Breast Cancer Wisconsin

Rose Hazenberg

2021-11-19

Contents


```
## Libraries
library(utils)
library(kableExtra)
library(dplyr)
library(tidyr)
library(ggplot2)
library(stats)
library(pheatmap)
library(cowplot)
library(ggbiplot)
library(png)
library(ggpubr)
```

1 Introduction

Despite a great deal of public awareness and scientific research, breast cancer continues to be the most common cancer and the second largest cause of cancer deaths among women. Approximately 12% of U.S. women will be diagnosed with breast cancer, and 3.5% will die of it. The annual mortality rate of approximately 28 deaths per 100,000 women has remained nearly constant over the past 20 years. A breast cancer victim's chances for long-term survival are improved by early detection of the disease, and early detection is in turn enhanced by accurate diagnosis.[1]

There are two significant applications of linear programming in the field of breast cancer research, one in diagnosis and one in prognosis. Both applications, in clinical practice, depend on the analysis of cellular images which is accomplished with a graphical computer program called **Xcyt**, written by one of the authors.[1]

First, a sample of fluid is taken from the patient's breast. The procedure involves using a small-gauge needle to take fluid, known as Fine Needle Aspiration (FNA), directly from a breast lump or mass. This allows an accurate diagnosis without the need for a surgical biopsy. From the FNA an image is generated which is then used for the program Xcyt.[1] Xcyt uses a curve-fitting program to determine the exact boundaries of the cancer cell based on a digital scan. Thus a digitized image is used to assess whether a lump in a breast could be malignant (cancerous) or benign (non-cancerous).[2]

Machine learning plays an important role in the classification of breast cancer. Many diagnosis processes have been discussed besides the images. These types of diagnosis images are used for classification using machine learning.[3] Machine learning is a form of Artificial Intelligence (AI) aimed building systems that can learn from the processed data or use data to perform better.[4]

The research question is: Is it possible to reliably predict the cancer stage based on the uniformity of cell size/shape using machine learning?

1.1 Data description

The data obtain information about breast cancer in Wisconsin. This was provided by Dr. William H. Wolberg of the University of Wisconsin Hospitals.

1.2 Data set information

The samples are clinical cases of Dr. Wolberg which are periodically classified and are ordered in a chronological grouping. The grouping is as followed:

Group 1: 367 instances (January 1989)
Group 2: 70 instances (October 1989)
Group 3: 31 instances (February 1990)
Group 4: 17 instances (April 1990)
Group 5: 48 instances (August 1990)
Group 6: 49 instances (Updated January 1991)
Group 7: 31 instances (June 1991)
Group 8: 86 instances (November 1991)

Total: 699 points (as of the donated database on 15 July 1992)

Table 1 shows the codebook of the data which provides an overview of the data frame and the variables.

```
## Load the codebook
codebook <- read.csv(file = "codebook.txt", sep = ";")
kable(codebook, caption = "Codebook with an overview of the data.")
```

Table 1: Codebook with an overview of the data.

Column.Name	Full.Name	Data.Type	Value.Range
ID	Sample code number	int	NA
clump_thickness	Clump Thickness	int	1 - 10
uniformity_of_cell_size	Uniformity of Cell Size	int	1 - 10
uniformity_of_cell_shape	Uniformity of Cell Shape	int	1 - 10
marginal_adhesion	Marginal Adhesion	int	1 - 10
single_epithelial_cell_size	Single Epithelial Cell Size	int	1 - 10
bare_nuclei	Bare Nuclei	int	1 - 10
bland_chromatin	Bland Chromatin	int	1 - 10
normal_nucleoli	Normal Nucleoli	int	1 - 10
mitoses	Mitoses	int	1 - 10
class	Class	factor	Benign, Malignant

2 Exploratory Data Analysis (EDA) of breast cancer

Before the data is visualized is the data loaded. After the following code, the data is available to use.

```
## Define the data file
datafile <- "data/breast-cancer-wisconsin.data"

breastcancer <- read.table(datafile, sep = ",", header = FALSE,
                           na.strings = "?")

## Change the column names
colnames(breastcancer) <- c("ID", "clump_thickness", "uniformity_of_cell_size",
                            "uniformity_of_cell_shape", "marginal_adhesion",
                            "single_epithelial_cell_size", "bare_nuclei",
                            "bland_chromatin", "normal_nucleoli", "mitoses",
                            "class")

## Create new variables for the column class
breastcancer <- breastcancer %>%
    mutate(class = factor(class, labels = c("Benign", "Malignant"),
                          levels = c(2, 4)))

## Inspect the data
str(breastcancer)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ ID                         : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561
##  $ clump_thickness            : int  5 5 3 6 4 8 1 2 2 4 ...
##  $ uniformity_of_cell_size    : int  1 4 1 8 1 10 1 1 1 2 ...
##  $ uniformity_of_cell_shape   : int  1 4 1 8 1 10 1 2 1 1 ...
##  $ marginal_adhesion          : int  1 5 1 1 3 8 1 1 1 1 ...
##  $ single_epithelial_cell_size: int  2 7 2 3 2 7 2 2 2 2 ...
##  $ bare_nuclei                : int  1 10 2 4 1 10 10 1 1 1 ...
##  $ bland_chromatin            : int  3 3 3 3 3 9 3 3 1 2 ...
##  $ normal_nucleoli            : int  1 2 1 7 1 7 1 1 1 1 ...
##  $ mitoses                    : int  1 1 1 1 1 1 1 1 5 1 ...
##  $ class                      : Factor w/ 2 levels "Benign","Malignant": 1 1 1 1 1 2 1 1 1 1 ...
```

```
## Show the first 6 rows of the breast cancer data
head(breastcancer)
```

```
##        id clump_thickness uniformity_of_cell_size uniformity_of_cell_shape
## 1 1000025               5                       1                       1
## 2 1002945               5                       4                       4
## 3 1015425               3                       1                       1
## 4 1016277               6                       8                       8
## 5 1017023               4                       1                       1
## 6 1017122               8                      10                      10
##   marginal_adhesion single_epithelial_cell_size bare_nuclei bland_chromatin
## 1                 1                           2           1               3
## 2                 5                           7          10               3
## 3                 1                           2           2               3
## 4                 1                           3           4               3
## 5                 3                           2           1               3
## 6                 8                           7          10               9
##   normal_nucleoli mitoses     class
## 1               1       1    Benign
## 2               2       1    Benign
## 3               1       1    Benign
## 4               7       1    Benign
## 5               1       1    Benign
## 6               7       1 Malignant
```

```
## Give an summary of the data
summary(breastcancer)
```

```
##        id           clump_thickness  uniformity_of_cell_size
##  Min.   :   61634   Min.   : 1.000   Min.   : 1.000
##  1st Qu.:  870688   1st Qu.: 2.000   1st Qu.: 1.000
##  Median : 1171710   Median : 4.000   Median : 1.000
##  Mean   : 1071704   Mean   : 4.418   Mean   : 3.134
##  3rd Qu.: 1238298   3rd Qu.: 6.000   3rd Qu.: 5.000
##  Max.   :13454352   Max.   :10.000   Max.   :10.000
##
##  uniformity_of_cell_shape marginal_adhesion single_epithelial_cell_size
##  Min.   : 1.000           Min.   : 1.000    Min.   : 1.000
##  1st Qu.: 1.000           1st Qu.: 1.000    1st Qu.: 2.000
##  Median : 1.000           Median : 1.000    Median : 2.000
##  Mean   : 3.207           Mean   : 2.807    Mean   : 3.216
##  3rd Qu.: 5.000           3rd Qu.: 4.000    3rd Qu.: 4.000
##  Max.   :10.000           Max.   :10.000    Max.   :10.000
##
##   bare_nuclei     bland_chromatin  normal_nucleoli     mitoses
##  Min.   : 1.000   Min.   : 1.000   Min.   : 1.000   Min.   : 1.000
##  1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.: 1.000   1st Qu.: 1.000
```

```
##  Median : 1.000   Median : 3.000   Median : 1.000   Median : 1.000
##  Mean   : 3.545   Mean   : 3.438   Mean   : 2.867   Mean   : 1.589
##  3rd Qu.: 6.000   3rd Qu.: 5.000   3rd Qu.: 4.000   3rd Qu.: 1.000
##  Max.   :10.000   Max.   :10.000   Max.   :10.000   Max.   :10.000
##  NA's   :16
##        class
##  Benign   :458
##  Malignant:241
##
##
##
##
##
```

The outcome of the summary shows that every column has the same minimum and maximum except the id and the class but these columns aren't properties. The 1st Qu, median, mean and 3rd Qu are different for every column but they all lie close to each other. At the minimum and maximum, you can see that the data only consist of the numbers 1 to 10.

Table 2: The number of NA's per column

	Number of NA's
ID	0
clump_thickness	0
uniformity_of_cell_size	0
uniformity_of_cell_shape	0
marginal_adhesion	0
single_epithelial_cell_size	0
bare_nuclei	16
bland_chromatin	0
normal_nucleoli	0
mitoses	0
class	0

Table 3: Head of the 'lengthens' data

ID	class	property	value
1000025	Benign	clump_thickness	5
1000025	Benign	uniformity_of_cell_size	1
1000025	Benign	uniformity_of_cell_shape	1
1000025	Benign	marginal_adhesion	1
1000025	Benign	single_epithelial_cell_size	2
1000025	Benign	bare_nuclei	1

2.1 Missing data

To check if there is missing data, a table is generated with the number of NA's per column with the following code.

```
kable(colSums(is.na(breastcancer)), col.names = "Number of NA's",
      caption = "The number of NA's per column")
```

Table 2 shows the number of NA's in the dataset of breast cancer. It shows that there are 16 NA's in de column bare nuclei, the other columns have 0 NA's. For the research question, there will be looked at the uniformity of cell size/shape and not at the column bare nuclei. Since there are only 16 NA's and in the column that isn't relevant so it doesn't affect the research question. To deal with these missing values column bare nuclei can be dropped. For the visualizations now it doesn't impact the data so these values aren't removed.

2.2 Displaying the variation of the breast cancer dataset

Here is the variation within the data and the distribution of the data examined.

```
## Tidying the data by using pivot_longer
breast_long <- pivot_longer(data = breastcancer,
                            cols = -c("ID", "class"),
                            names_to = "property",
                            values_to = "value")

## Show the first 6 rows of the breast cancer dataset after pivot_longer
kable(head(breast_long), digits = 6, caption = "Head of the 'lengthens' data")
```

In table 3 the number of rows is increased and the number of columns is decreased compared to the original

loaded data. With the new form of the data it can be visualized easily per property.

One of the visualization to examine the variation and the distribution is a histogram.

```
## Plot the histogram with geom_histogram
ggplot(data = breast_long, mapping = aes(x = property, y = value)) +
  geom_histogram(stat = 'identity', aes(color = class, fill = class)) +
  scale_fill_manual(values = c("steelblue", "plum1"),
                    aesthetics = c("color", "fill")) +
  labs(x="Property (n/a)", y="Total count of values (n/a)",
       color = "Class", fill = "Class") +
  coord_flip() +
  ## Get the names from the codebook as labels
  scale_x_discrete(labels = sort(codebook$Full.Name[2:10])) +
  theme_classic()
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```

```
## Warning: Removed 16 rows containing missing values (position_stack).
```

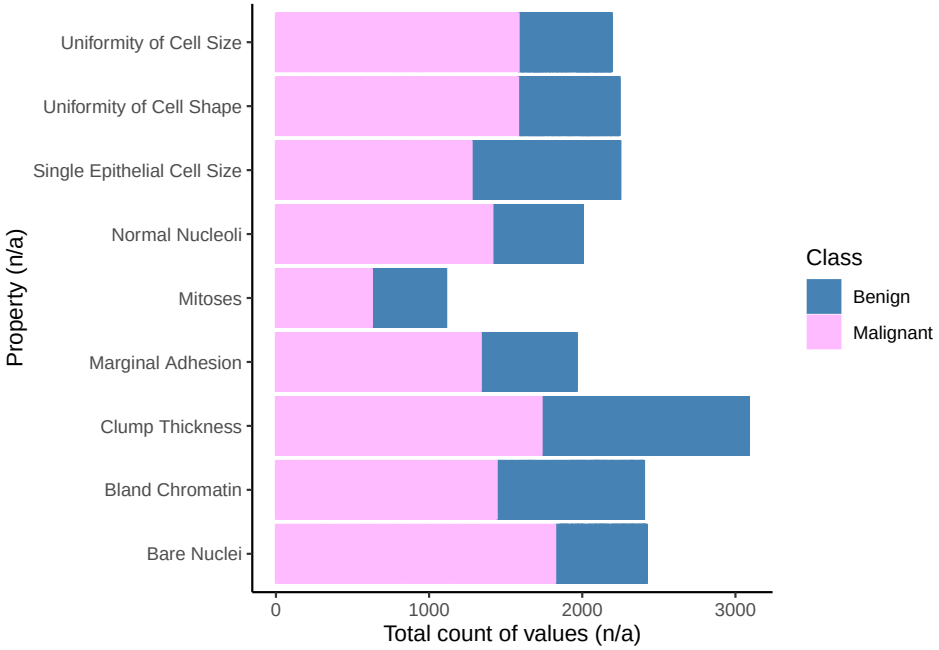


Figure 1: Histogram with the total count of the properties per stage of breast cancer.

Figure 1 shows the total count of the properties per stage in a histogram. The plot shows the properties for the class benign and malignant. The variance between the total of the properties per class lie close to each other except for the properties mitoses, here is the total the lowest, and clump thickness, here is the total the highest. Figure 1 shows that the distribution of the classes differ from each other because the class benign has a smaller bar than the class malignant and so a lower count for benign. The data consist of the number 1 to 10 and this says something about how benign or malignant a property is for the id of the person. So with this figure 1 you can't say which class belongs to the person only what the total count of a property is for a class. The class benign has mainly values of 1 so this explains why the bar of the class benign per property is smaller. And the class malignant has mainly values of 10 so this bar is longer but the class can also have values of 1. These values can't say which attribute is more informative in the model process because of the difference in values of the properties. But it can say that the class malignant has higher values than the class benign because the total count and the bar are bigger.

2.3 Underlying relationship

```
## Displays the plot with geom_jitter
ggplot(data = breast_long, mapping = aes(x = property, y = value,
                                         color = class)) +
  geom_jitter(alpha = 0.8, stat = "identity") +
  scale_colour_manual(name = "class", values = c("steelblue", "plum1"),
                      aesthetics = c("color", "fill")) +
  labs(x = "Property (n/a)", y = "Value (n/a)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  ## Get the names from the codebook as labels
  scale_x_discrete(labels = sort(codebook$Full.Name[2:10])) +
  scale_y_continuous(breaks = c(1:10))
```

```
## Warning: Removed 16 rows containing missing values (geom_point).
```

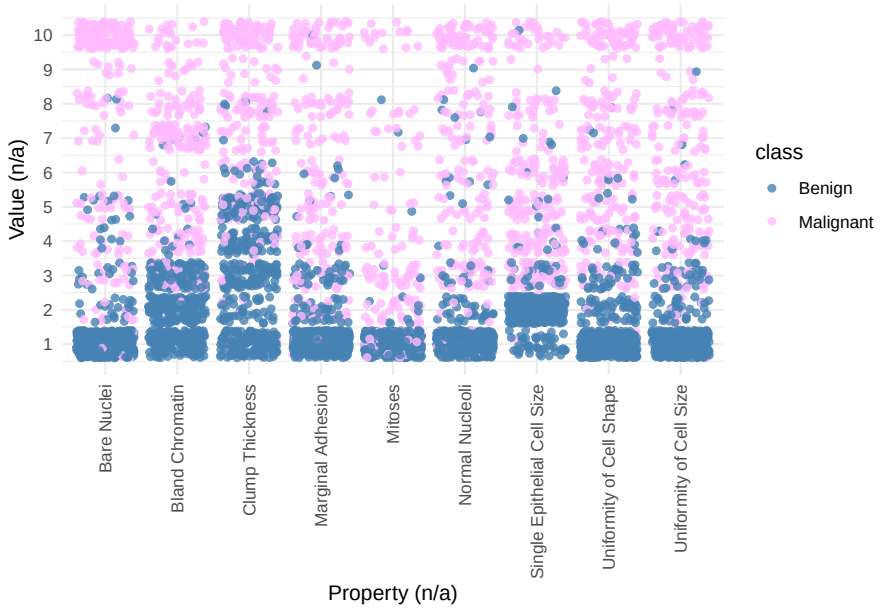


Figure 2: Relationship between the properties and their values for both the classes of breast cancer.

Figure 2 shows the relationship between the properties and their values for both classes. All of the properties have a variation for all values except mitoses, here the values are very low. Here is it easy to see that benign consists of mainly 1 values because at value 1 there are a lot of dots in one group for every property except single epithelial cell size there lies the group at value 2. Most of the values for benign lie from 1 until 5. For malignant are the dots scattered over the value range. But there are also groups at value 10 for most of the properties. So here can be seen that malignant mainly consist of value 10. The variation and the distribution of both classes for the properties uniformity of cell size and uniformity of cell shape look alike so these can both be useful to predict the cancer stage.

2.4 Displaying the distribution of breast cancer data

```
## Plot the plot with geom_line
ggplot(breast_long, aes(x = value, y = property)) +
  geom_line(aes(color = class, linetype = class)) +
  scale_color_manual(values = c("darkred", "steelblue")) +
  theme_minimal() +
  labs(x = "Value (n/a)", y = "Property (n/a)",
       colour = "Class", linetype = "Class") +
  ## Get the names from the codebook as labels
  scale_y_discrete(labels = sort(codebook$Full.Name[2:10])) +
  scale_x_continuous(breaks = c(1:10))
```

```
## Warning: Removed 16 row(s) containing missing values (geom_path).
```

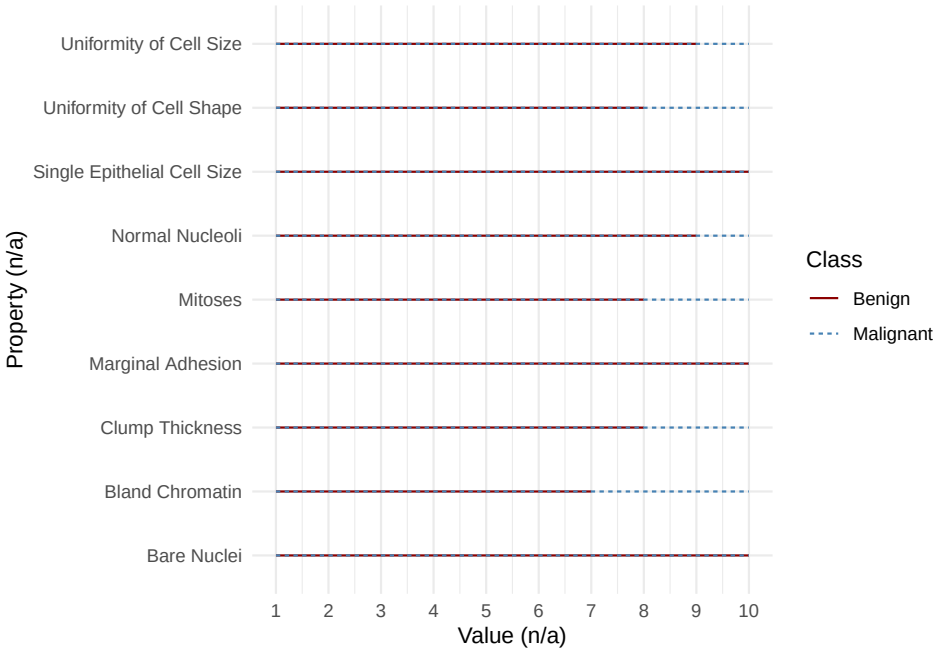


Figure 3: Correlation of the values per property divided into the two stages of breast cancer using a line plot.

Figure 3 displays the relationship between the properties and their values for both stages of breast cancer. The blue dotted line shows the stage benign and the red continuous line shows the stage malignant. In the figure 3 can be seen that every property has a line for malignant until 10 so malignant contains every value from 1 through 10. For benign are there three properties with a value of 1 through 10 which are Single Epithelial Cell Size, Marginal Adhesion, and Bare Nuclei. The other properties have for benign every value through between 7 and 9 where Bland Chromatin consist of values until 7. From the figure can be seen that benign consist mostly of the low values and that malignant consist of every value but mostly high values. Just like in figure 2 can be seen that benign has their values very low as seen at the blue color. Malignant on the other hand has more variation and higher values as seen in figure 2.

To check the class distribution, so if the classes are evenly or unevenly represented a bar plot is used.

```
## Plot the bar plot using ggplot
ggplot(data = breast_long, mapping = aes(class, value)) +
  geom_bar(stat = "identity", aes(color = class, fill = class)) +
  scale_fill_manual(values=c("steelblue", "plum1"),
                    aesthetics = c("color", "fill")) +
```

Table 4: Total number of the class

Class	Total
Benign	458
Malignant	241

```
  labs(x="Class (n/a)", y="Total count (n/a)", color = "Class",
       fill = "Class") +
  theme_minimal()
```

```
## Warning: Removed 16 rows containing missing values (position_stack).
```

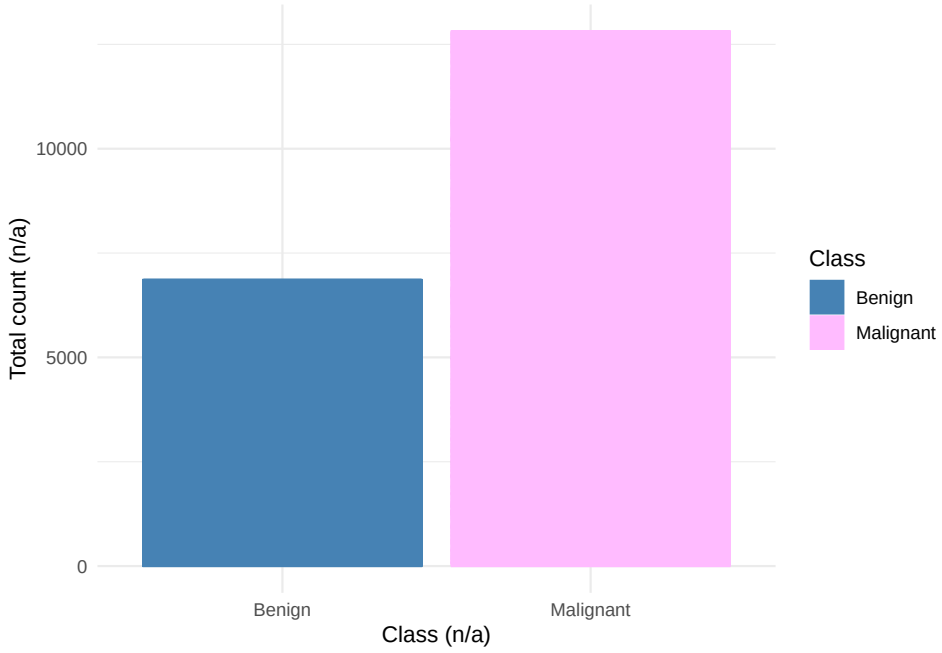


Figure 4: Bar plot of the total number of values per class.

The bar plot in figure 4 shows the total number of values per class. Here you can see that the classes are unevenly represented, the total is higher for the class malignant. But this can be, as described in figure 1, that the class benign mainly consist of low values such as 1 and that the class malignant mainly consists of high values such as 10. The figure shows the total of these values. So this could explain why the bar for the class malignant is higher. This doesn't say that there is more malignant than benign.

```
kable(table(breastcancer$class), col.names = c("Class", "Total"),
      caption = "Total number of the class")
```

Table 4 shows the total number of benign and malignant. Figure 4 shows that the total of malignant is higher than benign but in table 4 you can see that the total of the class benign is more. This is because table 4 shows the total number of rows with the class and figure 4 shows the total count of the values 1 to 10 for each class. So in table 4 you can see that there isn't more malignant than benign. So here are the classes also unevenly represented but then the other way around.

2.5 Correlation of breast cancer

```
## Plot the count plot
ggplot(data = breast_long, mapping = aes(x = property, y = value)) +
  geom_count(aes(color = class), alpha = 0.6) +
  geom_smooth(aes(group = class, color = class), formula = y ~ x,
              method = "lm", se = FALSE) +
  scale_fill_manual(values = c("steelblue", "darkred"),
                    aesthetics = c("color", "fill")) +
  labs(x="Property (n/a)", y="Value (n/a)", color = "Class", fill = "Class") +
  theme_minimal() +
  theme(axis.text = element_text(angle = 90)) +
  ## Get the names from the codebook as labels
  scale_x_discrete(labels = sort(codebook$Full.Name[2:10])) +
  scale_y_continuous(breaks = c(1:10))
```

```
## Warning: Removed 16 rows containing non-finite values (stat_sum).
```

```
## Warning: Removed 16 rows containing non-finite values (stat_smooth).
```

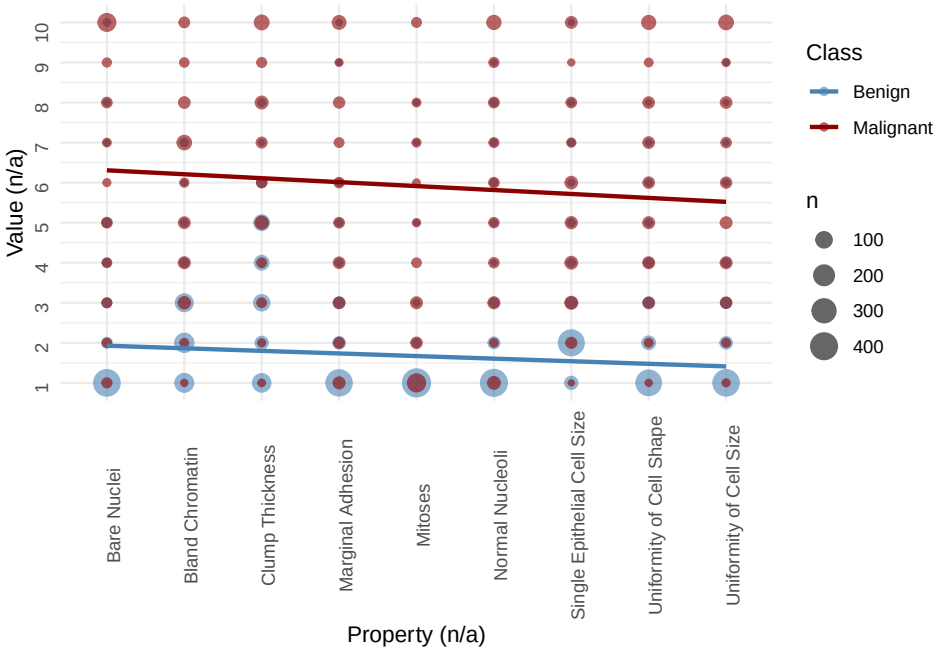


Figure 5: Bubble plot with the total number of values per property divided into the stage of cancer. A predicted linear regression is added for each stage of cancer.

The bubble plot in figure 5 shows the total count of the values, the bubbles, per property for each class. The plot contains a linear regression of the blue and red lines per stage of cancer. The legend of the bubble plot contains n which has bubbles for each total count of the values. Benign in figure 5 is colored blue and can be seen mostly between values 1 through 4, for values 1 and 2 has benign big bubbles with an n of 400. There are small bubbles by values 4 and 5 but the bubbles consist mainly of malignant. The linear regression line of benign decreases from 2 through 1.5 so it has a negative relationship. For the class malignant are there bubble everywhere except at value 9 by mitoses. Mitoses has as only property for malignant the biggest bubble of n and consist mostly of low values. Malignant has mainly high values as seen in 5 which are non-transparent, the transparent for malignant has also a count for benign. The linear regression of malignant lies higher than the one for benign, its starts around 6.5 through 5.5 and has also a negative relationship because it decreases.

```r
## Get the correlation matrix
cormat <- signif(cor(na.omit(breastcancer[2:10])))

## Plot the heatmap
pheatmap(cormat, labels_row = codebook$Full.Name[-1],
         labels_col = codebook$Full.Name[-1])
```

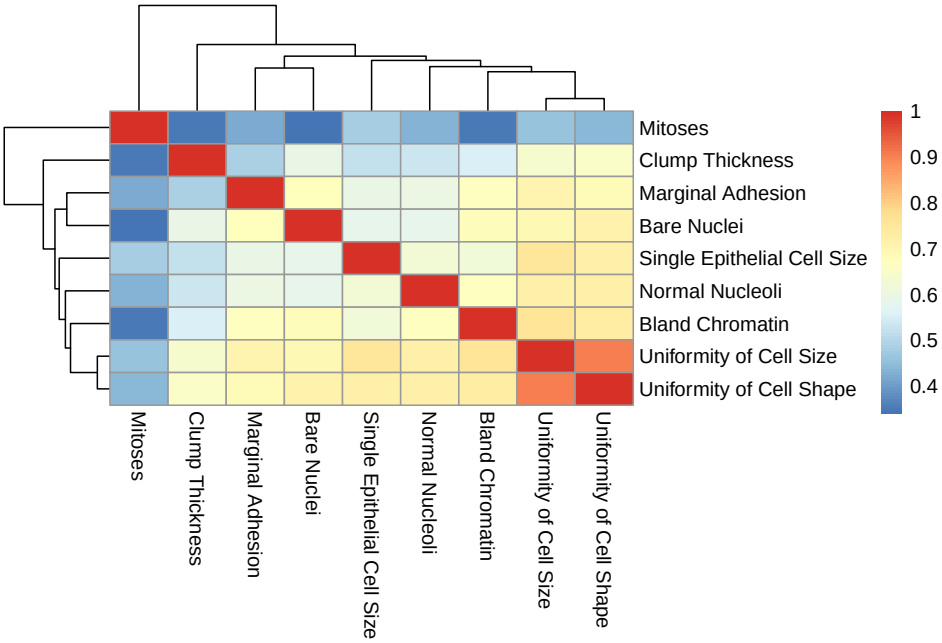


Figure 6: Heatmap with the correlation matrix of the properties.

Figure 6 shows the correlation of the properties. Red shows that the properties are highly correlated and blue shows a low correlation. Here in figure 6 are the properties highly correlated with themselves because they contain the same values. The heatmap shows a patron in the middle because this contains a correlation value between 0.5 and 1 for every property. Only the outside of the heatmap shows a lower correlation for the property mitoses compared with the other properties. Here is the correlation value between 0 and 0.5 and is colored blue. There are two properties with a higher correlation for each other these are Uniformity of cell size and Uniformity of cell shape. The blocks of these properties are slightly lighter than the dark ones. So this shows that these properties have a high correlation with each other. This can also be seen at the branches of the dendrogram because here are the two properties in one cluster.

2.6 Clustering of the dataset

The next figures are clusters based on two properties the uniformity of cell size and shape.

```
## Make 2 clusters
kmeans.cluster <- kmeans(is.na(breastcancer), centers = 2)

clusters <- as.factor(kmeans.cluster$cluster)

## Plot cluster 1 for uniformity of cell size
clust1 <- ggplot(breastcancer, aes(uniformity_of_cell_size, class,
                                   color = clusters)) +
  geom_point(size = 0.5, position = position_jitter(height = 0.3,
                                                    width = 0.3)) +
  labs(x="Uniformity of cell size", y="Class") +
  theme_minimal() +
  scale_x_continuous(breaks = c(1:10))

## Plot cluster 2 for uniformity of cell shape
clust2 <- ggplot(breastcancer, aes(uniformity_of_cell_shape, class,
                                   color = clusters)) +
  geom_point(size = 0.5, position = position_jitter(height = 0.3,
                                                    width = 0.3)) +
  labs(x="Uniformity of cell shape", y="Class") +
  theme_minimal() +
  scale_x_continuous(breaks = c(1:10))

## Combine the two plots
combined <- plot_grid(clust1 + theme(legend.position = "none"),
                      clust2 + theme(legend.position = "none"),
                      labels = c('A', 'B'), label_size = 12)

## Get one legend for both plots
combined_legend <- get_legend(clust2 + guides(color = guide_legend(nrow = 1)) +
    theme(legend.position = "bottom"))

## Plot the combined plot
plot_grid(combined, combined_legend, ncol = 1, rel_heights = c(1, .1))
```

Figure 7A shows a K-means cluster of the property uniformity of cell size for both classes. It shows that there is a cluster for the class benign at value 1, and there is a small cluster at values 2 and 3. The other blue dots are scattered between values 4 and 9. For the class malignant is there a cluster at value 10 and smaller clusters at value 3 until 8. For this class are the blue dots are more scattered over the values. The cluster for both classes can be explained that benign has in total more 1 values and malignant has in total more 10 values. By both classes are the pink dots scattered in the plot these are small clusters.

Figure 7B looks almost the same as in figure 7A but here are the blue dots more scattered between the values 3 until 8 and there are more pink clusters. The class malignant has in one eye the same shape of clusters as in figure 7A and it also has two pink clusters.

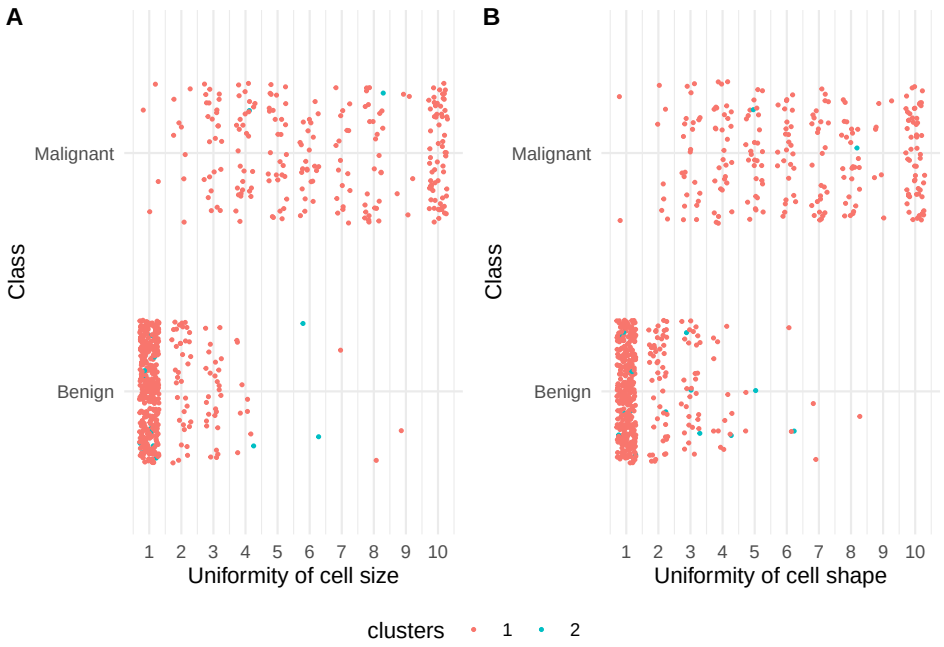


Figure 7: kMeans cluster of uniformity of cell size and shape per class.

2.6.1 Principal Components Analysis (PCA) of the dataset

```
## Remove NA's
bc <- na.omit(breastcancer[,2:10])

## Call prcomp for principal component analysis
bc_pca <- prcomp(as.matrix(bc), scale. = T, center = T)
screeplot(bc_pca, type = "l", main = "A: Scree plot")

summary(bc_pca)
```

```
## Importance of components:
##                            PC1     PC2     PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.4289 0.88088 0.73434 0.67796 0.61667 0.54943 0.54259
## Proportion of Variance 0.6555 0.08622 0.05992 0.05107 0.04225 0.03354 0.03271
## Cumulative Proportion  0.6555 0.74172 0.80163 0.85270 0.89496 0.92850 0.96121
##                            PC8     PC9
## Standard deviation     0.51062 0.29729
## Proportion of Variance 0.02897 0.00982
## Cumulative Proportion  0.99018 1.00000
```

```
## Get the rows with the NA's to removed for grouping in ggbiplot
removed <- as.numeric(subset(row.names(breastcancer),
                             is.na(breastcancer$bare_nuclei)))

## Plot the PCA
ggbiplot(bc_pca, obs.scale = 1, var.scale = 1, ellipse = TRUE,
                     circle = TRUE, group = breastcancer$class[-removed]) +
  scale_color_discrete(name = "") +
  labs(subtitle = "B") +
  theme_minimal() +
```

```
theme(legend.direction = 'horizontal', legend.position = 'top')
```

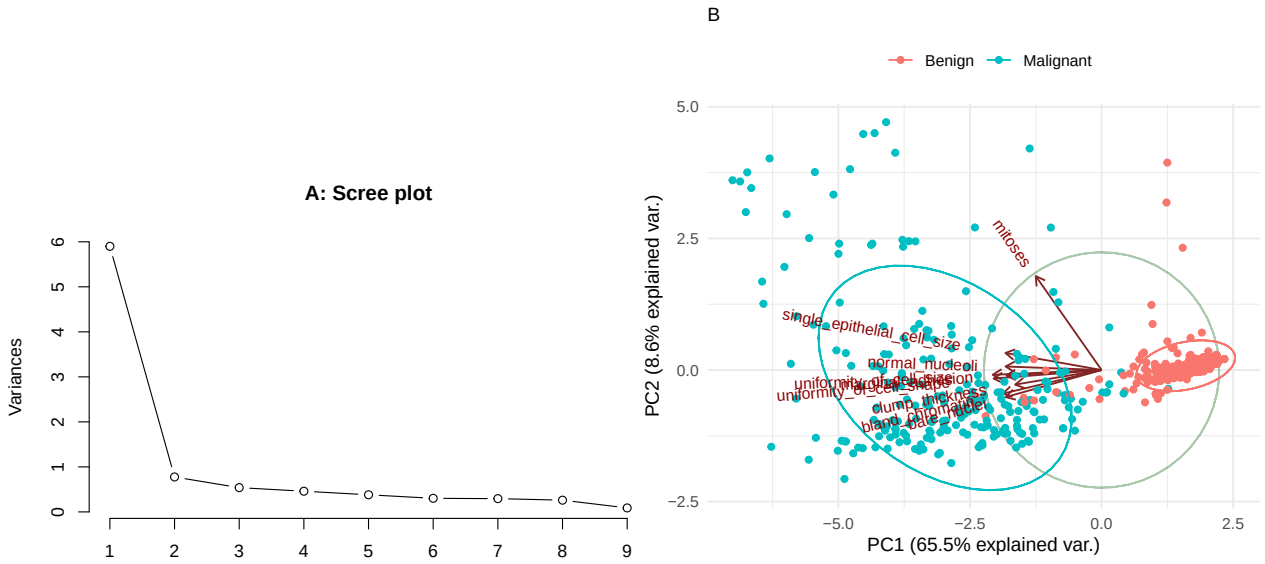


Figure 8: PCA plot with the variances of the data breast cancer. A: Scree plot with the variance of each PC. B: PCA plot with scores of the variation of PC1 and PC2.

The summary of PCA shows the variance related measures of components. Here can be seen at the proportion of variance that first component, PC1, explains 65,5% of the variability in the data set. Second component, PC2, explains 8,62% of the total variability in the data set. This indicates a total variation that is shown at the line cumulative proportion of 74,17%. The percentage of PC1 and PC2 are shown in the labels of the PCA plot 8B on the axes.

Figure 8 shows a principal component analysis (PCA) where figure 8A shows the scree plot of the variances and figure 8B shows a biplot for the first two principal components. The scree plot 8A shows how much variation each PC captures from the data where the x-axis shows the number of PC's and the y-axis shows the variances, the amount of variation. In the scree plot can be seen that PC1 and PC2 describe the data the best because these points lie higher than the other points. From 2 until 9 have the lines the least variation between the points. Thus from PC1 and PC2 is the PCA plot made as seen in figure 8B. The PCA plot 8B shows the scores of samples displayed as points and the loadings of variables displayed as vectors. The points are divided into two classes benign and malignant as seen in the legend. The points of benign all lie at the right side of the PCA and form a cluster shown by the circle, except for a few points. And the points of malignant all lie at the left side of the PCA and from also a cluster shown by the circle, except for a few points. The PCA has a PC origin at [0,0], how further the vectors are from the PC origin, how more influence they have on that PC. In the plot can be seen that mitoses has a longer vector and points in a different direction compared to the other properties. Because mitoses has a longer vector, it has a negative correlation and it lies further from the PC origin. Mitoses is also outside of the cluster from benign while the other properties lie in the cluster of benign.

3 Cleaning the breast cancer dataset

```
## Get the total number of value per column
kable(colSums(na.omit(breastcancer[2:10])),
      col.names = "Total number of values",
      caption = "The total number of values per column") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 5: The total number of values per column

	Total number of values
clump_thickness	3034
uniformity_of_cell_size	2152
uniformity_of_cell_shape	2196
marginal_adhesion	1933
single_epithelial_cell_size	2209
bare_nuclei	2421
bland_chromatin	2353
normal_nucleoli	1960
mitoses	1095

```
## Remove the column of mitoses
breast_clean <- breastcancer[-10]

## Tidying the data by using pivot_longer
clean_long <- pivot_longer(data = breast_clean,
                           cols = -c("ID", "class"),
                           names_to = "property",
                           values_to = "value")
```

Table 5 shows the total number of values per property. In the table can be seen that there most of the total lies between 1900 and 3000 but there is only one exception that is for mitoses here are there only 1095. In all the figures above especially in figure 6 and 8B can be seen that mitoses is different than the other properties. Because in the heatmap 6 shows mitoses for both classes a very low correlation value of 0. In the PCA plot 8B points the vector of mitoses the other way compared to the other properties which point in the same direction and lie in the same cluster. Therefore the column mitoses is removed from the dataset.

```r
## Get the correlation matrix
cormat <- signif(cor(na.omit(breast_clean[2:9])))

## Plot the heatmap
pheatmap(cormat, labels_row = codebook$Full.Name[-1],
         labels_col = codebook$Full.Name[-1])
```

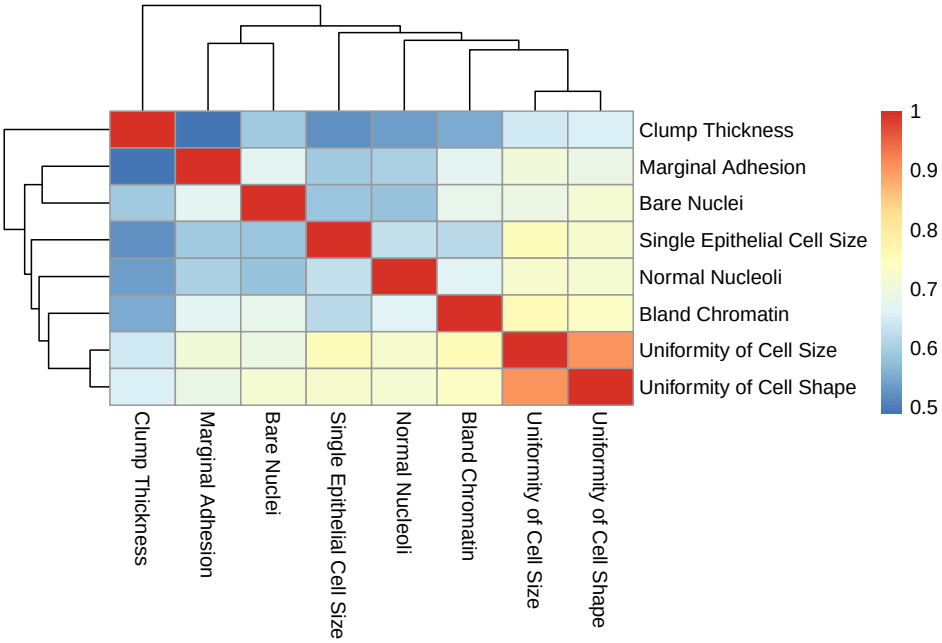


Figure 9: Heatmap with the correlation matrix of the properties. After the removal of the property mitoses.

Figure 9 shows the correlation matrix in a heatmap for all properties. Here can be seen that the property mitoses is removed compared to the heatmap in figure 6. The legend with the colors is also adapted, here is the lowest value 0.5, and in figure 6 0.3. In the heatmap 9 are the properties highly correlated with themselves because they contain the same values. The heatmap shows a better patron than in figure 6 because mitoses isn't shown here otherwise, the figure has remained the same. The two properties Uniformity of cell size and Uniformity of cell shape show a high correlation with each other and are in the same cluster.

```
## Create a csv file for Weka
write.csv(breast_clean, "/homes/crhazenberg/Thema9/MachineLearning/breastcancer.csv",
          row.names = FALSE, na = "")
```

4 Determine quality metrics

After a machine learning performance shows the performance an accuracy which is the default metric. But other metrics can be applied and are maybe more accurate and relevant.

For the dataset, there are two classes benign and malignant but it is important that these classes are correctly predicted and performed. Because when the test shows benign then you don't want to be malignant after all and when you are malignant you want to be sure that you are and not that it is actually benign so that the test is incorrect.

So by decreasing the number of false positives (FP). Thus it is important that the predicted class benign, which is actually malignant, is reduced. It is not important that the false negatives are reduced because the prediction is malignant and the actual class is benign and this is good because you want to be benign before and after a test and not malignant. As seen in table 6 where the first row is the predicted class and the second and third row is the real class of breast cancer. Here stand FP by the predicted class benign but the actual class is malignant. It is important that the classification is accurate but it should be fast because if the test shows malignant then they need to treat it as soon as possible before it gets worse. So in a more accurate metric should FP be decreased.

```
matrix <- read.csv("wekadata/matrix.csv", sep = ";", header = FALSE)
## Set rownames and colnames to null
rownames(matrix) <- colnames(matrix) <- NULL
kable(matrix, caption = "Confusion matrix of the breast cancer dataset") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 6: Confusion matrix of the breast cancer dataset

Predicted		Benign	Malignant
Real	Benign	TP	FN
	Malignant	FP	TN

The confusion matrix consist of the following layout in Weka: For this matrix is benign positive and malignant negative. So it is positive that you don't have cancer and you are benign. And it is negative that you have cancer because you don't want cancer but you are malignant.

=== Confusion Matrix ===

a b <– classified as
TP FN | a = Benign
FP TN | b = Malignant

5 Performance of Machine Learning algorithms

With the clean dataset, we start investigating the performance of all standard ML algorithms with the standard settings in Weka.

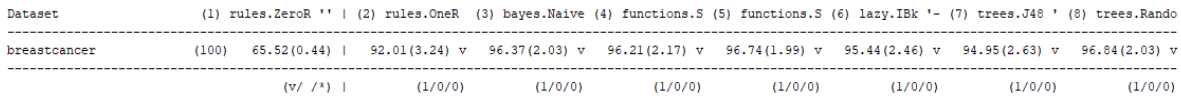
```
classifiers <- read.csv("wekadata/classifiers.csv", header = TRUE, sep = ";")
kable(classifiers, caption = "Classifications using 10-fold cross validation") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 7: Classifications using 10-fold cross validation

Classifier	Speed	Accuracy	TP	FP	FN	TN
ZeroR	0	65,5222	458	241	0	0
OneR	0,01	92,7039	444	37	14	204
NaiveBayes	0	96,4235	439	6	19	235
SimpleLogistic	0,29	95,7082	446	18	12	223
SMO	0,03	96,5665	444	10	14	231
IBk	0	95,1359	443	19	15	222
J48	0,02	94,9928	438	15	20	226
RandomForest	0,17	97,1388	443	5	15	236

Table 7 shows the results of the machine learning algorithms with 8 classifier categories. The results show the speed, accuracy, True Positive (TP), False Positive(FP), True Negative(TN), and False Negative(FN), the last four are part of the confusion matrix in Weka. The accuracy is shown in percentages. The first row shows the outcome of the classifier ZeroR, here can be seen that the accuracy is 65,5222% and that there are only TP 458 and FP 241, FN and TN have both a value of 0. OneR has an accuracy of 92,7039% which is already higher than for ZeroR. OneR has 444 TP, 37 FP, 14 FN, and 204 TN which has thus more FN and TN than ZeroR. The next classifier in the table is NaiveBayes, this classifier has a higher accuracy than both ZeroR and OneR which is 96,4235% NaiveBayes contains 439 TP, 6 FP, 19 FN, and 235 TN. Another classifier that is used is SimpleLogistic which has an accuracy of 95,7082% it also has 446 TP, 18 FP, 12 FN, and 223 TN. The fifth row of table 7 contains the results of the classifier SMO. SMO has an accuracy of 96.5665% which has so far the highest accuracy. It has 444 TP, 10 FP, 14 FN, and 231 TN. The next classifier is IBk with an accuracy of 95,1359% and it has 443 TP, 19 FP, 15 FN, and 222 TN. The seventh classifier is J48 with an accuracy of 94,9928%. J48 has 438 TP, 15 FP, 20 FN, and 226 TN. The last classifier is RandomForest which has an accuracy of 97,1388% and has thus the highest accuracy of all the classifiers. RandomForest has 443 TP, 5 FP, 15 FN, and 236 TN. In general, all the classifiers have a low speed which is the time it takes to build the model in seconds, SimpleLogistic takes the longest to build the model. Comparing the accuracy for all the classifiers you can be seen that ZeroR has the lowest accuracy this is because ZeroR has for both FN and TN 0. ZeroR shows the most likely class with only the value of both class benign and malignant which is the total number of the classes just like in 4. RandomForest has the highest accuracy and the other classifiers lie in between with little difference between them. Every classifier has in the 400 True Positives and in the 200 True Negatives except ZeroR. The False Positives and the False Negatives are different for every classifier but no major difference between them. Only OneR has a bigger difference in the number of False Positives compared to the other classifiers. The two algorithms with the highest accuracy are RandomForest and SMO. These algorithms are probably the most effective because of their high accuracy.

```
knitr::include_graphics("images/experimenteroutput.png")
```



Dataset	(1) rules.ZeroR ''	(2) rules.OneR	(3) bayes.Naive	(4) functions.S	(5) functions.S	(6) lazy.IBk '-	(7) trees.J48 '	(8) trees.Rando
breastcancer	(100) 65.52(0.44)	92.01(3.24) v	96.37(2.03) v	96.21(2.17) v	96.74(1.99) v	95.44(2.46) v	94.95(2.63) v	96.84(2.03) v
	(v/ /*)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)	(1/0/0)

Figure 10: Accuracy of the algorithms for breast cancer. v is here significant higher than the test base ZeroR and * is significant lower than ZeroR.

Image 10 shows the output of the experiment of the same algorithms in table 7 but with 10 repetitions. ZeroR is the test base so the other algorithms are compared to ZeroR which has an accuracy of 65,52% also in table 7. For OneR is the accuracy 92,01% with a standard deviation of 3,24 and in table 7 is this 92,7039% for 1 repeat. This is in line with each other because of the standard deviation it can have a difference of 3,24. OneR has a significantly higher accuracy than ZeroR which can be seen at the v after the standard deviation. NaiveBayes has an accuracy of 96,37% for 10 repeats with a standard deviation of 2,03 and it is also significantly higher than ZeroR. In table 7 has NaiveBayes an accuracy of 96,4235% for one repeat which is in line with each other. SimpleLogistic has an accuracy of 96,21% in image 10 with a standard deviation of 2,17 and it is also significantly higher than ZeroR. In table 7 has SimpleLogistic an accuracy of 95,7082% which is lower than in image 10 but it has a standard deviation of 2,17 so this has no effect because it has another number of repetitions. SMO has an accuracy of 96,74% with a standard deviation of 1,99 and is also significant higher than ZeroR. In table 7 has SMO an accuracy of 96,5665% which has a low difference with each other. IBk has an accuracy of 95,44% and a standard deviation of 2,46. IBk has an accuracy of 95,1359% in table 7 and has a little difference. Number 7, J48 has an accuracy of 94,95% with a standard deviation of 2,63. J48 has a very low difference with 1 repeat as seen in table 7. The last one RandomForest has an accuracy of 96,84% but it has a higher accuracy for one repeat as seen in table 7 which is 97,1388% but it has a standard deviation of 2,03 so it can differ. For the two best algorithms it doesn't matter because SMO and RandomForest have still the highest accuracy after 10 repeats.

```
knitr::include_graphics("images/costsensitiveclassifier.png")
```

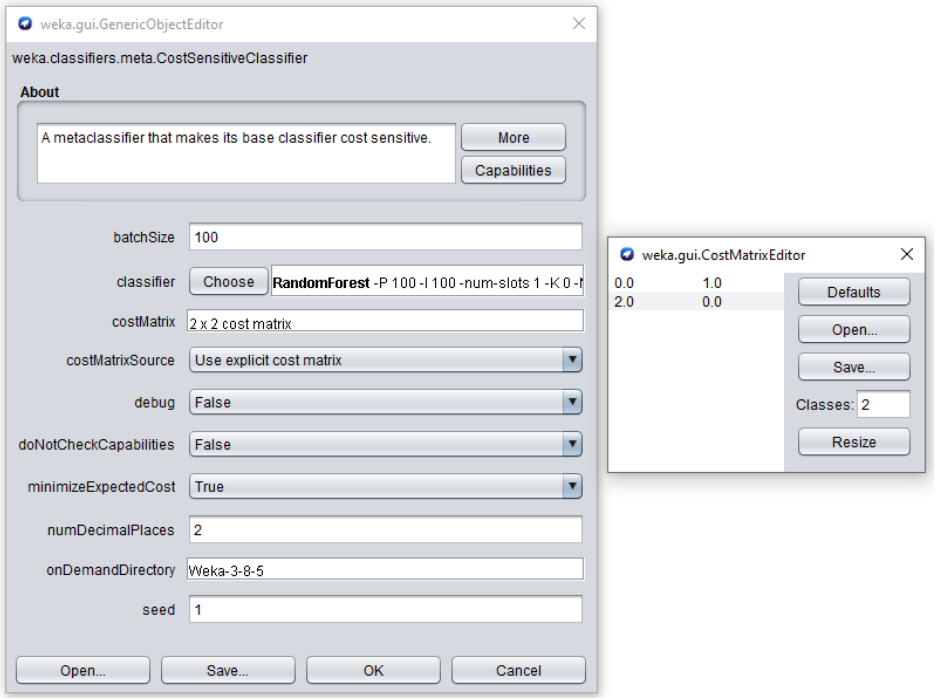


Figure 11: The settings of the CostSensitiveClassifier and the CostMatrix.

The image 11 shows the settings of the CostSensitiveClassifier. The minimizeExpectedCost is set to True, this is called cost-sensitive classification. Here the cost is limited by adjusting the probability threshold while applying the model. The model doesn't change. In the CostMatrixEditor the 2.0 is adapted so that the FP in the cost matrix is reduced.

```
cost <- read.csv("wekadata/costsensitiveclassifier.csv", header = TRUE,
                 sep = ";")
kable(cost, caption = "CostSensitiveClassifier for the two algorithms") %>%
   kable_styling(latex_options = "HOLD_position")
```

Table 8: CostSensitiveClassifier for the two algorithms

Classifier	costMatrix	TP	FP	FN	TN	Accuracy
RandomForest	1	443	5	15	236	97,1388
RandomForest	2	436	2	22	239	96,5665
RandomForest	3	435	2	23	239	96,4235
RandomForest	4	430	1	28	240	95,8512
SMO	1	444	10	14	231	96,5665
SMO	2	444	10	14	231	96,5665
SMO	3	444	10	14	231	96,5665
SMO	4	444	10	14	231	96,5665

Table 8 shows the CostSensitiveClassifier for the most effective algorithms RandomForest and SMO. The costMatrix of the CostSensitiveClassifier is adapted for each build. For RandomForest has each row a different confusion matrix and accuracy while SMO has the same confusion matrix and accuracy for each build. In the

costMatrix is the cost of the False Negative in the confusion matrix increased so the False Positive decreases. This is necessary because we don't want people who are actually benign predicted to be malignant. With a normal matrix of equal values, the number of FP is 5 for RandomForest. With an increase of the costMatrix for FP, the number decreases to 2 and 1. The accuracy also decreases first is the accuracy 97,1388 as seen at the first row of table 8. Even the number of TP decreases but the number of FN and TN increases. FN increases because there are people who are tested malignant but they are actually benign, so it isn't bad that the number of FN increases. The classifier SMO has for every costMatrix the same accuracy and the same values. For RandomForest is the costMatrix of 2 is the best because this has an high accuracy and the TP is still high compared to the costMatrix of 3 and 4. This is important because the people who are benign are actually benign and we want people to be benign not malignant.

6 Attribute Selection

Here is the effect of the Attribute Selection methods investigated.

6.1 Single attribute evaluation

For single attribute evaluation, which deletes attributes that doesn't say much about the classes, are there four evaluators applied.

```
knitr::include_graphics("images/settingattributeselection.png")
```

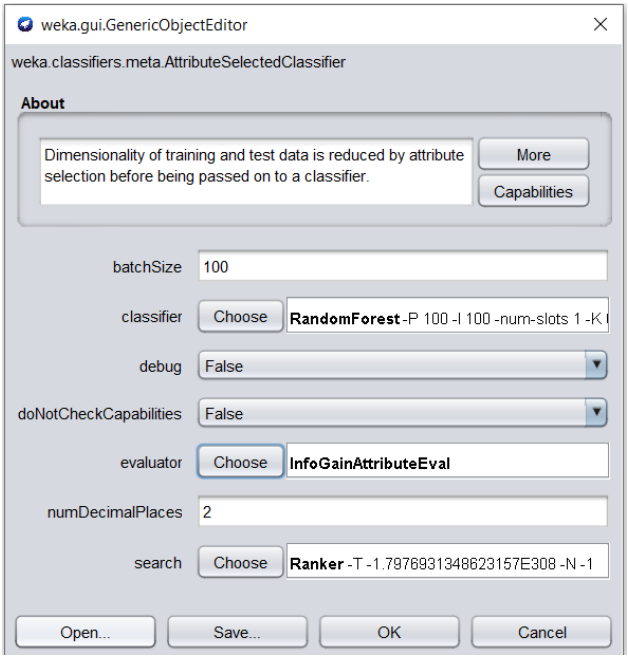


Figure 12: The settings of the AttributeSelectedClassifier.

In image 12 is the classifier changed to RandomForest and SMO the two best algorithms. The evaluator is also adapted to *CorrelationAttributeEval*, *OneRAttributeEval*, *InfoGainAttributeEval*, and *GainRatioAttributeEval*. The method search isn't changed this is only set to Ranker.

```
## Read .png to create images
correlation <- readPNG("images/correlation.png")
oner <- readPNG("images/oner.png")
infogain <- readPNG("images/infogain.png")
gainratio <- readPNG("images/gainratio.png")

## Plot each image to combine
im_A <- ggplot() +
    annotation_custom(rasterGrob(correlation))

im_B <- ggplot() +
    annotation_custom(rasterGrob(oner))

im_C <- ggplot() +
    annotation_custom(rasterGrob(infogain))

im_D <- ggplot() +
```

```
    annotation_custom(rasterGrob(gainratio))

## Combine and plot the images
plot_grid(im_A, im_B, im_C, im_D, labels = c("A", "B", "C", "D"))
```

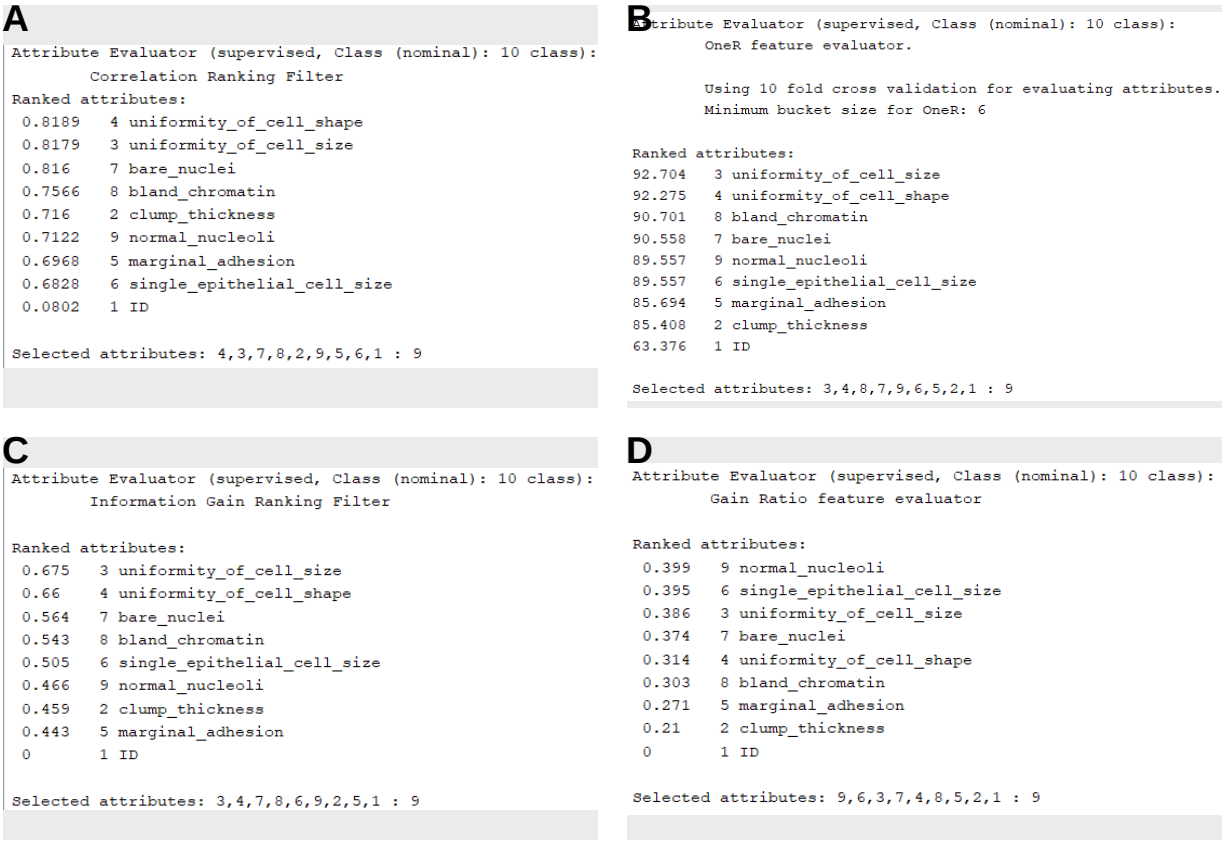


Figure 13: The outcome of the evaluators from the AttributeSelectedClassifierf for RandomForest and SMO. Subfigure A shows the outcome of the CorrelationAttributeEval. Subfigure B shows the outcome OneRAttributeEval. Subfigure C shows the outcome of InfoGainAttributeEval. Subfigure D shows the outcome GainRatioAttributeEval.

Image 13 shows for every evaluator the ranked attributes and the selected attributes for both RandomForest and SMO because the outcome is the same for both algorithms. Subfigure 13A shows the output of the correlation ranking filter with the ranked and the selected attributes. The ranked attributes show the correlation with the attribute ranked from high to low, so uniformity_of_cell_shape has the highest correlation and uniformity_of_cell_size has the second highest correlation. The order of the ranked attributes is also shown by the selected attributes which gives an overview of the order of the attributes. Subfigure 13B shows the ranked and selected attributes from the OneR feature evaluator. The ranked attributes show the accuracy of OneR and their attribute. Here is uniformity_of_cell_size ranked the highest and uniformity_of_cell_shape is now ranked second. The overview of the selected attribute is different compare to the correlation ranking. In subfigure 13C contains the information gain ranking filter with the information gain by ranked attributes with the attribute. Here is uniformity_of_cell_size the highest and uniformity_of_cell_size second just like for OneR but the order of the attribute is a little different. For gain ration feature evaluator in subfigure 13D can be seen that normal_nucleoli is ranked the highest with an information gain of 0.399 and is single_epithelial_cell_size ranked second best. So here is the order shown in selected attributes thus different than the other subfigures. ID has for the evaluator of the AttributeSelectedClassifier the lowest ranking and has for information gain and gain ration even 0. This is because ID isn't an attribute to determine the stage

of cancer.

```
single_attribute <- read.csv("wekadata/singleattribute.csv", header = TRUE,
                             sep = ";")
kable(single_attribute, caption = "Outcome of the AttributeSelectedClassifier
      using 10-fold cross validation") %>%
   kable_styling(latex_options = "HOLD_position")
```

Table 9: Outcome of the AttributeSelectedClassifier using 10-fold cross validation

Classifier	Evaluator	Search	Accuracy	TP	FP	FN	TN
RandomForest	CorrelationAttributeEval	Ranker	97,1388	443	5	15	236
RandomForest	OneRAttributeEval	Ranker	97,1388	443	5	15	236
RandomForest	InfoGainAttributeEval	Ranker	97,1388	443	5	15	236
RandomForest	GainRatioAttributeEval	Ranker	97,1388	443	5	15	236
SMO	CorrelationAttributeEval	Ranker	96,5665	444	10	14	231
SMO	OneRAttributeEval	Ranker	96,5665	444	10	14	231
SMO	InfoGainAttributeEval	Ranker	96,5665	444	10	14	231
SMO	GainRatioAttributeEval	Ranker	96,5665	444	10	14	231

Table 9 shows the options and the outcome of the AttributeSelectedClassifier. The outcome for every classifier and the options evaluator and search are the same per classifier. The only difference can be seen in image 13 but here is the outcome of the evaluator per classifier also the same. So 13A shows for both RandomForest and SMO the outcome of CorrelationAttributeEval. 13B shows for both the classifiers the outcome of OneRAttributeEval. The image shows 13C the outcome of InfoGainAttributeEval for RandomForest and SMO. And the last one 13D shows the outcome of the GainRatioAttributeEval for RandomForest and SMO. In table 9 have accuracy, TP, FP, FN, and TN the same values as in table 7. So these evaluators of the AttributeSelectedClassifier don't have an impact on the accuracy. Just like in figure 13 where there isn't a difference between RandomForest and SMO is here no difference between the evaluators for the same algorithm.

6.2 Attribute subset selection

For attribute subset evaluation, which deletes also attributes that don't look alike, is there one evaluator applied *CfsSubsetEval* in AttributeSelectedClassifier with the search *ExhaustiveSearch* and *BestFirst*.

```
cfssubset <- read.csv("wekadata/cfssubseteval.csv", header = TRUE, sep = ";")
kable(cfssubset, caption = "Outcome of the AttributeSelectedClassifier
      with CfsSubsetEval and different a search") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 10: Outcome of the AttributeSelectedClassifier with CfsSubsetEval and different a search

Classifier	Evaluator	Search	Accuracy	TP	FP	FN	TN
RandomForest	CfsSubsetEval	ExhaustiveSearch	96,2804	443	11	15	230
RandomForest	CfsSubsetEval	BestFirst	96,2804	443	11	15	230
SMO	CfsSubsetEval	ExhaustiveSearch	96,7096	445	10	13	231
SMO	CfsSubsetEval	BestFirst	96,7096	445	10	13	231

Table 10 shows the outcome of the AttributeSelectedClassifier with the evaluator CfsSubsetEval with the search ExhaustiveSearch and BestFirst. The first two rows show the two different search for RandomForest and the last two rows shows the search for deSMO. RandomForest has for both ExhaustiveSearch and BestFirst the same accuracy of 96,2084%, TP of 443, FP of 11, FN of 15, and TN of 230. Compared to table 7 has RandomForest now more FP 11 instead of 5 which is worse because there shouldn't be more people who are tested for benign but are actually malignant. For SMO is the everything also the same, with an accuracy of 96,7096%, TP of 445, FP of 10, FN of 13, and TN of 231. SMO has a higher accuracy compared to table 7 which has an accuracy of 96,5665% and it has 1 more TP and 1 less FN.

6.3 Meta learners

The Paired T-Tester and the paired T-Tester (corrected) have both the same effect in the experimenter for the algorithms RandomForest and SMO by a number of repetitions of 10. For both statistical tests has RandomForest an accuracy of 96.84% and SMO has an accuracy of 96,74% for 10 repeats. For 1 repeat has RandomForest an accuracy of 97,14% just like in table 7 and SMO has an accuracy of 96,57% just like 7.

The meta learners that are investigated are *Vote*, *Bagging*, *Boosting*, and *Stacking*.

```
meta_learners <- read.csv("wekadata/metalearners.csv", header = TRUE, sep = ";")
kable(meta_learners, caption = "Meta learners with 10-fold cross validation and
      10 repetitions") %>%
  kable_styling(latex_options="scale_down") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 11: Meta learners with 10-fold cross validation and 10 repetitions

Algorithm	Classifier	combinationRule	metaClassfier	Accuracy	TP	FP	FN	TN
Vote	RandomForest, SMO, NaiveBayes	Average of Probabilities	n/a	97,01	444	7	14	234
Vote	RandomForest, SMO, NaiveBayes	Majority Voting	n/a	96,97	444	7	14	234
Bagging	RandomForest	n/a	n/a	96,91	443	7	15	234
Bagging	SMO	n/a	n/a	96,74	445	10	13	231
Bagging	NaiveBayes	n/a	n/a	96,38	438	6	20	235
AdaBoostM1	RandomForest	n/a	n/a	96,90	443	7	15	234
AdaBoostM1	SMO	n/a	n/a	96,74	445	9	13	232
AdaBoostM1	NaiveBayes	n/a	n/a	96,00	444	14	14	227
Stacking	RandomForest, SMO, NaiveBayes	n/a	RandomForest	96,44	442	9	16	233
Stacking	RandomForest, SMO, NaiveBayes	n/a	SMO	96,41	442	9	16	232
Stacking	RandomForest, SMO, NaiveBayes	n/a	NaiveBayes	96,95	443	6	15	235

Table 11 shows the outcome of the meta learners with 10-fold cross-validation and a repetition of 10. The first column shows the algorithm, the second column shows the classifier that is needed for the algorithm, the third column only applies for the algorithm voting, and the fourth column only applies for the algorithm stacking which needs a metaClassifier. The other columns show the accuracy, TP, FP, FN, and TN just like the tables described above. The classifiers that are used are RandomForest, SMO, and NaiveBayes. NaiveBayes is added, besides the two best classifiers, because NaiveBayes has in 7 a second best accuracy after SMO and it has 6 FP 1 more than RandomForest which is important because there need to be a low number of people who are tested benign but are actually malignant. For the first two rows, the *Vote* algorithm was used with the classifiers RandomForest, SMO, and NaiveBayes and with the combinationRule, Average of Probabilities and Majority Voting. Only the accuracy is different for these two but the confusion matrix is the same. The accuracy is different because of the combinationRule. The Average of Probabilities has the highest accuracy of them both but also for all of the algorithms. The algorithm *Bagging* is used with the classifiers RandomForest, SMO, and NaiveBayes. The algorithm with RandomForest has an accuracy of 96,91% and has also 7 FP and one more FN than Vote. SMO has an accuracy of 96,74% and has more FP than Bagging with RandomForest. NaiveBayes has the lowest accuracy and has less TP and FP but more TN so here are there more malignant.

7 ROC curve analysis

The next section visualizes Receiver Operating Characteristics (ROC) curves for both RandomForest and SMO. ROC curve is a graph showing the performance of a classification model at all threshold settings. The ROC curve is created by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR) where TPR is on the y-axis and FDR is on the x-axis.

```
## Read roc curve data
roc_data_smo <- read.table("wekadata/roccurveSMO.arff", sep = ",",
                           comment.char = "@")

## Define the names
names(roc_data_smo) <- 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")

## Create colours for the classifier and threshold
colors <- c(classifier ="orange", threshold = "blue")

## Plot the ROC curve of SMO
roc_smo <- ggplot(data = roc_data_smo, mapping = aes(x = False_Positive_Rate,
                                                     y = True_Positive_Rate)) +
  geom_point(mapping = aes(color = "classifier")) +
  geom_line(aes(color = "classifier")) +
  geom_abline(aes(color = "threshold", slope = 1, intercept = 0)) +
  scale_color_manual(values = colors) +
  ggtitle("SMO") +
  xlab("False Positive Rate") +
  ylab("True Positive Rate") +
  theme_pubr() +
  theme(legend.title = element_blank())


roc_data_rf <- read.table("wekadata/roccurverrandomforest.arff", sep = ",",
                          comment.char = "@")
```

```
names(roc_data_rf) <- names(roc_data_smo)

roc_rf <- ggplot(data = roc_data_rf, mapping = aes(x = False_Positive_Rate,
                                                   y = True_Positive_Rate)) +
  geom_jitter(mapping = aes(color = "classifier")) +
  geom_line(aes(color = "classifier")) +
  geom_abline(aes(color = "threshold", slope = 1, intercept = 0)) +
  scale_color_manual(values = colors) +
  ggtitle("RandomForest") +
  xlab("False Positive Rate") +
  ylab("True Positive Rate") +
  theme_pubr() +
  theme(legend.title = element_blank())

## Combine the two plots
combined_roc <- plot_grid(roc_smo + theme(legend.position = "none"),
                          roc_rf + theme(legend.position = "none"),
                          labels = c('A', 'B'), label_size = 12)

## Combine the legend
legend <- get_legend(roc_rf + guides(color = guide_legend(nrow = 1)) +
                       theme(legend.position = "bottom"))

## Plot the combined plots
plot_grid(combined_roc, legend, ncol = 1)
```

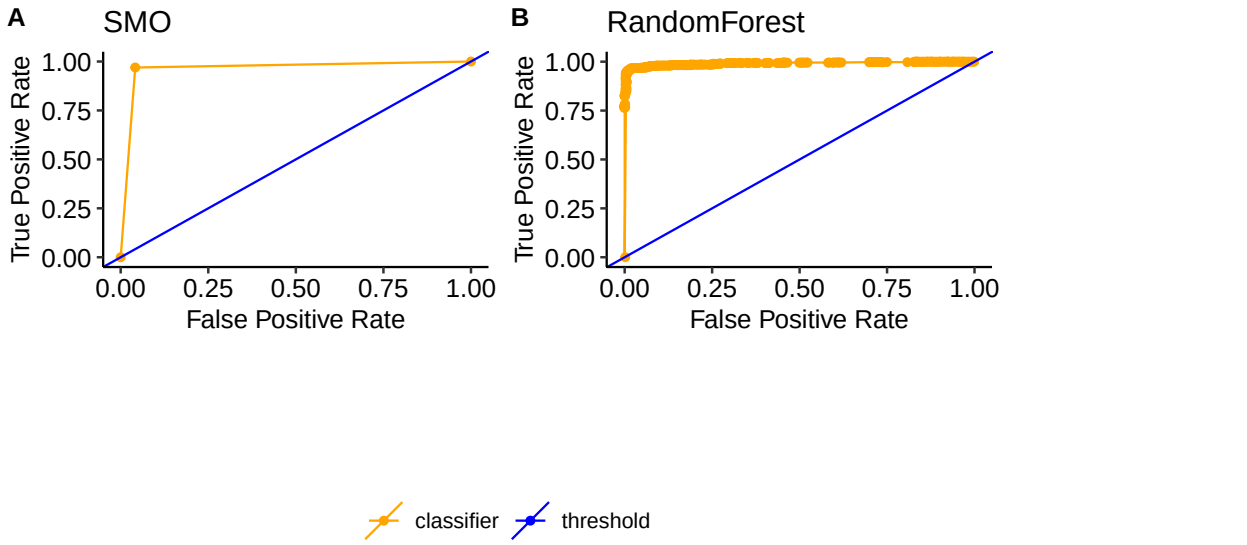


Figure 14: ROC curve for the classifiers SMO and RandomForest at all threshold setting. A shows the ROC curve of SMO and B shows the ROC curve of RandomForest

In figure 14 there are two figures with a ROC curve. The first one figure 14A shows the ROC curve from the algorithm SMO and the second figure 14B shows the ROC curve from the algorithm RandomForest. The TPR is plotted against the FPR at various thresholds which forms a line when the dots join. The area under the ROC curve which is shown in orange is called the Area Under The Curve (AUC) curve. The blue threshold line shown in the figures is a straight line which is the predicted probability of a observation. Figure 14A has three dots and is connected with a line. The line is almost 90 degrees and straight with an angle,

this shows an ideal situation because the two curves of TP and TN almost don't overlap. When the positive and negative classes overlap then there is an error and runs the model in a curve or in a straight line just like the threshold line. The ROC curve of RandomForest is shown in 14B here is also the TPR plotted against the FPR. The threshold line is here the same as for the ROC curve of SMO. Between the two figures is the biggest difference that RandomForest has more dots in the figure which means that it shows more threshold points. Just like the ROC curve of SMO has RandomForest almost the same curve but for RandomForest is the line slightly straighter. This means that this is almost an ideal situation, to be ideal the line should run at a 90 degrees angle because then there is no overlap between benign and malignant. Thus the ROC curve of RandomForest is slightly better than SMO. In Weka has the AUC a value of 0.964 for SMO and the value for AUC of RandomForest is 0.9912. This also indicates that the ROC curve of RandomForest better is than SMO because the AUC of RandomForest is closer to 1 which categories as very good.

8 Learning curve

From all the algorithms is RandomForest the best with the standard settings. So for the next part, a learning curve is created with RandomForest as the classifier.

```
knitr::include_graphics("images/learningcurve.png")
```

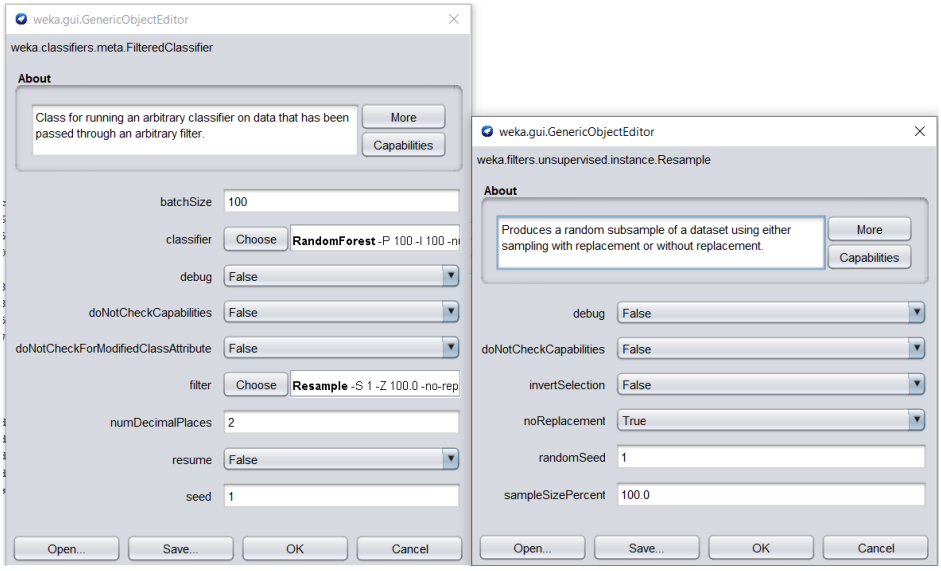


Figure 15: The settings of classifier FilteredClassifier to create a learning curve.

Image 15 shows the setting of the FilteredClassifier. The classifier is FilteredClassifier under meta and it has as classifier RandomForest and as filter Resample. The noReplacement is set to true so there are no replacements and the sampleSizePercent differs from 0% to 100% as seen in the right part of the image.

```
learning_data <- read.csv("wekadata/learningcurve.csv", header = TRUE,
                          sep = ";")

kable(learning_data, caption = "Accuracy of the FilteredClassifier with the
      different sampleSizePercent") %>%
  kable_styling(latex_options = "HOLD_position")
```

Table 12: Accuracy of the FilteredClassifier with the different sampleSizePercent

sampleSizePercent	Accuracy
1	86.4092
2	91.2732
5	93.4192
10	94.9928
20	95.1359
30	95.2790
40	95.4220
50	95.7082
60	96.9957
70	96.5665
80	96.5665
90	96.9957
100	96.5665

```
## Plot the learning curve
ggplot(learning_data, aes(x = sampleSizePercent, y = Accuracy)) +
  geom_point(colour = "darkred") +
  geom_line(colour = "darkred") +
  geom_hline(yintercept = 65, linetype ="dashed", color = "steelblue") +
  theme_minimal() +
  labs(title="Learning curve", x="Sample Size Percent (%)", y="Accuracy (%)") +
  scale_x_continuous(breaks = seq(0, 100, 10)) +
  scale_y_continuous(breaks = seq(50, 100, 5))
```

The table 12 contains two columns with the accuracy in the second column which gives the accuracy for the different sample size percentages from the classifier FilteredClassifier. Here can be seen that the accuracy of only for 1% of the sample size is below 90%. From the percentage, 20% until 50% is the accuracy in the 95% which is still growing as seen for the percentages 60 until 100 where the accuracy is in 96%. The highest accuracy is for 90% of the sample size. These values create a learning curve together which is shown in figure 16. Figure 16 has a red line with points which shows the accuracy for the different sample size percentages of RandomForest and the dashed blue line shows the baseline which is the accuracy of ZeroR for every percentage. The accuracy of ZeroR is the same because is the simplest classification method that relies only on predicting the class and ignores all attributes. And ZeroR has thus for all percentages the same accuracy. Compared to the baseline of ZeroR has RandomForest a much higher accuracy. RandomForest shows a smooth curve with one peak at 60% but otherwise is remains gradual. To get a reasonable performance estimate is there a least 60% of the data is needed because the learning curve has from 60% the same accuracies and the same difference between the accuracies. Compared to table 12 has 60% and 90% an accuracy of 96,9957% and for 70%, 80%, and 100% has RandomForest and accuracy of 96,5665%. Below 60% is the accuracy between 86% and 96% thus there is at least 60% of the data needed to get a reasonable performance.

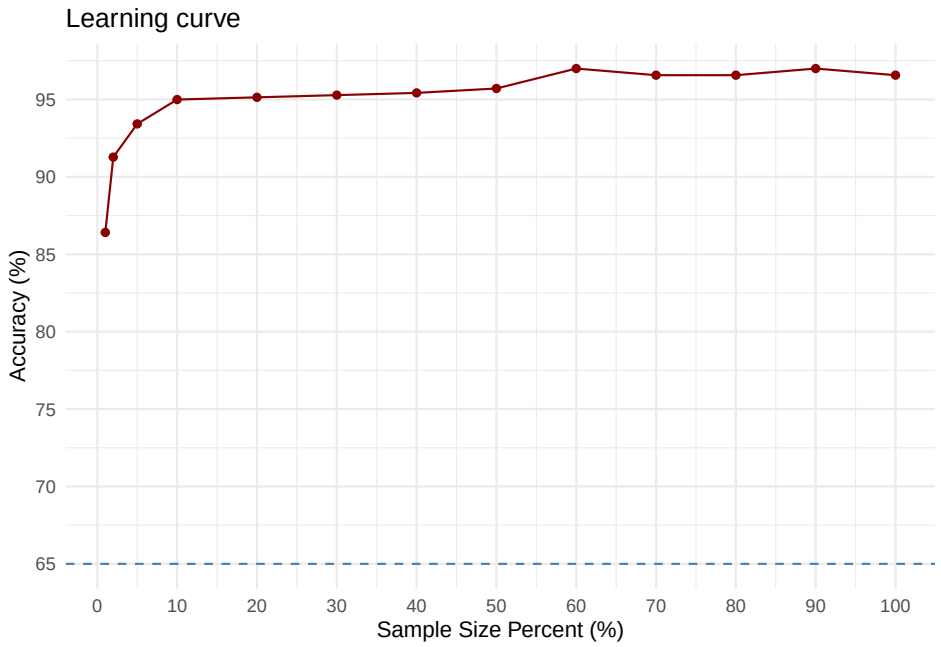


Figure 16: Learning curve of the accuracy for different sample size percentages compared to the baseline zeroR.

References

[1] Olvi L Mangasarian, W Nick Street, William H Wolberg: *Research article*, Breast Cancer Diagnosis and Prognosis via Linear Programming, December 19 1994, Retrieved from https://www.researchgate.net/publication/2302195 on 05-10-2021

[2] Neural Designer: *Machine Learning Examples*, Diagnose breast cancer from fine-needle aspirate images using Neural Designer, Retrieved from https://www.neuraldesigner.com/learning/examples/breast-cancer-diagnosis on 05-10-2021

[3] Kumar Sanjeev Priyanka 2021: *A Review Paper on Breast Cancer Detection Using Deep Learning*, IOP Conf. Ser.: Mater. Sci. Eng. 1022 012071, Retrieved from https://iopscience.iop.org/article/10.1088/1757-899X/1022/1/012071/pdf

[4] Oracle 2021: *machine learning*, What is machine learning?, Retrieved from https://www.oracle.com/nl/data-science/machine-learning/what-is-machine-learning/

[5] UCI Machine Learning Repository: *Breast Cancer Wisconsin (Original) Data Set*, Center for Machine Learning and Intelligent Systems