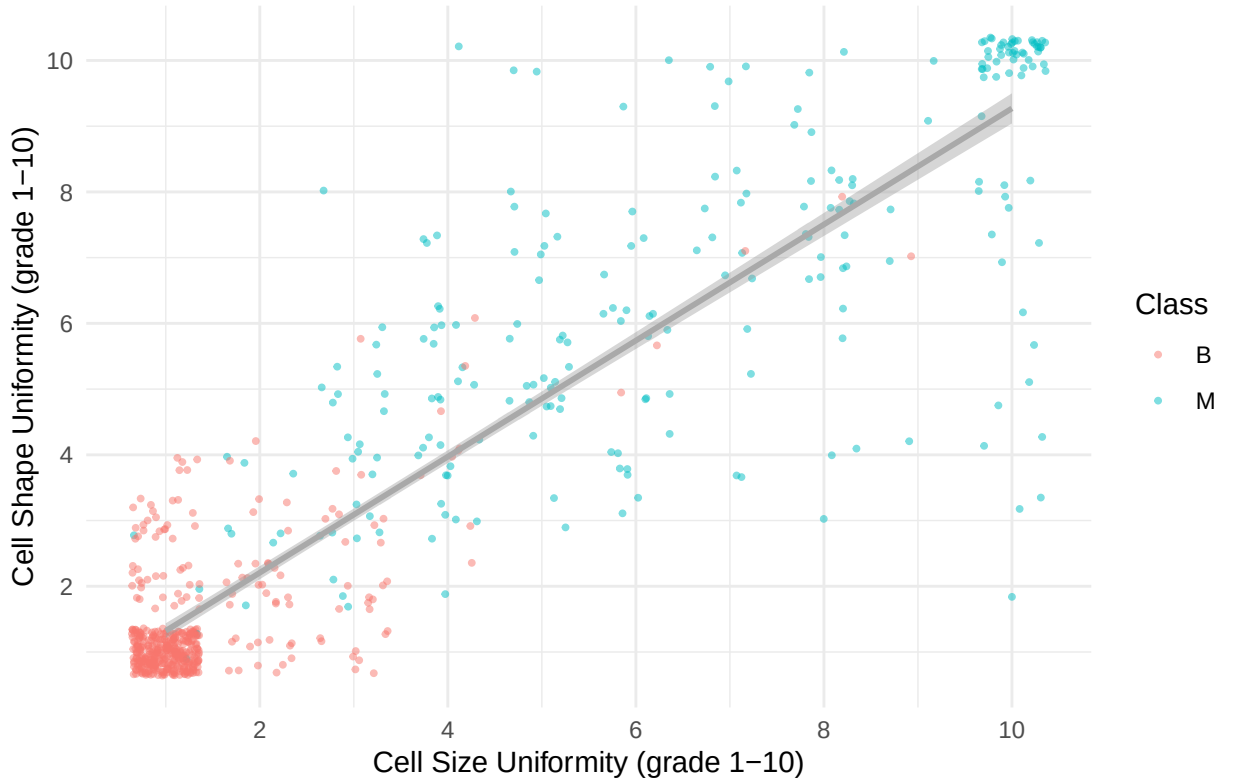


Figure 3 ~ Cell Shape as a function of Cell Size.
The color of the points show their class, red being benign, blue being malignant.

Figure 3 shows the correlation between the attributes Cell_Shape_Uniformity and Cell_Size_Uniformity. When looking at the spread of the two colors in this graph, the benign instances seem to form a group in the bottom left, while the malignant cases are spread across the middle and the top right. This division suggests these attributes would be good predictors for breast cancer malignancy. A simple perceptron could divide the two classes with some accuracy based on these two attributes alone.

How does cell shape relate to malignancy

Below we see how cell shape relates to malignancy, by plotting points with jitter to show how many points represent each grade.

```
ggplot(data = data, mapping = aes(x = Class, y = Cell_Shape_Uniformity)) +
  geom_jitter(width = 0.05, height = 0.2, col = "forestgreen", alpha = 0.5) +
  ylim(0, 10) +
  xlab("Class (B or M)") +
  ylab("Cell Shape Uniformity (grade 1-10)") +
  scale_y_continuous(breaks = seq(0, 10, by = 2)) +
  labs(caption="Figure 4 ~ Cell Shape as a function of Class. Jitter and transparancy is added to show t
  theme_minimal()
```

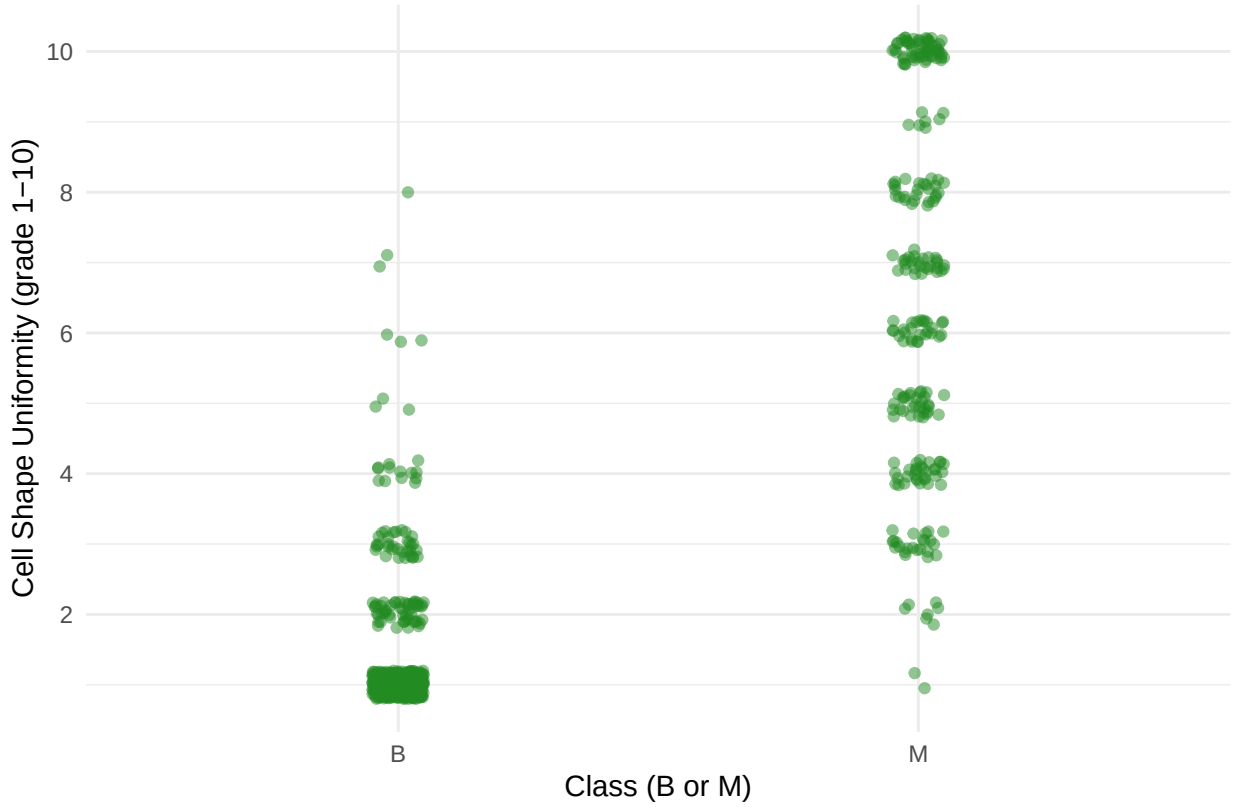


Figure 4 ~ Cell Shape as a function of Class. Jitter and transparancy is added to show the denisity of points at each grade.

Looking at figure 4, we can conclude that most of benign cases will get a cell shape grade of 2 or lower. The malignant cases, however, appear more spread out. This would cause inaccuracy when classifying with this attribute only.

Set data to csv

```
to.csv <- na.omit(data)
write.csv(to.csv, "breast_cancer_malignancy_dataset.csv", row.names = FALSE)
```

Trying machine learning algorithms

When testing different machine learning algorithms it is import a decision is made about sensitivity versus specificity. A very sensitive model may catch most if not all malignant cases, but might have a higher false

positive rate. A very specific model will eliminate false positives, but may miss more malignant cases. In this case I will opt for sensitivity over specificity, because the consequences of a missed malignant case are higher than that of a falsely diagnosed benign case.

In order to achieve this, we can add a cost matrix to the uses machine learning algorithm that ways false negatives higher than Before we start optimizing for this however, let us first look which algorithms perform best on their own, before we add a cost matrix.

In most tested models shown below, only attributes with a correlation of 0.75 or higher to the Class column, as found in the exploratory data analysis above (see figure 1), where used as predictors unless stated otherwise. All models where trained and tested in Weka

Cost Insensitive Classifiers

J48

This model was trained using only the following attributes as predictors:

- Cell_Size_Uniformity
- Cell_Shape_Uniformity
- Bare_Nuclei
- Bland_Chromatin

Accuracy: 9 correctly classified instances
Recall for M: 0.941
Precision for M: 0.930

Table 1: Confusion Matrix J48

classified as ->	B	M
B	426	17
M	14	225

Random Forest

This model was trained using only the following attributes as predictors:

- Cell_Size_Uniformity
- Cell_Shape_Uniformity
- Bare_Nuclei
- Bland_Chromatin

Accuracy: 96.04% correctly classified instances
Recall for M: 0.937
Precision for M: 0.949

Table 2: Confusion Matrix Random Forest

classified as ->	B	M
B	431	12
M	15	224

Logistic regression

This model was trained using only the following attributes as predictors:

- Cell_Size_Uniformity
- Cell_Shape_Uniformity
- Bare_Nuclei
- Bland_Chromatin

Accuracy: 95.89% correctly classified instances
Recall for M: 0.933
Precision for M: 0.949

Table 3: Confusion Matrix Logistic

classified as ->	B	M
B	431	12
M	16	223

Naive Bayes

This model was trained using only the following attributes as predictors:

- Cell_Size_Uniformity
- Cell_Shape_Uniformity
- Bare_Nuclei
- Bland_Chromatin

Accuracy: 95.31% correctly classified instances
Recall for M: 0.950
Precision for M: 0.919

Table 4: Confusion Matrix Naive Bayes

classified as ->	B	M
B	423	20
M	12	227

Logistic with more attributes

This model was trained using only the following attributes as predictors:

- Clump_Thickness
- Cell_Size_Uniformity
- Cell_Shape_Uniformity
- Marginal_Adhesion
- Single_Epithelial_Cell_Size
- Bare_Nuclei
- Bland_Chromatin
- Normal_Nucleoli

Accuracy: 96.77% correctly classified instances
Recall for M: 0.954
Precision for M: 0.954

Table 5: Confusion Matrix Logistic with more attributes

classified as ->	B	M
B	432	11
M	11	228

Cost Sensitive Classifiers

Logistic Cost Sensitve Classifier

This model was trained using only the following attributes as predictors:

- Cell_Size_Uniformity
- Cell_Shape_Uniformity
- Bare_Nuclei
- Bland_Chromatin

Accuracy: 96.77% correctly classified instances
Recall for M: 0.975
Precision for M: 0.936

Table 6: Confusion Matrix Logistic Cost Sensitive Classifier

classified as ->	B	M
B	427	16
M	6	223

ROC

```
roc_data <- read.table("ROC.arff",
                   sep = ",",
                   comment.char = "@")
names(roc_data) <- c("Instance_number",
                  "True_Positives",
                  "False_Negatives",
                  "False_Positives",
                  "True_Negatives",
                  "False_Positive_Rate",
                  "True_Positive_Rate",
                  "Precision",
                  "Recall",
                  "Fallout",
                  "FMeasure",
                  "Sample_Size",
                  "Lift",
                  "Threshold")



head(roc_data)
```

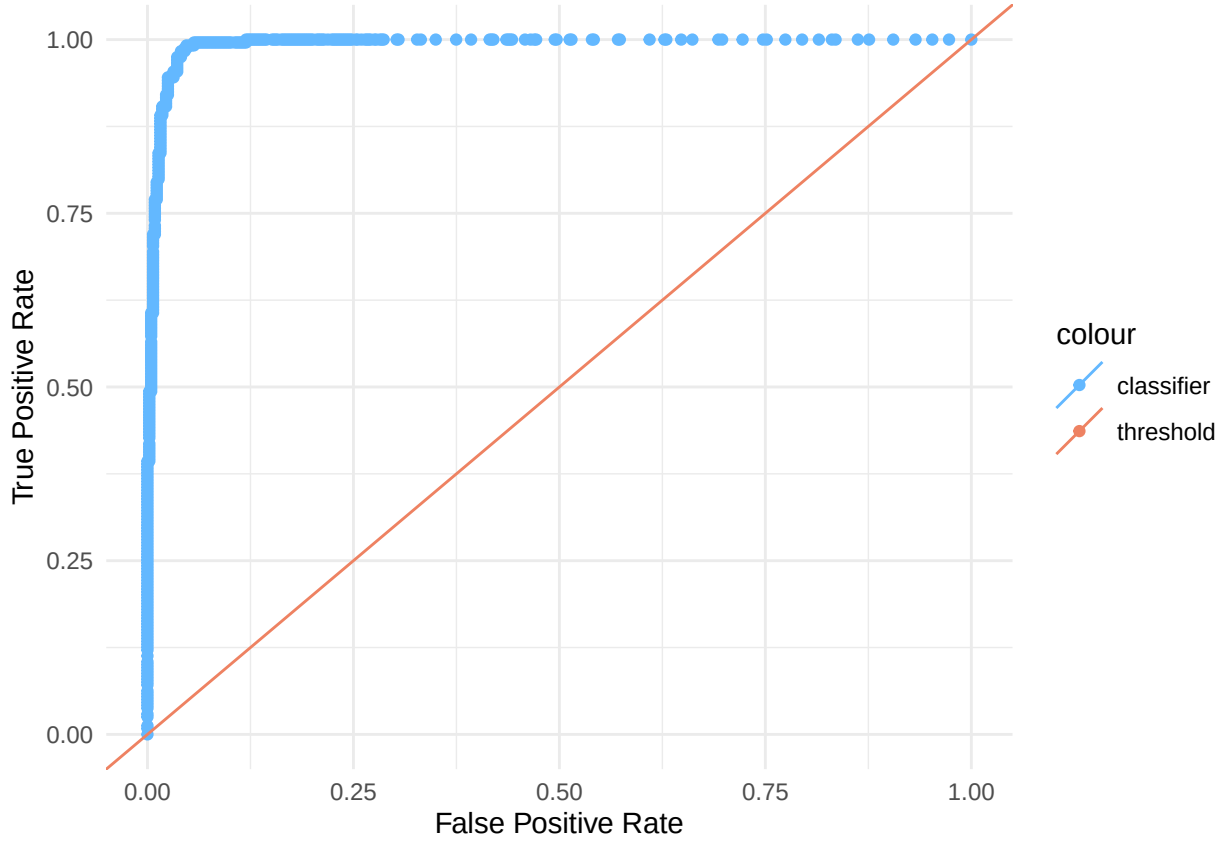
```
##   Instance_number True_Positives False_Negatives False_Positives True_Negatives
## 1               0            239               0             443              0
## 2               1            239               0             431             12
## 3               2            239               0             422             21
## 4               3            239               0             413             30
## 5               4            239               0             401             42
## 6               5            239               0             388             55
##   False_Positive_Rate True_Positive_Rate Precision Recall  Fallout FMeasure
## 1            1.000000                  1   0.35044      1  0.64956 0.519001
## 2            0.972912                  1  0.356716      1 0.643284 0.525853
## 3            0.952596                  1  0.361573      1 0.638427 0.531111
## 4            0.932280                  1  0.366564      1 0.633436 0.536476
## 5            0.905192                  1  0.373437      1 0.626563   0.5438
## 6            0.875847                  1   0.38118      1  0.61882 0.551963
##   Sample_Size     Lift Threshold
## 1    1.000000        1  0.005601
## 2    0.982405  1.01791  0.006906
## 3    0.969208  1.03177  0.007832
## 4    0.956012 1.046012  0.008255
## 5    0.938416 1.065625  0.008821
## 6    0.919355 1.087719  0.008936
```

```
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.0.5
```

```
colors <- c(classifier = "steelblue1", threshold = "salmon2")
plt <- ggplot(data = roc_data,
       mapping = aes(x = False_Positive_Rate, y = True_Positive_Rate)) +
```

```
    geom_point(mapping = aes(color = "classifier")) +
    geom_abline(aes(color = "threshold",
                    slope = 1,
                    intercept = 0)) +
    scale_color_manual(values = colors) +
    xlab("False Positive Rate") +
    ylab("True Positive Rate") +
    #theme_minimal() +
    theme_pubr() +
    theme(legend.title = element_blank()) +
    theme_minimal()
print(plt)
```



We are satisfied with this model, and will now create a data set suited for testing classification, by removing all unnecessary columns and writing to an arff and csv file.

```
library(farff)
```

```
## Warning: package 'farff' was built under R version 4.0.5
```

```
# make copy of  current data set
classify.data <- to.csv
# remove unwanted columns
classify.data[, c(1, 2, 5, 9, 10)] <- NULL
# make class na (because it needs to be classified by the model)
```

```
classify.data$Class <- NA
# write to arff file
type=NULL
writeARFF(classify.data, path = "./breastcancer_model_input.arff", overwrite=TRUE)
```