Thema 9 Log: Breast Cancer Wisconsin (Original) Data Set

Naomi Hindriks

9/21/2021

EDA : Breast Cancer Wisconsin (Original) Data Set

Data description

The data set: **Breast Cancer Wisconsin (Original) Data Set** is downloaded from the UCI machine learning repository. The data were collected by the University of Wisconsin Hospitals, Madison by Dr. William H. Wolberg.

The UCI website states that the data set contains 699 instances. According to the corresponding *breast-cancer-wisconsin.names* file (also downloaded from the UCI website) each instance is made up of 10 attributes, plus the class attribute. More detailed information of these attributes is shown in *Table 1*. The information found in Table 1 is a combination of information that was found in the *breast-cancer-wisconsin.names* file and User Manual Breast Cancer Diagnosis Web User Interface that includes an explanation on how to score the cytological characteristic.

According to the *breast-cancer-wisconsin.names* file there are 16 instances that contain a single missing attribute value, these are represented by "?" characters in the data file. It also states that out of the 699 instances there are 458 (65.5%) classified as benign and 241 (34.5%) classified as malignant.

To ensure the continued availability of the data and names files, they were copied to a personal repository.

```
attribute.info <- read.csv("data/attribute_info.csv", sep=";")

attribute.info.temp <- data.frame(
  "Column" = attribute.info$column,
  "Attribute" = attribute.info$full.name,
  "Unit" = attribute.info$unit,
  "Description" = attribute.info$description
  )

kbl(
    attribute.info.temp,
    row.names = F,
    caption = "Attribute Information. The cytological characteristics of breast FNAs (seen in rows 2-10)
    booktabs = T,
    linesep = "",
    longtable = T
  ) %>%
  kable_styling(latex_options = c("striped")) %>%
  column_spec(1:3, width = "1.5cm") %>%
  column_spec(4, width = "10cm")
```

Table 1: Attribute Information. The cytological characteristics of breast FNAs (seen in rows 2-10) get a score from 1 to 10 by an examining physician with 1 being the closest to benign and 10 the most anaplastic.

Column	Attribute	Unit	Description
1	Sample code number	id number	Unique number given to each sample
2	Clump Thickness	1-10	Assesses if cells are mono or multi-layered
3	Uniformity of Cell Size	1-10	Evaluate the consistency in size of the cells in the sample
4	Uniformity of Cell Shape	1-10	Evaluate the consistency in shape of the cells in the sample
5	Marginal Adhesion	1-10	Quantifies proportion of cells that stick together
6	Single Epithelial Cell Size	1-10	Measures the enlargement of epithelial cells size
7	Bare Nuclei	1-10	Proportion of nuclei surrounded by cytoplasm versus those that are not
8	Bland Chromatin	1-10	Rates the uniform texture of the nucleus in a range from fine to coarse
9	Normal Nucleoli	1-10	Determines whether the nucleoli are small and barely visible or larger, more visible, and more plentiful
10	Mitoses	1-10	Describes the level of mitotic activity
11	Class	2 or 4	Classification: 2 for benign and 4 for malignant

```
#clean up environment
remove(attribute.info.temp)
```

Data loading and prepping

```
data <- read.table(file = 'data/breast-cancer-wisconsin.data',
                                    header = F,
                                    sep = ",",
                                    na.strings = '?')

str(data)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ V1 : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561 1033078 1033078 ...
##  $ V2 : int  5 5 3 6 4 8 1 2 2 4 ...
##  $ V3 : int  1 4 1 8 1 10 1 1 1 2 ...
##  $ V4 : int  1 4 1 8 1 10 1 2 1 1 ...
##  $ V5 : int  1 5 1 1 3 8 1 1 1 1 ...
##  $ V6 : int  2 7 2 3 2 7 2 2 2 2 ...
```

```
##  $ V7 : int  1 10 2 4 1 10 10 1 1 1 ...
##  $ V8 : int  3 3 3 3 3 9 3 3 1 2 ...
##  $ V9 : int  1 2 1 7 1 7 1 1 1 1 ...
##  $ V10: int  1 1 1 1 1 1 1 1 5 1 ...
##  $ V11: int  2 2 2 2 2 4 2 2 2 2 ...
```

The data has been loaded, but it can be seen that the column names were not included in the data file. Furthermore the data does not seem to be of the correct data type, columns 2-10 should all be (ordered) factors. To give the columns the correct names and have easy access to the column descriptions I have created a simple csv file (data/attribute_info.csv).

```
names(data) <- attribute.info$name

data$class <- factor(data$class, levels = c(2, 4), labels = c("Benign", "Malignant"))

for(col.name in names(data)[2:10]) {
  data[, col.name] <- factor(data[, col.name], levels=1:10, ordered=T)
}

str(data)
```

```
## 'data.frame':    699 obs. of  11 variables:
##  $ id                    : int  1000025 1002945 1015425 1016277 1017023 1017122 1018099 1018561 1033
##  $ clump.thick           : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 5 5 3 6 4 8 1 2 2 4 ...
##  $ uni.cell.size         : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 1 1 2 ...
##  $ uni.cell.shape        : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 4 1 8 1 10 1 2 1 1 ...
##  $ marg.adhesion         : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 5 1 1 3 8 1 1 1 1 ...
##  $ single.epith.cell.size: Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 2 7 2 3 2 7 2 2 2 2 ...
##  $ bare.nuclei           : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 10 2 4 1 10 10 1 1 1 ...
##  $ bland.chrom           : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 3 3 3 3 3 9 3 3 1 2 ...
##  $ norm.nucleoli         : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 1 2 1 7 1 7 1 1 1 1 ...
##  $ mitoses               : Ord.factor w/ 10 levels "1"<"2"<"3"<"4"<..: 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 ...
```

```
#clean up environment
remove(col.name)
```

Now the columns have names and the values are of the correct data type.

Data verification

The original data description stated that 699 instances with 10 attributes + a class label are present.

```
dim(data)
```

```
## [1] 699  11
```

This checks out. The original data description also stated that there are 16 instances with a single missing value, the instances that have a mising value will be removed from the data set. This means there are no more than 16 missing values and the number of complete cases should be 699 - 16.

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

```
## [1] 16
```

```r
complete.instances <- complete.cases(data)

699 - sum(complete.instances)
```

```
## [1] 16
```

This is correct. According to the original data description the class distribution is as follows: benign: 458 (65.5%), malignant: 241 (34.5%).

```r
summary(data$class)
```

```
##    Benign Malignant
##       458       241
```

```r
format(summary(data$class) / nrow(data) * 100, digits = 3)
```

```
##    Benign Malignant
##    "65.5"    "34.5"
```

Again this checks out. The last thing that I am going to check are the *Sample code numbers* of instances. The data original description stated that this is an id number, therefore I assume all of these numbers should be unique.

```r
length(unique(data$id))
```

```
## [1] 645
```

The number of unique *sample code numbers* is 645, which is less than the 699 instances in the data. This is odd and requires further investigation. According to the original data description the data set is divided in 8 different groups, each group being collected in a different period of time. The groups 1 to 8 contain 367, 70, 31, 17, 48, 49, 31 and 86 instances respectively. Perhaps the *sample code numbers* of the instances are unique within their group.

```r
group.sizes <- c(367, 70, 31, 17, 48, 49, 31, 86)
duplicates.per.group <- c()

current.slice.start <- 0
i <- 0

for (group.size in group.sizes) {
  i <- i + 1
  group.row.numbers <- (current.slice.start + 1):(current.slice.start + group.size)
  current.slice.start <- current.slice.start + group.size

  duplicates <- sum(duplicated(data$id[group.row.numbers]))
  duplicates.per.group <- c(duplicates.per.group, duplicates)

  print(paste("Group ", i, ": ", duplicates , sep = ""))
}
```

```
## [1] "Group 1: 20"
## [1] "Group 2: 5"
## [1] "Group 3: 1"
## [1] "Group 4: 0"
## [1] "Group 5: 2"
## [1] "Group 6: 2"
## [1] "Group 7: 0"
## [1] "Group 8: 6"
```

```
# The total of duplicates when only looking inside group
sum(duplicates.per.group)
```

```
## [1] 36
```

```
# The total duplicates in and outside group
nrow(data) - length(unique(data$id))
```

```
## [1] 54
```

```
#clean up environment
remove(group.sizes, duplicates.per.group,
       current.slice.start, i, group.size,
       group.row.numbers, duplicates)
```

When looking at these numbers it is clear that there are duplicates within the groups and duplicates between different groups. This means that it is not logical that the duplicates are just duplicated rows that somehow got copied an extra time, because if that were the case we would expect to see only duplicates within groups. It is also not logical that the *sample code numbers* are reused in different groups since there are also duplicates within the groups. The next step is to check if the instances with duplicated *sample code numbers* have every attribute duplicated.

```
nrow(data[duplicated(data), ])
```

```
## [1] 8
```

There are 8 rows that are an exact copy of another row, this means that there are instances with the same *sample code number* but different values for the other attributes. Tables 2-47 show the instances that share their *sample code number* with at least one other instance. The tables that have duplicates where every attribute is the same have a red header.

```
duplicated.ids <- unique(data$id[duplicated(data$id)])

for (duplicate.id in duplicated.ids) {
  duplicate.entries.temp <- data[data$id == duplicate.id, ]

  if (sum(duplicated(duplicate.entries.temp)) > 0) {
    header.color <- "red"
  } else {
    header.color <- "white"
  }
```

```
  table <- kbl(
    duplicate.entries.temp,
    row.names = T,
    col.names = attribute.info$full.name,
    caption = paste("Instances with duplicate id:", duplicate.id),
    booktabs = T,
    linesep = ""
  ) %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position")) %>%
  column_spec(1:11, width = "1.5cm") %>%
  row_spec(0, background = header.color)

  print(table)
}
```

Table 2: Instances with duplicate id: 1033078

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
9	1033078	2	1	1	1	2	1	1	1	5	Benign
10	1033078	4	2	1	1	2	1	2	1	1	Benign

Table 3: Instances with duplicate id: 1070935

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
30	1070935	1	1	3	1	2	1	1	1	1	Benign
31	1070935	3	1	1	1	1	1	2	1	1	Benign

Table 4: Instances with duplicate id: 1143978

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
82	1143978	4	1	1	2	2	1	2	1	1	Benign
83	1143978	5	2	1	1	2	1	3	1	1	Benign

Table 5: Instances with duplicate id: 1171710

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
109	1171710	1	1	1	1	2	1	2	3	1	Benign
110	1171710	6	5	4	4	3	9	7	8	3	Malignant

Table 6: Instances with duplicate id: 1173347

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
116	1173347	1	1	1	1	2	5	1	1	1	Benign
117	1173347	8	3	3	1	2	2	3	2	1	Benign

Table 7: Instances with duplicate id: 1174057

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
121	1174057	1	1	2	2	2	1	3	1	1	Benign
122	1174057	4	2	1	1	2	2	3	1	1	Benign

Table 8: Instances with duplicate id: 1212422

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
195	1212422	3	1	1	1	2	1	3	1	1	Benign
196	1212422	4	1	1	1	2	1	3	1	1	Benign

Table 9: Instances with duplicate id: 1218860

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
208	1218860	1	1	1	1	1	1	3	1	1	Benign
209	1218860	1	1	1	1	1	1	3	1	1	Benign

Table 10: Instances with duplicate id: 1017023

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
5	1017023	4	1	1	3	2	1	3	1	1	Benign
253	1017023	6	3	3	5	3	10	3	5	3	Benign

Table 11: Instances with duplicate id: 1100524

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
43	1100524	6	10	10	2	8	10	7	3	3	Malignant
254	1100524	6	10	10	2	8	10	7	3	3	Malignant

Table 12: Instances with duplicate id: 1116116

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
63	1116116	9	10	10	1	10	8	3	3	1	Malignant
255	1116116	9	10	10	1	10	8	3	3	1	Malignant

Table 13: Instances with duplicate id: 1168736

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
105	1168736	10	10	10	10	10	1	8	8	8	Malignant
256	1168736	5	6	6	2	4	10	3	6	1	Malignant

Table 14: Instances with duplicate id: 1182404

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
137	1182404	4	1	1	1	2	1	2	1	1	Benign
257	1182404	3	1	1	1	2	1	1	1	1	Benign
258	1182404	3	1	1	1	2	1	2	1	1	Benign
266	1182404	5	1	4	1	2	1	3	2	1	Benign
449	1182404	1	1	1	1	1	1	1	1	1	Benign
498	1182404	4	2	1	1	2	1	1	1	1	Benign

Table 15: Instances with duplicate id: 1198641

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
169	1198641	3	1	1	1	2	1	3	1	1	Benign
259	1198641	3	1	1	1	2	1	3	1	1	Benign
267	1198641	10	10	6	3	3	10	4	3	2	Malignant

Table 16: Instances with duplicate id: 320675

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
268	320675	3	3	5	2	3	10	7	1	1	Malignant
273	320675	3	3	5	2	3	10	7	1	1	Malignant

Table 17: Instances with duplicate id: 733639

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
322	733639	3	1	1	1	2	NA	3	1	1	Benign
323	733639	3	1	1	1	2	1	3	1	1	Benign

Table 18: Instances with duplicate id: 704097

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
315	704097	1	1	1	1	1	1	2	1	1	Benign
339	704097	1	1	1	1	1	1	2	1	1	Benign

Table 19: Instances with duplicate id: 493452

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
372	493452	1	1	3	1	2	1	1	1	1	Benign
373	493452	4	1	2	1	2	1	2	1	1	Benign

Table 20: Instances with duplicate id: 560680

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
291	560680	1	1	1	1	2	1	1	1	1	Benign
375	560680	3	1	2	1	2	1	2	1	1	Benign

Table 21: Instances with duplicate id: 1114570

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
388	1114570	5	3	3	2	3	1	3	1	1	Benign
389	1114570	2	1	1	1	2	1	2	2	1	Benign

Table 22: Instances with duplicate id: 1158247

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
94	1158247	1	1	1	1	2	1	2	1	1	Benign
394	1158247	1	1	1	1	1	1	1	1	1	Benign

Table 23: Instances with duplicate id: 1276091

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
242	1276091	3	1	1	3	1	1	3	1	1	Benign
430	1276091	2	1	1	1	2	1	2	1	1	Benign
431	1276091	1	3	1	1	2	1	2	2	1	Benign
432	1276091	5	1	1	3	4	1	3	2	1	Benign
463	1276091	6	1	1	3	2	1	1	1	1	Benign

Table 24: Instances with duplicate id: 1293439

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
434	1293439	3	2	2	3	2	1	1	1	1	Benign
435	1293439	6	9	7	5	5	8	4	2	1	Benign

Table 25: Instances with duplicate id: 734111

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
443	734111	1	1	1	3	2	3	1	1	1	Benign
444	734111	1	1	1	1	2	2	1	1	1	Benign

Table 26: Instances with duplicate id: 1105524

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
48	1105524	1	1	1	1	2	1	2	1	1	Benign
469	1105524	4	1	1	1	2	1	1	1	1	Benign

Table 27: Instances with duplicate id: 1115293

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
62	1115293	1	1	1	1	2	2	2	1	1	Benign
491	1115293	1	1	1	1	2	1	1	1	1	Benign

Table 28: Instances with duplicate id: 1320077

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
517	1320077	1	1	1	1	1	1	1	1	1	Benign
518	1320077	1	1	1	1	1	1	2	1	1	Benign

Table 29: Instances with duplicate id: 769612

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
526	769612	3	1	1	2	2	1	1	1	1	Benign
527	769612	4	1	1	1	2	1	1	1	1	Benign

Table 30: Instances with duplicate id: 798429

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
338	798429	1	1	1	1	2	1	3	1	1	Benign
528	798429	4	1	1	1	2	1	3	1	1	Benign

Table 31: Instances with duplicate id: 1116192

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
65	1116192	1	1	1	1	2	1	2	1	1	Benign
538	1116192	5	1	2	1	2	1	3	1	1	Benign

Table 32: Instances with duplicate id: 1240603

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
548	1240603	2	1	1	1	1	1	1	1	1	Benign
549	1240603	3	1	1	1	1	1	1	1	1	Benign

Table 33: Instances with duplicate id: 1299924

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
512	1299924	5	1	1	1	2	1	2	1	1	Benign
553	1299924	3	2	2	2	2	1	4	2	1	Benign

Table 34: Instances with duplicate id: 1321942

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
561	1321942	5	1	1	1	2	1	3	1	1	Benign
562	1321942	5	1	1	1	2	1	3	1	1	Benign

Table 35: Instances with duplicate id: 385103

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
270	385103	1	1	1	1	2	1	3	1	1	Benign
576	385103	5	1	2	1	2	1	3	1	1	Benign

Table 36: Instances with duplicate id: 411453

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
272	411453	5	1	1	1	2	1	3	1	1	Benign
608	411453	1	1	1	1	2	1	1	1	1	Benign

Table 37: Instances with duplicate id: 822829

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
345	822829	7	6	4	8	10	10	9	5	3	Malignant
613	822829	8	10	10	10	6	10	10	10	10	Malignant

Table 38: Instances with duplicate id: 1061990

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
536	1061990	1	1	3	2	2	1	3	1	1	Benign
619	1061990	4	1	1	1	2	1	2	1	1	Benign

Table 39: Instances with duplicate id: 1238777

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
472	1238777	6	1	1	3	2	1	1	1	1	Benign
633	1238777	1	1	1	1	2	1	1	1	1	Benign

Table 40: Instances with duplicate id: 1277792

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
639	1277792	4	1	1	1	2	1	1	1	1	Benign
640	1277792	5	1	1	3	2	1	1	1	1	Benign

Table 41: Instances with duplicate id: 1299596

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
468	1299596	6	6	6	5	4	10	7	6	2	Malignant
645	1299596	2	1	1	1	2	1	1	1	1	Benign

Table 42: Instances with duplicate id: 1339781

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
661	1339781	1	1	1	1	2	1	2	1	1	Benign
662	1339781	4	1	1	1	2	1	3	1	1	Benign

Table 43: Instances with duplicate id: 1354840

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
673	1354840	2	1	1	1	2	1	3	1	1	Benign
674	1354840	5	3	2	1	3	1	1	1	1	Benign

Table 44: Instances with duplicate id: 466906

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
684	466906	1	1	1	1	2	1	1	1	1	Benign
685	466906	1	1	1	1	2	1	1	1	1	Benign

Table 45: Instances with duplicate id: 654546

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
690	654546	1	1	1	1	2	1	1	1	8	Benign
691	654546	1	1	1	3	2	1	1	1	1	Benign

Table 46: Instances with duplicate id: 695091

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
578	695091	1	1	1	1	2	1	2	1	1	Benign
692	695091	5	10	10	5	4	5	4	4	1	Malignant

Table 47: Instances with duplicate id: 897471

	Sample code number	Clump Thickness	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
698	897471	4	8	6	4	3	4	10	6	1	Malignant
699	897471	4	8	8	5	4	5	10	4	1	Malignant

```
#clean up environment
remove(duplicate.entries.temp, table, header.color, duplicated.ids, duplicate.id)
```

When inspecting these tables it becomes apparent that the duplicated data is sometimes in consecutive rows, but not always. It can also be observed that most of the instances with duplicated *sample code numbers* have the same class label, but not always. You can also see that most duplicates come in pairs, but they also come in bigger groups, up to 6 instances in one group (see Table 14). I do not see any pattern in how these rows are duplicated, nor can I think of any logical explanation for this. Since I do not want to risk using instances that are from the same person or sample I will keep only one instance per *sample code number*, and remove the duplicated rows.

Removing data

First the instances with a missing value will be removed. After that the instances with a duplicated *sample code number* will be removed.

```
# Keep unfiltered data in variable
unfiltered.data <- data

# only keep rows with complete instances
data <- data[complete.instances, ]

# verify 16 instances have been removed
dim(data)
```

```
## [1] 683  11
```

```
#clean up environment
remove(complete.instances)
```

After removing the instances with a missing value, there are 683 instances left, which is as expected because 699 - 16 = 683

```
# find instances with duplicated id
duplicates <- duplicated(data$id)

# remove duplicate instances from data
data <- data[!duplicates, ]

# Making the id the rowname and removing id column
#row.names(data) <- data$id
#data <- data[2:11]

# print what the dimensions are after cleaning the data
dim(data)
```

```
## [1] 630  11
```

Then after removing the instances with duplicated *sample code numbers* there are 630 instances left, these instances do not have missing values or duplicated *sample code numbers*.

Exploring variables

A first scan of the attributes.

```
summary(data)
```

```
##        id            clump.thick  uni.cell.size uni.cell.shape marg.adhesion
##  Min.   :   63375   1      :127   1      :339   1      :312    1      :355
##  1st Qu.:  877454   5      :118   10     : 62   10     : 54    2      : 54
##  Median : 1171998   3      : 96   3      : 47   2      : 52    10     : 54
##  Mean   : 1079385   4      : 68   2      : 40   3      : 51    3      : 53
##  3rd Qu.: 1238577   10     : 68   4      : 38   4      : 41    4      : 32
##  Max.   :13454352   2      : 47   5      : 29   5      : 31    8      : 25
##                     (Other):106   (Other): 75   (Other): 89    (Other): 57
##  single.epith.cell.size  bare.nuclei   bland.chrom  norm.nucleoli    mitoses
##  2      :343            1      :363   2      :149   1      :395   1      :515
##  3      : 66            10     :126   3      :145   10     : 59   2      : 34
##  4      : 44            3      : 28   1      :133   3      : 39   3      : 30
##  6      : 39            5      : 28   7      : 68   2      : 29   10     : 13
##  5      : 38            2      : 27   4      : 35   8      : 22   4      : 12
##  1      : 37            4      : 19   5      : 34   6      : 21   7      :  9
##  (Other): 63            (Other): 39   (Other): 66   (Other): 65   (Other): 17
##        class
##  Benign   :400
##  Malignant:230
##
##
##
##
##
```

For all attributes (except the class attribute) the most common value is 1 or 2. This makes sense, the most instances are classified as benign and lower numbers indicate more benign characteristics. Now I will make a table and bargraph to compare the class distribution before and after the filtering.

```
class.distribution <- data.frame(
  filtered = c(rep("before", nrow(unfiltered.data)), rep("after", nrow(data))),
  class = c(as.character(unfiltered.data$class), as.character(data$class)))


d1 <-  class.distribution %>% dplyr::group_by(filtered, class) %>%
  tally %>%
  bind_rows(class.distribution %>% dplyr::group_by(filtered) %>%
      tally %>%
      mutate(class="Total")) %>%
  mutate(pct = round(n/((sum(n)/2))*100)) %>%
  arrange(desc(filtered))
```

```
kbl(
  d1[,2:4],
  caption = "Data distribution before and after filtering, n is the number of instances and pct is the p
  kable_styling(latex_options = c("HOLD_position")) %>%
  pack_rows(index = table(fct_inorder(d1$filtered)))
```

Table 48: Data distribution before and after filtering, n is the number of instances and pct is the percentage of instances.

class	n	pct
before		
Benign	458	66
Malignant	241	34
Total	699	100
after		
Benign	400	63
Malignant	230	37
Total	630	100

```
ggplot(class.distribution, aes_string(x = "class", y = "..prop..")) +
  geom_bar(
    aes(fill = factor(filtered), group = -as.numeric(factor(filtered))),
      position = position_dodge()
      ) +
  geom_text(
    aes(label = ..count.., group = -as.numeric(factor(filtered))),
    stat = "count",
    position = position_dodge(width = 0.9),
    vjust = 2) +
  scale_y_continuous(labels=scales::percent) +
  scale_fill_manual(
      name = "Data set",
      values = c(hue_pal()(2)),
      breaks = c("before", "after"),
      labels = c("Data before filtering", "Data after filtering")) +
  labs(title="Class distribution before and after filtering of the data set") +
  xlab("Class") +
  ylab("Pecentage of data set")
```

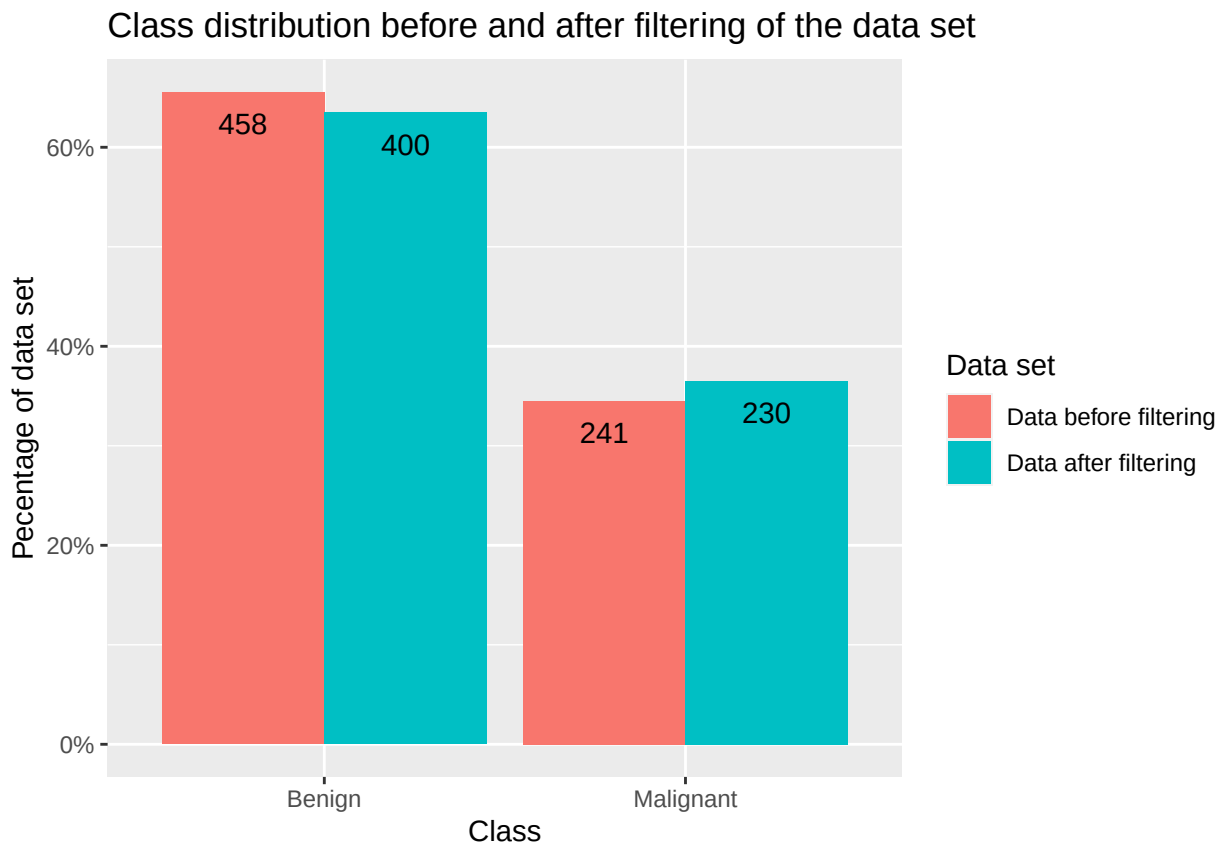

Figure 1: Distribution of the class attribute of the data before and after the filtering steps (the filtering steps are: removing rows with missing data, and than removing duplicated sample code numbers). The numbers in the bars are the actual number of instances in the data set.

When looking at *Table 48* and *Figure 1* you can see that the class distribution hasn't changed all that much from the filtering steps. This is a positive sign, if it had changed a lot the distribution after filtering might not have been representative anymore. Luckily that is not the case.

I will make bar plots to show the distribution of the cytological characteristic. I chose bar plots for this because the data is ordinal.

```
long.data <- pivot_longer(data, 2:10)

names.labs <- attribute.info$full.name
names(names.labs) <- attribute.info$name

ggplot(long.data, aes(x=value)) +
  geom_bar(aes(y = ..prop.., fill = name, group = class), stat="count") +
  labs(y = "Percent", fill="Attribute") +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_discrete(
    name = "Attribute",
    breaks = sort(attribute.info$name),
    labels = attribute.info$full.name[order(attribute.info$name)]
    ) +
  labs(
```

```
  title="Distribution of cytological characteristics scores",
  x ="Score on a scale from 1 to 10",
  y = "Percentage of instances"
  ) +
facet_grid(
  name ~ class,
  scales = "free",
  margin = "class",
  labeller = labeller(name = names.labs)
) +
theme(legend.position = "bottom", strip.text.y = element_blank())
```

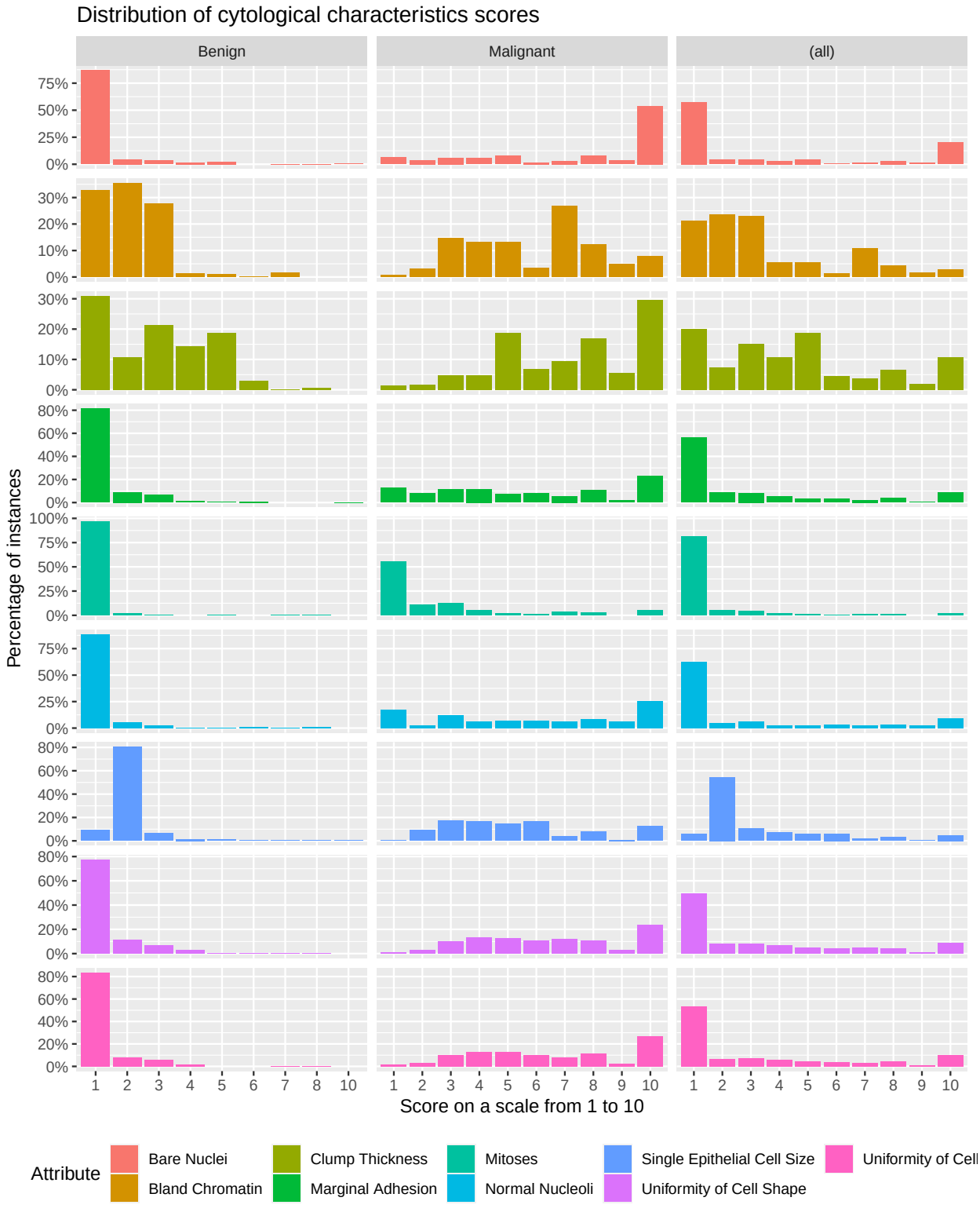

Figure 2: Distribution of data in percentage for 9 different cytological characteristics for benign instances, malignant instances and for all instances together

When looking at the distribution in *Figure 2* it seems there is a difference in distribution between the benign and malignant samples for every attribute. When looking at the overall distributions for the attributes it is important to keep in mind that the benign samples have a bigger influence on this because there are more instances with that class. The seemingly big difference in distributions for all of these attributes is a positive sign for machine learning, all of these attributes could be useful in differentiating benign and malignant samples from one another.

Now I will test if the differences observed in *Figure 2* are significant. I will use the Mann–Whitney U test for this as this is a non-parametric test that can be used for data that do not follow a normal distribution and ordinal data. For each attribute I will test if the malignant samples are significantly greater than the benign samples.

```
temp <- data.frame("attribute" = character(), "estimate" = double(), "pval" = double())

temp.data <- data

temp.data$class <- relevel(temp.data$class, ref = "Malignant")

for(attr in colnames(temp.data)[2:10]) {
  res <- wilcox.test(
    as.numeric(temp.data[,attr]) ~ temp.data$class,
    conf.int = TRUE,
    alternative = "greater"
  )

  full.name <- attribute.info$full.name[attribute.info$name == attr]
  temp <- rbind(temp, data.frame("attribute" = full.name,
                                 "estimate" = res$estimate,
                                 "pval" = res$p.value))
}

row.names(temp) <- NULL;

kbl(
  temp,
  col.names = c("Attribute name", "Estimate median of difference", "P value"),
  booktabs = T,
  digits = c(100),
  linesep = "",
  caption = "Results of one-sided Mann-Whitney U test for each attribute where the null hypothesis is tl
) %>%
  kable_styling(latex_options = c("HOLD_position", "striped"))
```

Table 49: Results of one-sided Mann–Whitney U test for each attribute where the null hypothesis is that the distribution of the malignant samples **is not** higher than that of the benign samples. And the alterernative hypothesis is that the distribution of the Malignant samples **is** higher than that of the benign samples. All of the p-values are well below 0.05 so we reject the null hypothesis for each attribute.

Attribute name	Estimate median of difference	P value
Clump Thickness	4.0000287732	1.736520e-68
Uniformity of Cell Size	5.0000208895	0.000000e+00
Uniformity of Cell Shape	5.0000340393	3.000000e-100
Marginal Adhesion	4.0000478861	1.197571e-77
Single Epithelial Cell Size	2.9999504915	1.339529e-84
Bare Nuclei	7.9999934047	8.290000e-98
Bland Chromatin	4.0000370221	3.375664e-79
Normal Nucleoli	4.9999897428	2.285017e-79
Mitoses	0.0000555406	1.003309e-39

```
remove(temp, temp.data)
```

In *Table 49* the results of one-sided Mann-Whitney U tests are displayed. Each row of the table represents a Mann-Whitney U tests on an attribute. The null hypothesis for each test is that the distribution of malignant samples **is not** shifted to the right compared to the distribution of the benign samples. The alternative hypothesis is that the distribution of malignant samples **is** shifted to the right compared to the distribution of the benign samples. If the p-value for one such test is lower than 0.05 the null hypothesis is rejected. Since all the p-values are well below 0.05 the null hypothesis is rejected in all the tests. The estimate median of difference is the median of the difference between all the instances in the two samples. The size of the estimate median of difference of the bare nuclei is the largest. This is logical when looking at *Figure 2* you can see that most of the benign samples have a score of 1 and most of the malignant samples have a score of 10. The estimate median of difference of the mitosis also stands out. This estimate is very small, close to zero even. This also makes sense when looking at the *Figure 2* as you can see that for both benign and malignant cases most instances have a score of 1, but in the malignant cases the data is spread out more. So even though the change is not big there is a significant difference in distribution of mitosis between benign and malignant samples.

Relation between attributes

Now I will look at the correlation between the different attributes. My expectation is that some or all of the attributes will be positively correlated because the scores of the attributes range from 1 to 10, with 1 being the closest to benign and 10 the most anaplastic. This means that if the sample is benign, all or most of the scores will be 1 or close to 1, and if the sample is malignant the scores will tend to be higher than 1. This idea is also supported by the fact that for all the attributes the distributions of the malignant samples were significantly shifted to the right compared to the benign distributions, they all shift in the same direction. For calculating the correlation score I will not use the Pearson method, as it makes the assumption that the data is normally distributed, in this case the data is not normally distributed, so I will have to find a different method. I have chosen to use Kendall τ_b rank correlation as seems to be better at handling ties.

```
df <- data

for(i in 2:10) {
  df[,i] <- as.numeric(df[,i])
}
colnames(df) <- attribute.info$full.name[-1]
```

```r
# Calculate  p values for correlation coefficients
correlation.p.values.kendall <- cor_pmat(df[,2:10], method = "kendall")

# Plot correlation coefficients for attributes
ggcorrplot(
  cor(df[,2:10], method = "kendall"),
  type = "lower",
  outline.col = "white",
  lab = TRUE,
  p.mat = correlation.p.values.kendall
) +
  ggtitle("Correlation between the attributes")
```

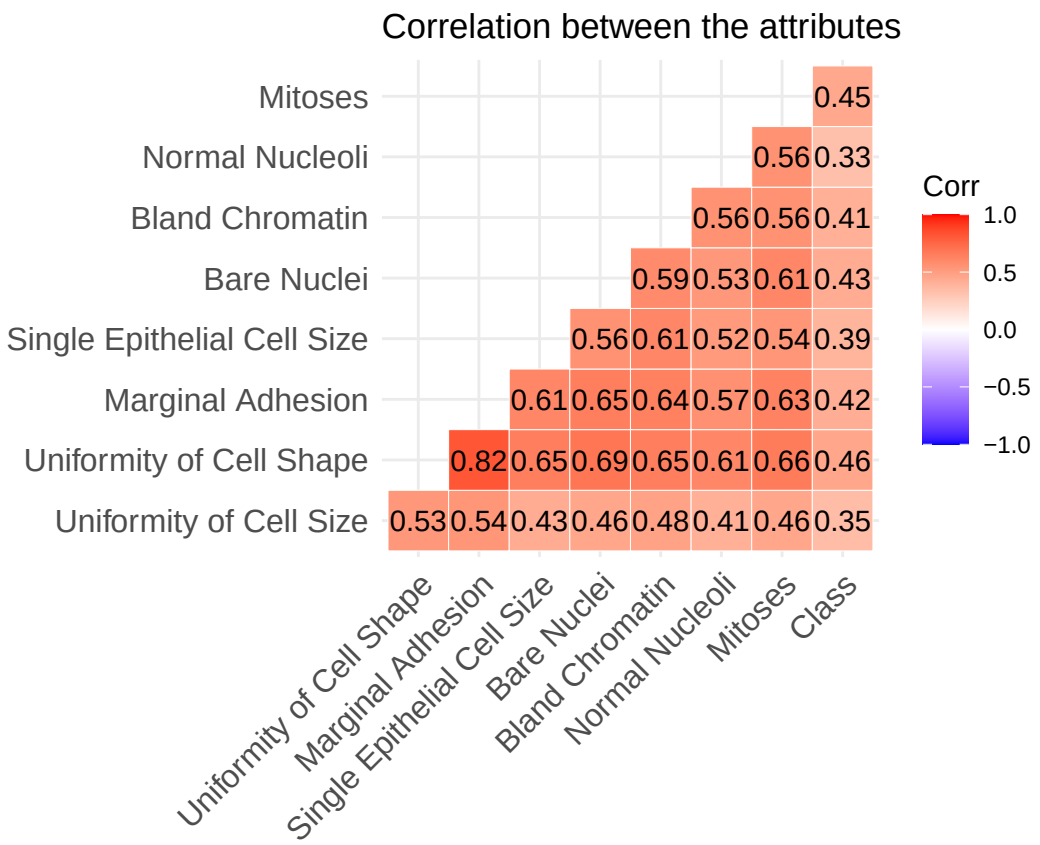

Figure 3: Correlations between the attributes

```r
# Print p values in table
kbl(
  correlation.p.values.kendall,
  booktabs = T,
  linesep = "",
  caption = "P values of Kendall $\\tau_{b}$ correlation coefficient",
  digits = 50,
  escape = F
) %>%
```

```
  column_spec(1:10, width = "3cm") %>%
  column_spec(1:10, width = "2cm") %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "scale_down"))
```

Table 50: P values of Kendall τ_b correlation coefficient

	Uniformity of Cell Size	Uniformity of Cell Shape	Marginal Adhesion	Single Epithelial Cell Size	Bare Nuclei	Bland Chromatin	Normal Nucleoli	Mitoses	Class
Uniformity of Cell Size	0.000000e+00	0.000000e+00	0.000000e+00	9.225525e-43	6.300000e-49	2.000000e-50	1.051763e-40	3.149500e-46	8.093515e-26
Uniformity of Cell Shape	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.999307e-39
Marginal Adhesion	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.799174e-34
Single Epithelial Cell Size	9.225525e-43	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	9.335989e-29
Bare Nuclei	6.300000e-49	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	7.454364e-35
Bland Chromatin	2.000000e-50	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.748064e-31
Normal Nucleoli	1.051763e-40	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	3.602932e-22
Mitoses	3.149500e-46	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	6.340816e-37
Class	8.093515e-26	4.999307e-39	6.799174e-34	9.335989e-29	7.454364e-35	6.748064e-31	3.602932e-22	6.340816e-37	0.000000e+00

When looking at the correlations between the variables in *Figure 3* it is clear that most of the attributes have a moderate to strong relationship. Clump thickness and mitosis seem to have a weak to moderate relationship to the other attributes. The attributes uniformity of cell size and uniformity of cell shape stand out as having a very strong relationship with each other (with a Kendall's τ_b correlation coefficient of 0.82). As expected all the correlations are positive. When looking at the p-values corresponding to the correlation coefficients in *Table 50* it can be seen that the p-values are all very small meaning that all the correlation coefficients in *Figure 3* are statistically significant. These correlation coefficients could impact specific classifier learning algorithms. One such example is the Naive Bayes classifier, as it assumes that the values of the attributes are independent of each other.

Principal component analysis

Next I will conduct principal component analysis, so we can see if two distinct groups can be seen based on the principal components. I have some doubts whether or not I should scale the data before performing the PCA. All of the attributes are on a scale from 1 to 10. Before making the decision I will check the variance of the different attributes.

```
vars <- apply(data[c(-ncol(data))], 2, var)
attr.names <- attribute.info[attribute.info$name %in% names(vars),][,c("full.name", "name")]

var.per.attr.df = data.frame(attr.name = attr.names$full.name, vars = vars[attr.names$name])

kbl(
  var.per.attr.df,
  caption = "Variance per attribute",
  row.names = F,
  col.names = c("Attribute", "Variance"),
  booktabs = T,
  linesep = "",
  digits = 3) %>%
```

```
  kable_styling(latex_options = c("striped", "HOLD_position")) %>%
  column_spec(1:2, width = "7cm")
```

Table 51: Variance per attribute

Attribute	Variance
Sample code number	4.11021e+11
Clump Thickness	8.19300e+00
Uniformity of Cell Size	9.44100e+00
Uniformity of Cell Shape	9.01700e+00
Marginal Adhesion	8.56900e+00
Single Epithelial Cell Size	5.08900e+00
Bare Nuclei	1.34630e+01
Bland Chromatin	6.10100e+00
Normal Nucleoli	9.73300e+00
Mitoses	3.10500e+00

The variance of the attributes is displayed in *Table 51*. The variance is not the same for the attributes. This means that when performing PCA that the attribute with the highest variance will have the most influence on the principal components, and the attributes with the lowest variance will only have a little influence on the principal components. The question remains whether the attributes values are on the same scale. All the attributes are on an ordinal scale from 1 to 10, in this manual you can find a more extensive explanation how these are scored. When reading the table on page 4 of this document it becomes clear that the attributes all are scored on a scale from 100% normal to 100% anaplastic, but the steps between the percentages differ slightly (most steps of all the attributes are 10% difference, but some include steps of 15%, and some have 1 step of 20%). It is also hard to say if 10% difference in mitosis means the same as 10% difference in clump thickness for example. Another problem for a correct PCA is that not all the steps are of the size, in an ideal PCA every step would have the same meaning.

Now I will do the analysis. I have chosen to do the analysis twice, once with scaling and once without scaling and compare the results with each other.

```
scaled_options = c(TRUE, FALSE)

pca.list = list()
plot.list = list()

for(scaled_option in  scaled_options) {
  # Get principal components
  pca.res <- prcomp(df[2:10], scale. = scaled_option, center = TRUE)
  pca.list[[paste("scaled.", scaled_option, sep = "")]] <- pca.res

  # Calculate explained variance
  var.explained.df <- data.frame(
    PC= paste0("PC",1:9),
    var.explained=(pca.res$sdev)^2/sum((pca.res$sdev)^2)
  )

  # Plot explained variance for PC's
  new.scree.plot <- ggplot(var.explained.df, aes(x=PC,y=var.explained, group=1))+
    geom_point(size=4)+
    geom_line()+
```

```
    ylab("Variance explained") +
    xlab("Principal component")

  # Get points to plot in PCA plot
  df.pca <- data.frame(pca.res$x, class=data$class)
  df.benign <- df.pca[df.pca$class == "Benign", ]
  df.malignant <- df.pca[df.pca$class == "Malignant", ]

  # PCA plot
  new.pca.plot <- ggplot(df.pca, aes(PC1, PC2, col=class)) +
    geom_point() +
    coord_cartesian(xlim = 1.2 * c(min(df.pca$PC1), max(df.pca$PC1)),
                    ylim = 1.2 * c(min(df.pca$PC2), max(df.pca$PC2))) +
    geom_encircle(data = df.benign) +
    geom_encircle(data = df.malignant) +
    xlab("Principal component 1") +
    ylab("Principal component 2") +
    theme(
      legend.direction = "horizontal",
      legend.background = element_rect(linetype = "solid", size = 1))

  leg <- get_legend(new.pca.plot)
  new.pca.plot <- new.pca.plot + theme(legend.position = "none")

  plot.list[[paste("scree.plot.scaled.", scaled_option, sep = "")]] <- new.scree.plot

  plot.list[[paste("pca.plot.scaled.", scaled_option, sep = "")]] <- new.pca.plot

}

# Add legend to plotlist
plot.list[["leg"]] <- leg

# Titles for rows and columns wrapped plot
row1 <- ggplot() +
  annotate(
    geom = 'text',
    x=1, y=1,
    label="Scaled = TRUE",
    angle = 90,
    size = 5,
    fontface = 2) +
  theme_void()
row2 <- ggplot() +
  annotate(
    geom = 'text',
    x=1, y=1,
    label="Scaled = FALSE",
    angle = 90,
    size = 5,
    fontface = 2) +
  theme_void()
col1 <- ggplot() +
```

```
  annotate(
    geom = 'text',
    x=1, y=1,
    label="Explained variance",
    size = 5.5,
    fontface = 2) +
  theme_void()
col2 <- ggplot() +
  annotate(
    geom = 'text',
    x=1, y=1,
    label="PCA plot",
    size = 5.5,
    fontface = 2) +
  theme_void()

title.list <- list(a = row1, b = row2, e = col1, f = col2)

layoutplot <- "
#ccccdddd
aeeeeffff
aeeeeffff
aeeeeffff
bgggghhhh
bgggghhhh
bgggghhhh
#####oooo
"

wrap_plots(plotlist = c(title.list, plot.list), guides = 'collect', design = layoutplot)
```

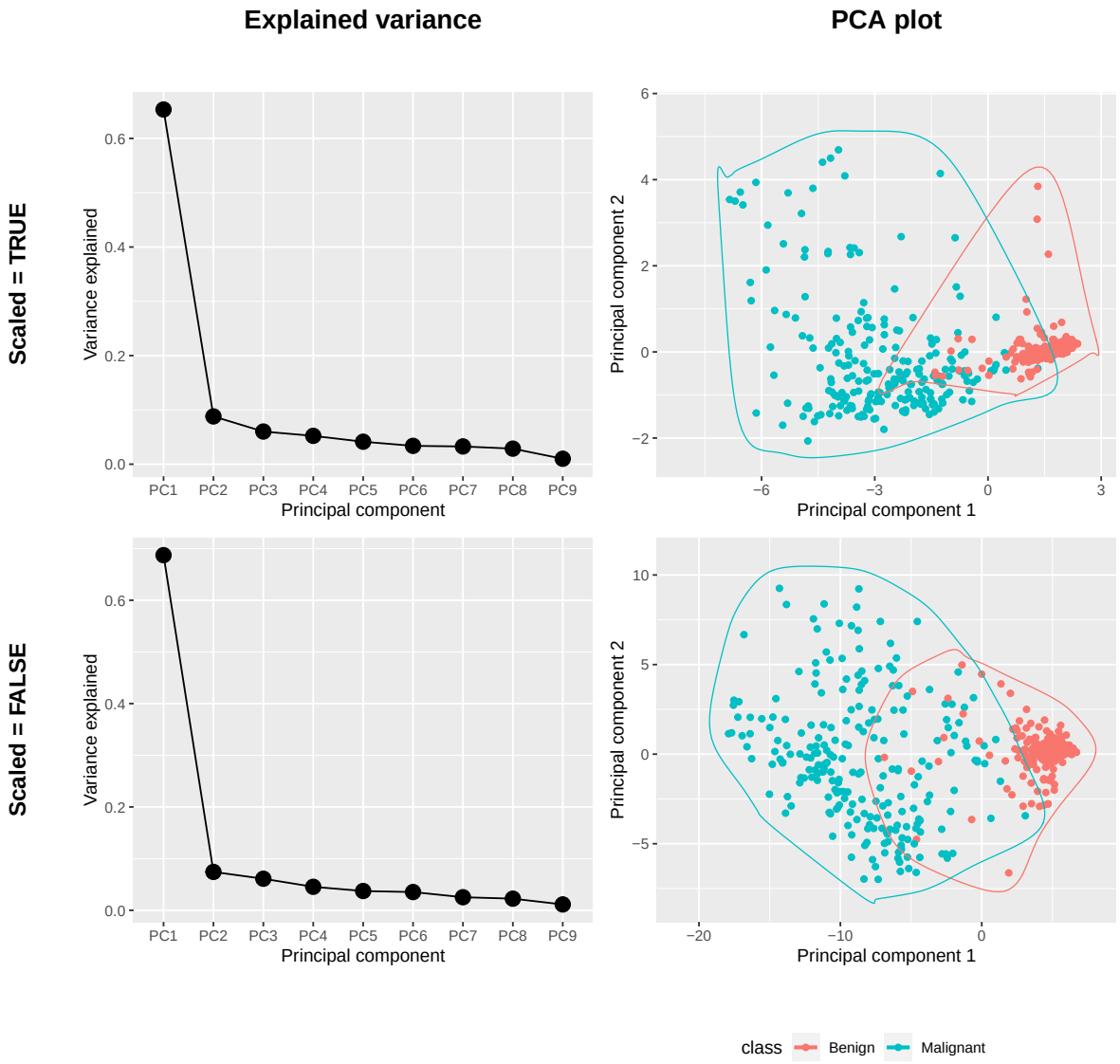

Figure 4: Principal component analysis

```
kbl(
    pca.list[["scaled.TRUE"]]$rotation,
    caption = "PCA: Rotation matrix of the 9 cytological characteristics to each principal component, s
    booktabs = T,
    linesep = ""
  ) %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position")) %>%
  column_spec(1:9, width = "2cm")
```

Table 52: PCA: Rotation matrix of the 9 cytological characteristics to each principal component, scaled = TRUE

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Uniformity of Cell Size	-0.3026067	-0.1277601	0.8730402	-0.0509633	0.0200015	0.2080380	0.0594103	0.2830144	-0.0018746
Uniformity of Cell Shape	-0.3821300	-0.0320382	-0.0386545	0.1856231	-0.1336961	0.2422388	-0.1310226	-0.4164032	-0.7415439
Marginal Adhesion	-0.3780364	-0.0701172	0.0287437	0.1593851	-0.0888896	0.1701730	-0.0535618	-0.5990616	0.6537113
Single Epithelial Cell Size	-0.3333947	-0.0734380	-0.3931759	-0.5003666	0.0309380	0.5636711	0.3050514	0.2529206	0.0528526
Bare Nuclei	-0.3360831	0.1847634	-0.1429394	0.3430707	-0.7025839	-0.1980931	0.1571675	0.3895393	0.0739862
Bland Chromatin	-0.3335249	-0.2678899	0.0273743	-0.5168253	-0.0492067	-0.6806530	0.1963920	-0.1930489	-0.0871259
Normal Nucleoli	-0.3465219	-0.2393446	-0.1911542	0.0159296	0.2005512	-0.1029457	-0.7827851	0.3403353	0.0802731
Mitoses	-0.3355534	0.0240361	-0.1277002	0.4830818	0.6411883	-0.1749484	0.4228656	0.1273752	-0.0195346
Class	-0.2268878	0.8992081	0.0828547	-0.2621999	0.1596186	-0.0873072	-0.1689347	-0.0510517	0.0093337

```
kbl(
    pca.list[["scaled.FALSE"]]$rotation,
    caption = "PCA: Rotation matrix of the 9 cytological characteristics to each principal component, s
    booktabs = T,
    linesep = ""
  ) %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position")) %>%
  column_spec(1:9, width = "2cm")
```

Table 53: PCA: Rotation matrix of the 9 cytological characteristics to each principal component, scaled = FALSE

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Uniformity of Cell Size	-0.2984508	-0.0683715	0.8550914	-0.1235371	0.3674030	-0.1086196	-0.0643288	-0.0941102	0.0056192
Uniformity of Cell Shape	-0.4011692	0.2316964	0.0016082	-0.2542141	-0.3547529	-0.1012452	0.1872058	0.1000193	0.7345765
Marginal Adhesion	-0.3903768	0.1644123	0.0625476	-0.1638881	-0.3824158	-0.0908336	0.4185489	0.1888991	-0.6514939
Single Epithelial Cell Size	-0.3361819	-0.1293465	-0.4820717	-0.4363429	0.6011305	-0.2200799	0.1288686	-0.1373377	-0.0469165
Bare Nuclei	-0.2501787	0.1993757	-0.0558834	-0.1757126	-0.2191651	0.4036996	-0.3528604	-0.7162597	-0.1229510
Bland Chromatin	-0.4388940	-0.7711234	-0.0686198	0.3265365	-0.1363697	0.2707260	0.0675639	0.0130594	0.0691472
Normal Nucleoli	-0.2920065	0.0021110	-0.1107320	0.0551576	-0.1433030	-0.4273832	-0.7704391	0.2996685	-0.1133753
Mitoses	-0.3590016	0.4785523	-0.1097836	0.7218032	0.2996154	0.0001368	0.1198952	-0.0650426	0.0260471
Class	-0.1233318	0.1856918	-0.0191080	-0.2032637	0.2337317	0.7087615	-0.1788822	0.5651236	-0.0170201

Now it is time for the fun part: interpreting the results of the PCA's. In the PCA plots in *Figure 4* it can be seen that with or without scaling, the benign and malignant samples clearly form two distinct clusters, this is a positive sign for making a classifier for this data. The in the unscaled plot the clusters might seem a bit more distinct than in the scaled plot.

In the scaled PCA plot (top right) the variance in the second principal component seems to represent a small amount of instances. This second component does seem to say something about malignancy, you can see that only 3 benign instances are higher than 2, a lot more malignant instances are in this area (even though benign is the majority class). Looking at *Table 52* we can observe that the second component is dominated by the mitosis attribute. Comparing the second principle component to the distribution of mitosis in *Figure*

2 the distribution of the second component makes sense, most of the instances have a mitosis of 1, but from the instances that have a mitosis score higher than 1 most of them are malignant.

In the unscaled PCA plot (bottom right) principal component 1 seems to have a dividing line between benign and malignant at around -2.5. On principal component 2 the benign instances seem to be concentrated around 0 and the malignant instances are more dispersed.

Comparing the explained variance plots in *Figure 4* you can see that they are quite the same. In both cases the first component explains by far most of the variance. In the unscaled PCA the first component explains a little more of the variance than the scaled version and the second component a little less. The difference in these plots doesn't seem significant. It is important to keep in mind that in the unscaled version the different attribute do not contribute equally to the total variance.

When comparing *Table 52* and *Table 53* you can see that in *Table 52*, the scaled PCA, the attributes contribute quite equally to principal component 1, the smallest contributor to component 1 is mitosis, but mitosis does dominate the second component. This is explained by mitosis having a smaller relationship to all the other attributes, as can be seen in the correlation plot (the greater the correlation between two attributes, the more they can be displayed on the same axis). In *Table 53*, the unscaled PCA, the attribute with the biggest variance (bare nuclei) contributes most to the first as well as the second principal component and mitosis does not play a big role in either of the two.

Keeping the correlations between the attributes in mind, and the fact that all the attributes contribute more equally I think that the PCA plot with scaling is the better visualization of the multidimensional data, but the unscaled plot seems to have a better division between benign and malignant instances, this could be caused by the fact that the attribute with the biggest variance is also the best predictor of malignancy. However this is not the core purpose of PCA. If we want to find the axis that represents the biggest separation in class we should do *linear discriminant analysis*.

Machine Learning Phase

To do the machine learning with the data I will use the free machine learning software Weka (Waikato Environment for Knowledge Analysis). To load the data in the Weka environment I will save the cleaned and filtered data to a csv file. Before I do this I will change the cytological characteristic attributes to numeric, because it is ordered data, and if it is saved as a factor Weka will treat is as nominal data without order.

```
data.to.save <- data[-1]
data.to.save$class <- factor(data.to.save$class, levels = c("Malignant", "Benign"))

for (i in 1:9) {
  data.to.save[, i]  <- as.numeric(data.to.save[, i])
}

write.arff(data.to.save, file= "data/filtered_data.arff", relation = "Filtered Breast Cancer Wisconsin

remove(data.to.save)
```

Relevant quality metrics

In *Table 54* the confusion matrix for the Breast Cancer Wisconsin (Original) Data Set can be seen.

Accuracy

There are different metrics for evaluating the performance of machine learning classifier using the confusion matrix. In machine learning accuracy (ACC) is the default quality metric. ACC is the fraction of predictions that were classified correctly, it is calculated like this: $\frac{TP+TN}{TP+TN+FP+FN}$ In the case of the Breast Cancer Wisconsin (Original) Data Set this means $\frac{\text{Malignant instances classified as malignant + Benign instances classified as benign}}{\text{Total instances}}$.

```
df = data.frame(
  actual.positive = c(
    "Malignant sample predicted as malignant\n(True Positive, TP)",
    "Malignant sample predicted as benign\n(False Negative, FN)"),
  actual.negative = c(
    "Benign sample predicted as malignant\n(False Positive, FP)",
    "Benign sample predicted as benign\n(True Negative, TN)")
)
row.names(df) <- c("Malignant", "Benign")

df %>%
mutate_all(linebreak) %>%
kbl(
  format = "latex",
  col.names = c("Malignant", "Benign"),
  row.names = TRUE,
  booktabs = T,
  caption = "Meaning of confusion matrix in case of the Breast Cancer Wisconsin (Original) Data Set",
  escape = F
) %>%
  add_header_above(c(" " = 1, "Actual class" = 2)) %>%
  pack_rows("Predicted value", 1, 2) %>%
```

```
kable_styling(latex_options = c("striped", "scale_down", "HOLD_position")) %>%
column_spec(1:3, width = "5cm")
```

Table 54: Meaning of confusion matrix in case of the Breast Cancer Wisconsin (Original) Data Set

	Actual class	
	Malignant	Benign
Predicted value		
Malignant	Malignant sample predicted as malignant (True Positive, TP)	Benign sample predicted as malignant (False Positive, FP)
Benign	Malignant sample predicted as benign (False Negative, FN)	Benign sample predicted as benign (True Negative, TN)

ACC has its shortcomings, it does tell us something about the amount of correctly classified instances, but not whether those correctly classified instances are true positive (TP) or true negative (TN). Likewise the error rate (ER), 1 - ACC, tells us the fraction of incorrectly classified instances, but does not tell us anything about which instances are most likely to be classified incorrectly. In the case of classifying breast cancer, the proportion of samples that are FP and FN is quite important. When a sample from a patient is classified as benign while it should be malignant, a false negative, this patient is sent home thinking they do not have cancer, not getting further examination and treatment. It goes without saying that the consequences of this prediction could be disastrous, and could even result in premature death of this patient. On the other hand, if a patient is classified as malignant while being benign, false positive, the further examination that this patient will undergo will reveal their true class. That being said making this mistake is still undesirable: it can cause the patient a lot of distress, and the further examination that is performed is costly for the health care system.

Sensitivity and specificity

Two metrics that give insight in the amount of correctly classified instances within a class are the sensitivity and specificity. The sensitivity, also called true positive rate (TPR), can be calculated like so: $\frac{TP}{TP+FN}$. In our case a high sensitivity means that from the samples that are malignant most are classified as malignant. The specificity, also called the true negative rate (TNR), also has an equation: $\frac{TN}{TN+FP}$. A high specificity means that from the samples that are benign, most are also classified as benign.

As explained in the previous paragraph the proportion of TP to FN is more important to us than TN to FP, so the sensitivity is more important to us than the specificity, but we can never look at the sensitivity alone. When looking at the sensitivity on their own, it it not as informative as it might seem, it is very easy to get a sensitivity of 1 by classifying all instances as positive. The same is true for specificity when classifying all instances as negative. The goal is to get a sensitivity as close to 1 as possible while not letting the specificity (and accuracy) drop too low.

Another shortcoming of ACC is that it is sensitive to imbalanced data, this shortcoming can be dissolved using the sensitivity and specificity. First lets illustrate this problem with an example: imagine having a data set with 95 positive instances and 5 negative instances. Classifying every instance in this data set results in a high ACC of 0.95, but all of the negative instances are incorrectly classified. If done the other way around, classifying every instance as negative, the ACC would only be 0.05, even though the algorithm might be just as bad. The solution to this problem is the balanced ACC (bACC), the bACC score is calculated as follows: $\frac{\text{sensitivity} + \text{specificity}}{2}$. In the example the bACC of both classifiers would be 0.5 ($\frac{1+0}{2}$). Since the Breast Cancer Wisconsin (Original) Data Set is not as imbalanced, it doen't seem neccasary to use the bACC instead of the ACC

The ROC curve and AUC measure

A useful way of visualizing the trade off between sensitivity and specificity is with the receiver operating characteristic (ROC) curve, see *Figure 5*. In the ROC curve is the sensitivity as function of $1 - specificity$. In an ROC curve the coordinate (0, 0) represents the place where the sensitivity is 0 and the specificity is 1, so classifying every instance as negative. The point (1, 1) is the place where the sensitivity is 1 and the specificity is 0, so classifying every instance as positive. A perfect algorithm would go through the point (0, 1) where the sensitivity is 1 and specificity is also 1. Moving along the ROC curve you are moving the threshold you set for an algorithm (an example of a threshold is classifying an instance as benign only when you are 90% sure the class is benign). A metric calculated with the ROC curve is the area under the curve (AUC). The AUC is, as the name suggests, a metric that has the value of the area under the curve of the ROC curve of an algorithm. A problem with the AUC is that it ignores the trade off between the sensitivity and specificity. AUC provides an aggregate measure of performance across all possible classification thresholds, but from the AUC you cannot see at which thresholds the algorithm performs best. We want to select the algorithm which performs best at the threshold that gives us a high sensitivity, and that can't be seen in the AUC. But the AUC is still a relatively good measure in determining the overall performance of an algorithm.

```
knitr::include_graphics("ROC-curve.png")
```

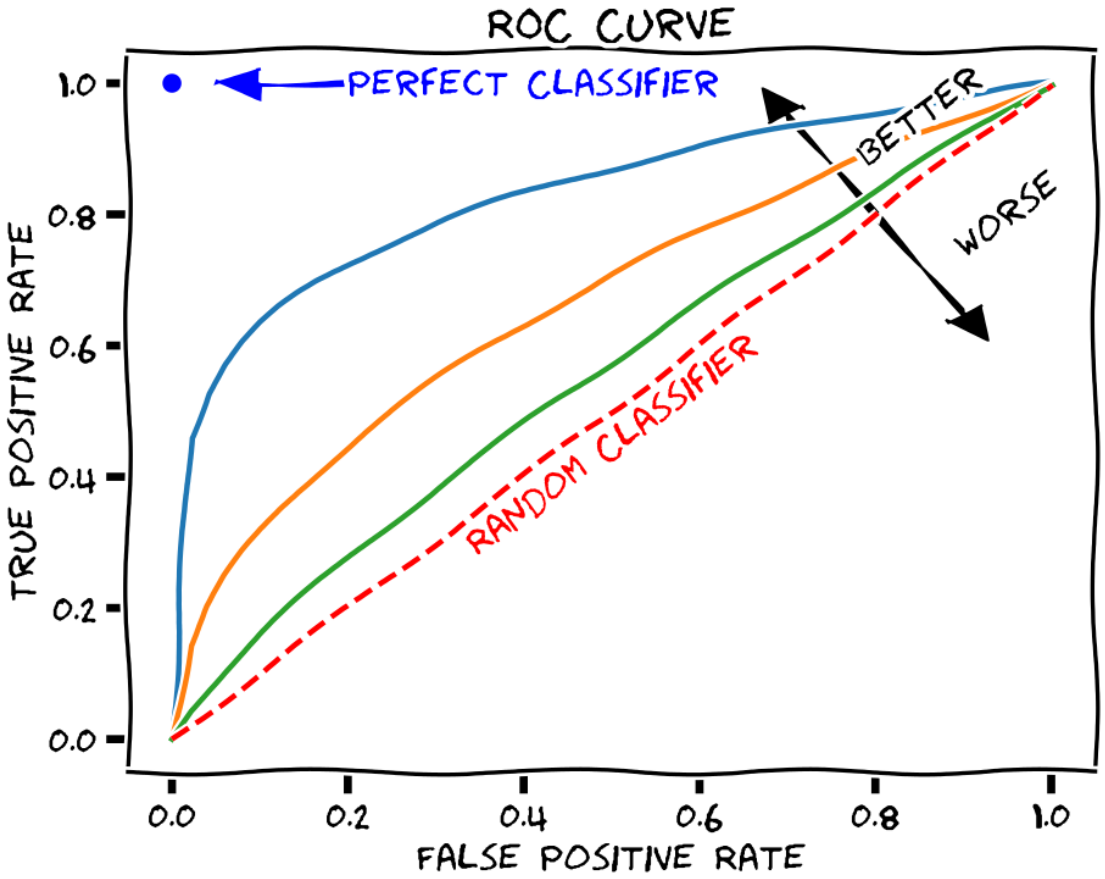

Figure 5: Receiver Operating Characteristic (ROC) curve with False Positive Rate and True Positive Rate. The diagonal shows the performance of a random classifier. Three example classifiers (blue, orange, green) are shown. Source: MartinThoma, CC0, via Wikimedia Commons. A side note for this image is that a line worse than random classification is not useless, you only have to turn the classification around and the curve will be better than the random classification.

Precision and F measure

A measure that might be useful in our case is the F score. The F score combines the precision and sensitivity (sometimes referred to as recall). Precision is the amount of correctly classified instances of all the positively classified instances. This is calculated as follows $\frac{TP}{TP+FP}$. So if we maximize the precision we minimize the false positives. Since the F score depends on the precision as well as the recall it maximizes the amount of true positives while minimizing the false positives. The equation for the default F score, called the F_1 measure, is show in the equation below (1). In this F_1 measure the precision and sensitivity are equally weighted. To give β times more importance to recall over precision the F_β score can be used as shown below (2). Two common measures using the F score are the F_2 and $F_{0.5}$ scores where recall is double as important as precision and recall is half as important as precision respectively. In our case we would want to give more importance to recall, since recall is a measure for the proportion of correctly classified malignant instances and this is very important in this case (as described in the sensitivity and specificity paragraph).

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{1}$$

$$F_\beta = 1 + \beta^2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2}$$

It is important to note that the F measure is not perfect, it has it's downsides as well. An important downside is that it only focuses on one class, in our case the malignant class, it completely ignores the amount of TN, in our case real benign samples that are classified as such. The F-measure is biased by the majority class (just like accuracy, recall and precision).

Choosing a quality metric

All the quality metrics described above have their shortcomings. For every algorithm I will report the accuracy, confusion matrix (TP, FP, FN and TN) as well as the the specificity, sensitivity, AUC and F_2 score.

Testing algorithms

Algorithms to test

Now it is time to see how different machine learning algorithms perform on this data. First the ZeroR and OneR algorithms will be performed to establish a baseline. I will compare the ZeroR and OneR resutls with the following algorithms: Naive Bayes, Simple Logistic, SVM (SMO), Nearest Neighbor (IBk), J48 Decision Tree and Random Forest.

As I have already mentioned above the cost of wrongfully classifying a malignant sample as benign is far worse than the other way around, therefore I will try every algorithm with different classification costs. I will test all the algorithms with no specific cost set and costs set to 2:1, 10:1, 50:1 and 100:1 for FN:FP.

The algorithms are tested using the *Weka (version 3.8.4) Experimenter*. For every test 10-fold cross validation is used, and 10 repetitions are run, to get a more reliable result. To set a cost for an algorithm the CostSensitiveClassifier is used with the costMatrix set to the correct cost. The field minimizeExpectedCost is set to true, this means the cost will be calculated after the model is build and then used to set a threshold to classify the instances (as opposed to giving a weight to every instance before building the model, which is what happens when this field is set to false). For IB5 IBK is used with the K field set to 5 and for IB10 the K field is set to 10. All the other parameters are set to the default value. The experiment configuration can be found in the *Classification_options_experiment.exp* file found in the data folder of this repository. To rerun the experiment download the *Classification_options_experiment.exp* and *filtered_data.arff* file from the data

folder in the repository, open the Weka Experimenter and open the *Classification_options_experiment.exp*, and add the *filtered_data.arff* to the datasets panel, now run the experiment.

First results

The results of the performed experiments are shown in *Table 55*. It can be seen that the accuracy for all algorithms except ZeroR is pretty high, for almost all algorithms the accuracy is above 90%. Even for OneR, which is based on a single attribute in the data, the accuracy is above 90%. For multiple algorithms (Simple Logistic, Nearest Neighbor (IB5), Nearest Neighbor (IB10), J48 Decision Tree and Random Forest) the best accuracy is not achieved by using a cost of 1:1, but by a cost of 2:1. This might be because the natural class distribution of a data set does not always give the best classifier, by changing the cost this might compensate for the natural class imbalance. Although the differences in accuracy observed between a cost of 1:1 and 2:1 might not be significant. Further statistical test would be necessary to check the significance.

Nearest Neighbor (IB5) had the best accuracy score (97.635 %) and after that Random Forest had the best accuracy score (97.508 %), followed by the Nearest Neighbor (IB10) algorithm (97.238 %) and Simple Logistic algorithm (97.175 %).

Aside from ZeroR and OneR the J48 tree has the lowest accuracy (95.063 %), second worst accuracy is IB1 (96.365 %) and Naive Bayes (96.429 %), but as you can see the difference between the highest and lowest scoring algorithm is only 2.572 %. It might not seem like a big difference, but might very well be a significant one.

When looking at the AUC score you can see that a weird thing is going on. The AUC score should be the same for every cost, since changing the cost only changes the threshold, and AUC is a measure that comprises every threshold. In our case the 1:1 cost always performs the best or equal to the others costs. The reason that this happened seems to be that Weka makes the ROC curve for the cost sensitive classifier by only plotting the threshold instance. So when comparing the AUC scores of the different algorithms I will only take the AUC score of the 1:1 cost into account.

The F_2 score on the other hand performs best at a cost of 2:1 or higher, which reflects the importance that is given to recall in this score.

The algorithm that performed best according to the AUC is the Simple Logistic algorithm (with a score of 0.996). Closely followed by the IB10, Random Forest and IB5 algorithm (with a score of 0.995, 0.994 and 0.993 respectively). The worst AUC score goes to the J48 tree (0.950) followed by IB1 (0.959) and SVM (0.969), again apart from ZeroR and OneR.

When looking at the F_2 score the algorithms that performed best are IB5 and Random Forest (both with a score of 0.982). folowed by IB10 (0.980) and Simple Logistic (0.978). The lowest F_2 score, not looking at ZeroR and OneR, belongs to the J48 tree (0.937), followed by the IB1 (0.945) and SVM (0.962) algorithms.

This means that J48 performs worst when looking at the accuracy, AUC and F_2 scores. Which probably indicates that J48 is not a great algorithm for this data set. IB1 also scores poorly for all these scores. And SVM ended in the bottom three for F_2 as well as AUC score.

The IBk (IB5 and IB10) both seem to get pretty high scores for accuracy, F_2 and AUC scores. The Random Forest algorithm also ended in the top 3 for these three scores. And the Simple Logistic had the best AUC score and ended in the fourth place for the F_2 and the accuracy score.

Naive Bayes ended somewhere in the middle for all the scores. But it would be interesting to see how Naive Bayes performs when we change the *useSupervisedDiscretization* setting, since the data is not normally distributed, this might give a much better result.

```
weka.experiment <- read.arff("data/weka_experiment.arff")

algorithms <- c("ZeroR", "OneR", "Naive Bayes", "Simple Logistic", "SVM (SMO)",
                "Nearest Neighbor (IB1)", "Nearest Neighbor (IB5)",
```

```
                "Nearest Neighbor (IB10)", "J48 Decision Tree", "Random Forest")
weka.experiment$Algorithm <- factor(rep(algorithms, each = 500), levels = algorithms)

costs <- c("1:1", "2:1", "10:1", "50:1", "100:1")
weka.experiment$Cost <- factor(rep(costs, each = 100, times = 10), levels = costs)

f.2.score <- function (TP, FP, FN) {
  recall <- mean(TP / (TP + FN))
  precision <- mean(TP / (TP + FP))
  score <- 5 * ((precision * recall) / (4 * precision + recall))
  return(score)
}

by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm, Cost) %>%
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Cost = first(Cost),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm, Cost) %>%
  summarize(
    .groups = "keep",
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 <- format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized, select = -Algorithm),
  row.names = F,
```

```
  col.names = c("Cost", "Training", "Testing",
                " ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of different machine learning algorithms us
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  pack_rows(index = table(summarized$Algorithm)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "1.2cm") %>%
  column_spec(c(2:4, 9:12), width = "1cm") %>%
  footnote(general = c("Cost = the cost of FN:FP",  "ACC = accuracy (\\\\%)", "TP = true positive", "FP
```

Table 55: Results of experiment run in Weka Experimenter of different machine learning algorithms used on the filtered Breast Cancer Wisconsin (Original) Data Set. All the algorithms were tested using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Cost	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
ZeroR											
1:1	0.0025	0.0011	63.492	0.00	0.00	400.00	230.00	0.000	1.000	0.500	NaN
2:1	0.0027	0.0012	36.508	230.00	400.00	0.00	0.00	1.000	0.000	0.500	0.742
10:1	0.0019	0.0004	36.508	230.00	400.00	0.00	0.00	1.000	0.000	0.500	0.742
50:1	0.0014	0.0003	36.508	230.00	400.00	0.00	0.00	1.000	0.000	0.500	0.742
100:1	0.0005	0.0001	36.508	230.00	400.00	0.00	0.00	1.000	0.000	0.500	0.742
OneR											
1:1	0.0078	0.0004	91.857	209.60	30.90	369.10	20.40	0.911	0.923	0.917	0.903
2:1	0.0073	0.0001	91.857	209.60	30.90	369.10	20.40	0.911	0.923	0.917	0.903
10:1	0.0053	0.0002	91.857	209.60	30.90	369.10	20.40	0.911	0.923	0.917	0.903
50:1	0.0047	0.0001	91.857	209.60	30.90	369.10	20.40	0.911	0.923	0.917	0.903
100:1	0.0067	0.0002	91.857	209.60	30.90	369.10	20.40	0.911	0.923	0.917	0.903
Naive Bayes											
1:1	0.0067	0.0026	96.429	224.90	17.40	382.60	5.10	0.978	0.957	0.986	0.967
2:1	0.0040	0.0025	96.413	225.20	17.80	382.20	4.80	0.979	0.956	0.967	0.968
10:1	0.0042	0.0031	96.365	226.10	19.00	381.00	3.90	0.983	0.953	0.968	0.970
50:1	0.0038	0.0017	96.444	227.80	20.20	379.80	2.20	0.990	0.950	0.970	0.975
100:1	0.0045	0.0014	96.302	228.00	21.30	378.70	2.00	0.991	0.947	0.969	0.975
Simple Logistic											
1:1	0.2526	0.0004	96.762	217.80	8.20	391.80	12.20	0.947	0.980	0.996	0.950
2:1	0.2243	0.0007	97.175	222.40	10.20	389.80	7.60	0.967	0.975	0.971	0.965
10:1	0.2204	0.0003	96.397	228.80	21.50	378.50	1.20	0.995	0.946	0.971	0.978
50:1	0.2245	0.0003	87.476	230.00	78.90	321.10	0.00	1.000	0.803	0.901	0.936
100:1	0.2256	0.0004	76.492	230.00	148.10	251.90	0.00	1.000	0.630	0.815	0.886
SVM (SMO)											
1:1	0.0440	0.0008	97.048	221.60	10.20	389.80	8.40	0.963	0.975	0.969	0.962
2:1	0.0251	0.0011	97.048	221.60	10.20	389.80	8.40	0.963	0.975	0.969	0.962
10:1	0.0210	0.0007	97.048	221.60	10.20	389.80	8.40	0.963	0.975	0.969	0.962
50:1	0.0226	0.0009	97.048	221.60	10.20	389.80	8.40	0.963	0.975	0.969	0.962

Table 55: Results of experiment run in Weka Experimenter of different machine learning algorithms used on the filtered Breast Cancer Wisconsin (Original) Data Set. All the algorithms were tested using 10-fold cross validation and the iteration was set to 10 repetitions. *(continued)*

	Time			Confusion matrix							
Cost	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
100:1	0.0204	0.0003	97.048	221.60	10.20	389.80	8.40	0.963	0.975	0.969	0.962
Nearest Neighbor (IB1)											
1:1	0.0008	0.0229	96.365	216.50	9.40	390.60	13.50	0.941	0.977	0.959	0.945
2:1	0.0006	0.0212	96.365	216.50	9.40	390.60	13.50	0.941	0.977	0.959	0.945
10:1	0.0005	0.0215	96.365	216.50	9.40	390.60	13.50	0.941	0.977	0.959	0.945
50:1	0.0007	0.0211	96.365	216.50	9.40	390.60	13.50	0.941	0.977	0.959	0.945
100:1	0.0014	0.0203	96.365	216.50	9.40	390.60	13.50	0.941	0.977	0.959	0.945
Nearest Neighbor (IB5)											
1:1	0.0002	0.0254	97.540	222.90	8.40	391.60	7.10	0.969	0.979	0.993	0.968
2:1	0.0005	0.0164	97.635	227.50	12.40	387.60	2.50	0.989	0.969	0.979	0.981
10:1	0.0009	0.0160	97.222	229.00	16.50	383.50	1.00	0.996	0.959	0.977	0.982
50:1	0.0007	0.0150	97.222	229.00	16.50	383.50	1.00	0.996	0.959	0.977	0.982
100:1	0.0004	0.0168	97.222	229.00	16.50	383.50	1.00	0.996	0.959	0.977	0.982
Nearest Neighbor (IB10)											
1:1	0.0006	0.0199	97.175	221.30	9.10	390.90	8.70	0.962	0.977	0.995	0.962
2:1	0.0006	0.0180	97.238	225.00	12.40	387.60	5.00	0.978	0.969	0.974	0.972
10:1	0.0004	0.0188	96.778	228.90	19.20	380.80	1.10	0.995	0.952	0.974	0.980
50:1	0.0003	0.0177	96.619	229.00	20.30	379.70	1.00	0.996	0.949	0.972	0.979
100:1	0.0004	0.0177	96.619	229.00	20.30	379.70	1.00	0.996	0.949	0.972	0.979
J48 Decision Tree											
1:1	0.0313	0.0004	94.984	213.10	14.70	385.30	16.90	0.927	0.963	0.950	0.928
2:1	0.0157	0.0002	95.063	213.60	14.70	385.30	16.40	0.929	0.963	0.946	0.930
10:1	0.0125	0.0004	94.603	217.00	21.00	379.00	13.00	0.943	0.948	0.945	0.937
50:1	0.0118	0.0002	92.492	219.40	36.70	363.30	10.60	0.954	0.908	0.931	0.933
100:1	0.0122	0.0000	92.492	219.40	36.70	363.30	10.60	0.954	0.908	0.931	0.933
Random Forest											
1:1	0.3043	0.0068	97.286	223.00	10.10	389.90	7.00	0.970	0.975	0.994	0.967
2:1	0.2838	0.0070	97.508	228.30	14.00	386.00	1.70	0.993	0.965	0.979	0.982
10:1	0.2848	0.0070	94.413	229.00	34.20	365.80	1.00	0.996	0.914	0.955	0.968
50:1	0.3104	0.0078	89.857	229.20	63.10	336.90	0.80	0.997	0.842	0.919	0.945
100:1	0.2917	0.0072	87.603	229.80	77.90	322.10	0.20	0.999	0.805	0.902	0.936

Column explanation:
Cost = the cost of FN:FP
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

Tweaking best algorithms

When looking at these results the IBk seems to be the winner, and the Random Forest seems to end in the second place. Therefor I will try to optimize the settings used for these algorithms. I am also curious to see how Naive Bayes will perform when the *useSupervisedDiscretization* setting is set to true, so I will also try to optimize the Naive Bayes Algorithm.

Tweaking IBk To try to get a better result from the IBk algorithm I will try to change some settings.

CrossValidate First I will run an experiment with the *crossValidate* setting set to True and KNN set to 100, this way every time a classifier is build hold-one out cross validation is used to determine the optimal number of neighbours on the training set. The experiment is run with 10-fold cross validation and repeated 10 times. In *Table 56* can be seen that for 9 out of 10 repetitions the most common optimal amount of neighbours is 2, that the mean of optimal amount of neighbours is between 3.1 and 4.4 and the median is between 2 and 4. In *Figure 6* can be seen how many times each KNN setting was determined to be optimal, from this figure it is very clear that using 2 neighbours is optimal most cases. Only in a small numer of cases (11/100) a KNN setting higher than 5 is optimal. More than 13 neighbours is never optimal in this experiment.

```
weka.experiment <- read.arff("data/weka_experiment_IBK_crossValidate.arff")

weka.experiment$Key_Run <- factor(weka.experiment$Key_Run, levels = 1:10)

KNN_summarised <- weka.experiment %>%
  dplyr::group_by(Key_Run) %>%
  dplyr::summarise(
    .groups = "keep",
    most_common_KNN = as.numeric(names(sort(table(measureKNN), decreasing = T)[1])),
    count_most_common_KNN = sort(table(measureKNN), decreasing = T)[1],
    mean_KNN = mean(measureKNN),
    median_KNN = median(measureKNN)
  )

kbl(
  KNN_summarised,
  row.names = F,
  col.names = c("Repetition", "Most common", "Number of most common amount of\nneighbours was optimal nu
  caption = "Amount of neighbours that are optimal for classifying the filtered Breast Cancer Wisconsin
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  kable_styling(latex_options = c("HOLD_position", "striped")) %>%
  column_spec(1:5, width = "2cm") %>%
  column_spec(3, width = "4cm")
```

Table 56: Amount of neighbours that are optimal for classifying the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run in Weka Experimenter using the IBk algorithm with the crossValidate setting set to True and the KNN setting set to 100. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

Repetition	Most common	Number of most common amount of neighbours was optimal number	Mean	Median
1	2	4	3.4	3.5
2	2	5	3.1	2.0
3	2	7	3.3	2.0
4	2	6	3.3	2.0
5	2	6	3.7	2.0
6	2	6	3.1	2.0
7	2	5	3.5	3.0
8	2	5	3.5	3.0
9	2	4	4.4	4.0
10	5	4	3.8	4.0

```
ggplot(weka.experiment, aes(as.factor(measureKNN))) +
  geom_bar() +
  geom_text(aes(label = ..count..), stat = "count", vjust = -0.5) +
  labs(
    title = "The number of neighbours used",
    y = "Amount of times number of neighbours was optimal",
    x = "Number of neighbours"
  )
```

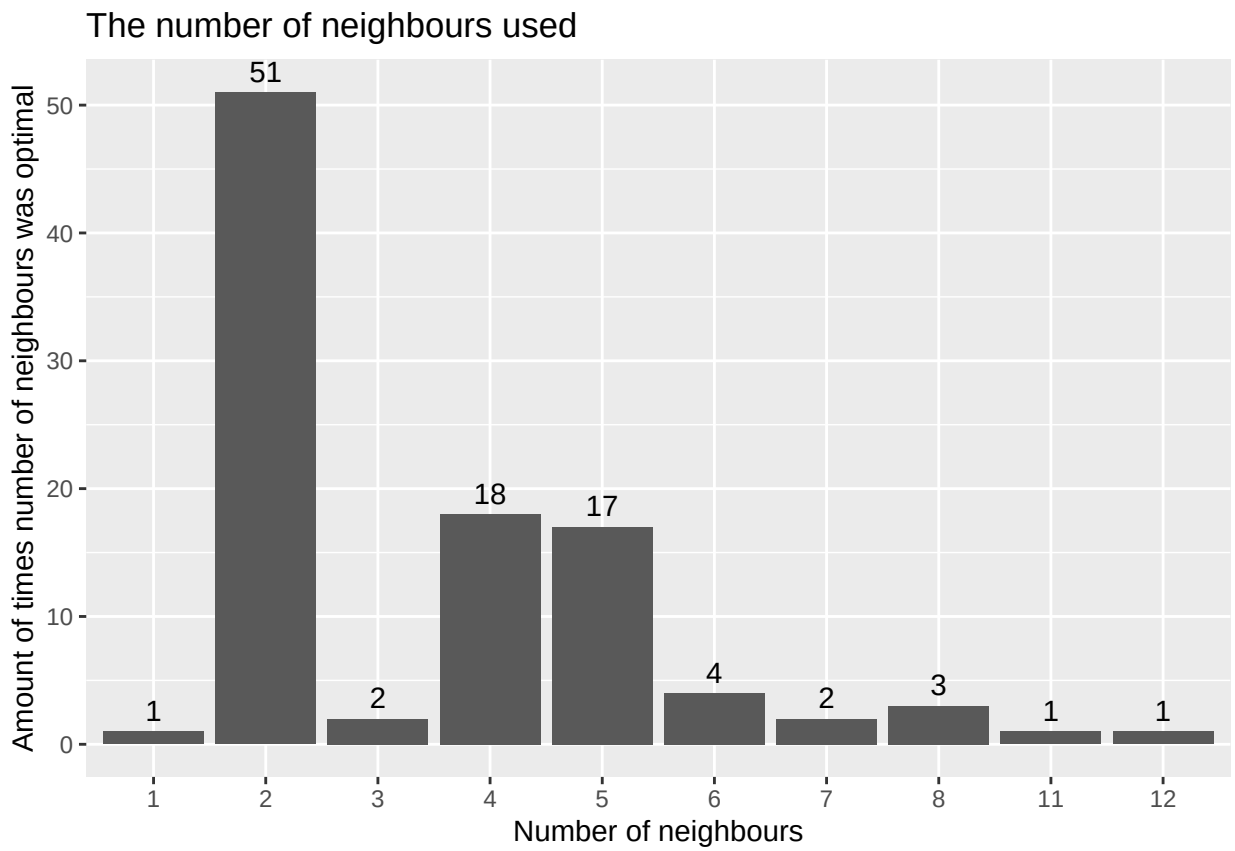

Figure 6: Number of times a specific number of neighbours was optimal for the IBk algorithm using hold-one-out cross-validation on the training set (between 1 and 100 neighbours were tested). The experiment was done uding the Weka Experimenter on the Filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and repeated 10 times. This resulted in 100 runs. In this figure isn can be observed that 2 neighbours was optimal in 51 out of 100 runs

Table 57 show the results of running 10-fold cross validation repeated 10 times with the IBk algorithm with different settings. The $crossValidation = True$ setting is used with a KNN of 100. $KNN = 2$ setting has the highest accuracy and the highest true positive rate. $KNN = 4$ has the highest AUC score and $KNN = 2$ the highest F_2 score. Below this table the result of pairwise comparison between the settings using paired t tests is printed. The tests are done on the data that is grouped per run. To correct for multiple tests the p adjustment method "Hlom" is used. The test is two sided, and we will be using $\alpha = 0.05$. The null hypothesis (H_0) is that the true difference between these group means is zero. The alternate hypothesis (H_a) is that the true difference is different from zero.

- For accuracy it can be seen that the null hypothesis is rejected in the cases:
  - $crossValidation = True$ with $KNN = 2$
  - $KNN = 2$ and $KNN = 3$.
  - $KNN = 2$ and $KNN = 4$.
- For AUC the null hypothesis is rejected in the cases:
  - $crossValidation = True$ with $KNN = 2$
  - $KNN = 2$ with $KNN = 4$
  - $KNN = 3$ with $KNN = 4$
- For F_2 score the null hypothesis is rejected in the cases:

  - $crossValidation = True$ with $KNN = 2$
  - $crossValidation = True$ with $KNN = 3$
  - $KNN = 2$ with $KNN = 3$
  - $KNN = 2$ with $KNN = 4$
  - $KNN = 3$ with $KNN = 4$
- For number of false negatives the null hypothesis is rejected in the cases:
  - $crossValidation = True$ with $KNN = 2$
  - $crossValidation = True$ with $KNN = 3$
  - $KNN = 2$ with $KNN = 3$
  - $KNN = 2$ with $KNN = 4$
  - $KNN = 3$ with $KNN = 4$

Based on these results I have decided to use the setting $KNN = 2$ for further investigation of the IBk algorithm, it has the lowest number of false negatives, the highest accuracy and F_2 score and is faster than the $crossValidation = True$ setting.

```
weka.experiment <- read.arff("data/weka_experiment_IBK_KNN.arff")

weka.experiment$Algorithm <- rep(c("KNN = 2", "KNN = 3", "KNN = 4", "crossValidation = True"), each=100

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)

by_run
```

```
## # A tibble: 500 x 15
## # Groups:   Key_Run, Algorithm, Cost [500]
##    Key_Run Algorithm Cost  Num_true_positives Num_false_positi~ Num_false_negat~
##    <fct>   <fct>     <fct>              <dbl>             <dbl>            <dbl>
##  1 1       ZeroR     1:1                    0                 0              230
##  2 1       ZeroR     2:1                  230               400                0
##  3 1       ZeroR     10:1                 230               400                0
##  4 1       ZeroR     50:1                 230               400                0
##  5 1       ZeroR     100:1                230               400                0
##  6 1       OneR      1:1                  205                32               25
##  7 1       OneR      2:1                  205                32               25
##  8 1       OneR      10:1                 205                32               25
##  9 1       OneR      50:1                 205                32               25
## 10 1       OneR      100:1                205                32               25
## # ... with 490 more rows, and 9 more variables: Num_true_negatives <dbl>,
## #   N <dbl>, Percent_correct <dbl>, True_positive_rate <dbl>,
## #   True_negative_rate <dbl>, Elapsed_Time_training <dbl>,
## #   Elapsed_Time_testing <dbl>, Area_under_ROC <dbl>, F_2_score <dbl>
```

```
by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
```

```
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  summarize(
    .groups = "keep",
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 <- format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                " ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of the IBK algorithm with the crossValidate
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(c(1:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 57: Results of experiment run in Weka Experimenter of the IBK algorithm with the crossValidate setting set to True used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
crossVal = True	5e-04	0.5204	97.206	224.00	11.60	388.40	6.00	0.974	0.971	0.991	0.969
KNN = 2	7e-04	0.0144	97.619	226.30	11.30	388.70	3.70	0.984	0.972	0.989	0.977
KNN = 3	2e-04	0.0145	97.143	221.70	9.70	390.30	8.30	0.964	0.976	0.990	0.963
KNN = 4	4e-04	0.0160	97.381	224.40	10.90	389.10	5.60	0.976	0.973	0.992	0.971

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

```
pairwise.t.test(by_run$Percent_correct, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Percent_correct and by_run$Algorithm
##
##         crossValidation = True KNN = 2 KNN = 3
## KNN = 2 0.0095                 -       -
## KNN = 3 0.5338                 0.0015  -
## KNN = 4 0.0797                 0.0268  0.0659
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Area_under_ROC, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Area_under_ROC and by_run$Algorithm
##
##         crossValidation = True KNN = 2 KNN = 3
## KNN = 2 0.02935                -       -
## KNN = 3 0.08608                0.05286 -
## KNN = 4 0.05286                0.00033 0.00029
```

```
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$F_2_score, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$F_2_score and by_run$Algorithm
##
##         crossValidation = True KNN = 2 KNN = 3
## KNN = 2 0.00312                -       -
## KNN = 3 0.01412                3.2e-05 -
## KNN = 4 0.20385                0.00055 0.00033
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Num_false_negatives, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Num_false_negatives and by_run$Algorithm
##
##         crossValidation = True KNN = 2 KNN = 3
## KNN = 2 0.00365                -       -
## KNN = 3 0.00415                1.1e-05 -
## KNN = 4 0.37320                0.00029 4.3e-05
##
## P value adjustment method: holm
```

Distance Weighting Now that I have decided to use the IBk algorithm with *KNN = 2*, I will do an experiment to see if setting a method for weighting the distance will make a difference. In *Table @ref(tab:ibk-distance-weighting-results})* the scores for different weighting settings can be seen. It can be seen that all the metrics we are interested in (accuracy, AUC and F_2 score) are the best for the algorithm without distance weighting. The algorithm without distance weighting also has the lowest number of false negatives.

From the t-tests below the table it becomes clear that the algorithm without distance weighting is significantly different from the other two algorithms on accuracy, AUC, F_2 score and number of false negatives.

Based on these results I will continue with the setting *distanceWeighting = No distance weighting.*

```
weka.experiment <- read.arff("data/weka_experiment_IBK_distanceWeight.arff")

weka.experiment$Algorithm <- rep(c("No distance weighting", "Weight by 1/distance", "Weight by 1-distanc

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)
```

```r
by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  summarize(
    .groups = "keep",
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 <- format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                " ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of the IBK algorithm with the distanceWeigh
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 58: Results of experiment run in Weka Experimenter of the IBK algorithm with the distanceWeighting setting set different values, the algorithms were used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
No distance weighting	5e-04	0.0170	97.619	226.30	11.30	388.70	3.70	0.984	0.972	0.989	0.977
Weight by 1-distance	5e-04	0.0182	96.603	217.30	8.70	391.30	12.70	0.945	0.978	0.987	0.948
Weight by 1/distance	3e-04	0.0164	96.603	217.30	8.70	391.30	12.70	0.945	0.978	0.987	0.948

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

```
pairwise.t.test(by_run$Percent_correct, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Percent_correct and by_run$Algorithm
##
##                       No distance weighting Weight by 1-distance
## Weight by 1-distance 4.6e-11               -
## Weight by 1/distance 4.6e-11               -
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Area_under_ROC, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Area_under_ROC and by_run$Algorithm
##
##                       No distance weighting Weight by 1-distance
## Weight by 1-distance 0.00045               -
## Weight by 1/distance 0.00042               0.14090
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$F_2_score, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$F_2_score and by_run$Algorithm
##
##                       No distance weighting Weight by 1-distance
## Weight by 1-distance 8e-13                 -
## Weight by 1/distance 8e-13                 -
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Num_false_negatives, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Num_false_negatives and by_run$Algorithm
##
##                       No distance weighting Weight by 1-distance
## Weight by 1-distance 9.5e-13               -
## Weight by 1/distance 9.5e-13               -
##
## P value adjustment method: holm
```

Distance Metrics Now I will test to see if the difference metric that is being can get a better classifier result. In *Table 59* the results can be seen and the accompanying t tests below the table. Using the Chebyshev Distance is significantly different from the other two distance metrics in accuracy, F_2 score and the number of false negatives. Euclidean distance and Manhattan distance significantly differ from one another in F_2 score and number of false negatives. In the table it can be seen that the Chebyshev Distance performs the worst.

The significantly different scores between Euclidean and Manhattan distance are both in favor of the Euclidean distance, this means I will keep using the Euclidean Distance when further trying to optimize the IBk algorithm.

```
weka.experiment <- read.arff("data/weka_experiment_IBK_distanceMetric.arff")

weka.experiment$Algorithm <- rep(c("Euclidean Distance", "Chebyshev Distance", "Manhattan Distance"), ea

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)


by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
```

```
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  summarize(
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 = format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                " ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of the IBK algorithm with different distanc
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 59: Results of experiment run in Weka Experimenter of the IBK algorithm with different distance metrics, the algorithms were used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
Chebyshev Distance	5e-04	0.0172	96.238	219.60	13.30	386.70	10.40	0.955	0.967	0.988	0.952
Euclidean Distance	7e-04	0.0169	97.619	226.30	11.30	388.70	3.70	0.984	0.972	0.989	0.977
Manhattan Distance	9e-04	0.0190	97.460	223.90	9.90	390.10	6.10	0.973	0.975	0.990	0.970

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

```
pairwise.t.test(by_run$Percent_correct, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Percent_correct and by_run$Algorithm
##
##                    Chebyshev Distance Euclidean Distance
## Euclidean Distance 7.9e-09            -
## Manhattan Distance 8.5e-10            0.052
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Area_under_ROC, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Area_under_ROC and by_run$Algorithm
##
##                    Chebyshev Distance Euclidean Distance
## Euclidean Distance 0.84               -
## Manhattan Distance 0.36               0.36
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$F_2_score, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$F_2_score and by_run$Algorithm
##
##                    Chebyshev Distance Euclidean Distance
## Euclidean Distance 1.6e-08            -
## Manhattan Distance 5.7e-08            0.0018
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Num_false_negatives, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Num_false_negatives and by_run$Algorithm
##
##                    Chebyshev Distance Euclidean Distance
## Euclidean Distance 2.7e-08            -
## Manhattan Distance 8.7e-07            0.00096
##
## P value adjustment method: holm
```

Attribute Selection Now I will try to optimize the algorithm by attribute selection. I have run an experiment with weka with the WrapperSubsetEval algorithm, with the same IBk algorithm used within the wrapper and the CfsSubsetEval. Both algorithms were run using BestFirst search (bi-directional) as well as with Exhaustive search. The results are displayed in *Table 60*. Something interesting that stands out to me while looking at this table is that both CfsSubsetEval runs (the run with best first search as well as the run with exhaustive search) give the exact same values for all the quality metrics. The reason behind this can be seen in *Figure 7*: the CfsSubsetEval selects all attributes and thus results in the exact same prediction as the algorithm without attribute selection. As for the Wrapper algorithm, it is a lot slower than the algorithm without attribute selection, and none of the metrics we look at (accuracy, AUC, F_2 score, number of false negative, true positive rate and true negative rate) is better than the algorithm without attribute selection. I don't even see a point in running t tests on this, if the results differ significantly it will be in favor of the algorithm without attribute selection, if not I will still choose the slgorithm without attribute selection because it is a lot faster.

```
weka.experiment <- read.arff("data/weka_experiment_IBK_attributeSelect.arff")

weka.experiment$Algorithm <- factor(
  rep(c("No attribute selection", "Wrapper Best first", "Wrapper exhaustive", "CfsSubsetEval Best First"
  levels = c("No attribute selection", "Wrapper Best first", "Wrapper exhaustive", "CfsSubsetEval Best
)

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
```

```
    as.numeric(row["Num_false_negatives"]))
  }
)


by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  dplyr::summarize(
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 = format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                "ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of the IBK algorithm using different methods
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
```

```
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 60: Results of experiment run in Weka Experimenter of the IBK algorithm using different methods for selecting attributes, the algorithms were used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
No attribute selection	0.0003	0.0167	97.619	226.30	11.30	388.70	3.70	0.984	0.972	0.989	0.977
Wrapper Best first	11.0398	0.0141	96.905	225.00	14.50	385.50	5.00	0.978	0.964	0.986	0.970
Wrapper exhaustive	52.7312	0.0138	97.159	226.20	14.10	385.90	3.80	0.983	0.965	0.987	0.975
CfsSubsetEval Best First	0.0090	0.0170	97.619	226.30	11.30	388.70	3.70	0.984	0.972	0.989	0.977
CfsSubsetEval Exhaustive	0.0097	0.0163	97.619	226.30	11.30	388.70	3.70	0.984	0.972	0.989	0.977

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

```
ggplot(subset(weka.experiment, Algorithm != "No attribute selection"), aes(measureNumAttributesSelected
  geom_bar(aes(fill = Algorithm)) +
  labs(
    title = "Number of attributes used",
    x= "Number of attributes",
    y = "Times used"
  )
```

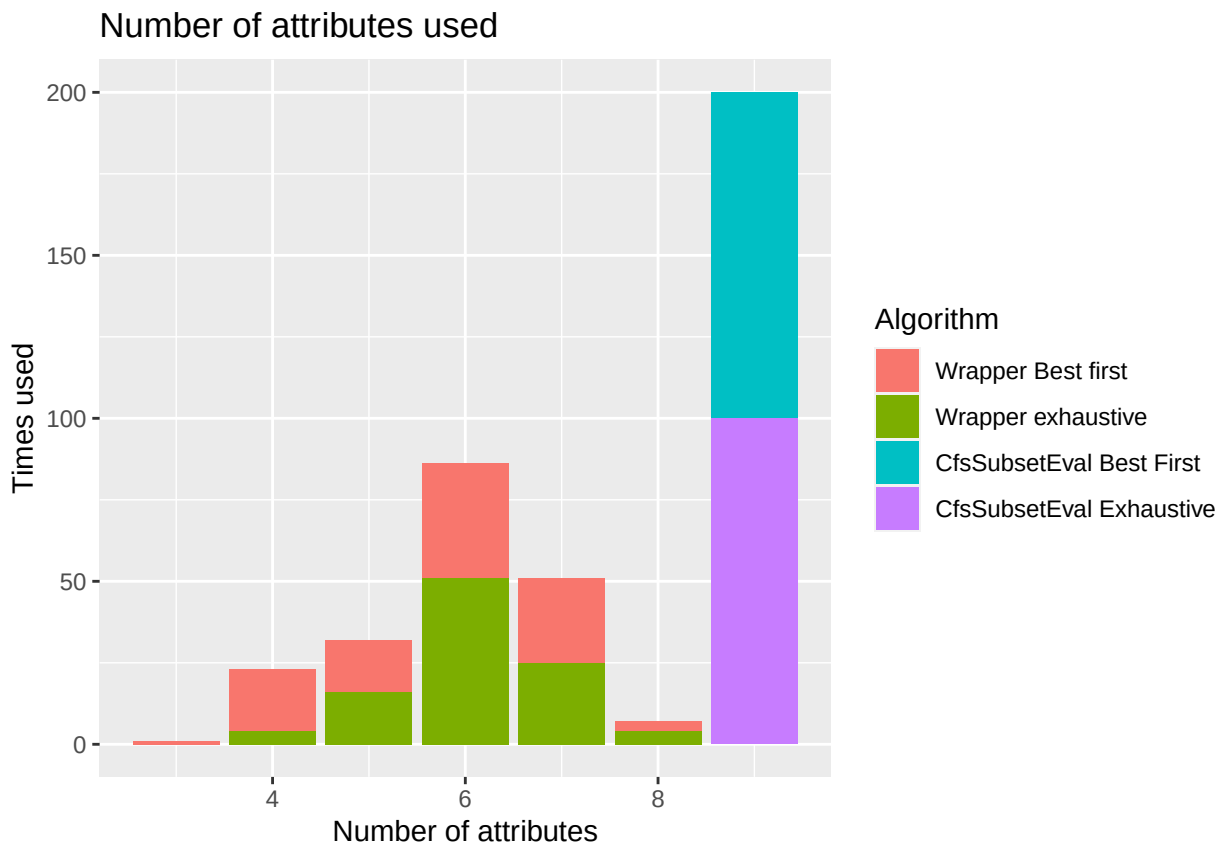

Figure 7: The results of running an experiment in Weka: using IBk (KNN=2) with different settings for attribute selection. cross-validation was set to 10, number of repititions was set to 10. Where BestFirst was used as search method the direction was set to bi-directional. Within the wrapper method the same IBk algorithm was used to determine the optimal attribute subset.

Bagging and Boosting The last try I will do to improve result of the IBk algorithm is by using bagging and boosting. I will run an experiment where I will run the normal IBk (with KNN=2), the Adaboost M1 and Bagging methods (both with IBk with KNN=2 as base algorithm). The results of this experiment are shown in *Table 61*. In the t tests below the table it can be seen that all the differences in accuracy, AUC, F_2 score and number of false negatives are significant. In the results table it can be seen that AdaBoost is worse than plain IBk all these significant values. Bagging has a lower accuracy, a higher AUC, a lower F_2 score and a higher number of false negatives compared to plain IBk. So the onlymetric that bagging is better at is AUC. It is not a lot higher compared to plain IBk, while the F_2 score is quite a bit lower. Since the plain IBk wins on most metrics, I will use the plain IBk (with KNN = 2).

```
weka.experiment <- read.arff("data/weka_experiment_IBK_boosting.arff")

weka.experiment$Algorithm <- factor(
  rep(c("No bagging/boosting", "AdaBoost M1", "Bagging"), each = 100),
  levels = c("No bagging/boosting", "AdaBoost M1", "Bagging")
)

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
```

```
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)


by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  summarize(
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 = format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                "ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of the IBK algorithm using AdaBoost and Bagg
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
```

```
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 61: Results of experiment run in Weka Experimenter of the IBK algorithm using AdaBoost and Bagging methods, the algorithms were used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
No bagging/boosting	0.0002	0.0162	97.619	226.30	11.30	388.70	3.70	0.984	0.972	0.989	0.977
AdaBoost M1	1.1849	0.0584	97.333	224.70	11.50	388.50	5.30	0.977	0.971	0.977	0.972
Bagging	0.0108	0.1714	96.825	219.10	9.10	390.90	10.90	0.953	0.977	0.992	0.954

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

```
pairwise.t.test(by_run$Percent_correct, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Percent_correct and by_run$Algorithm
##
##             No bagging/boosting AdaBoost M1
## AdaBoost M1 0.00124             -
## Bagging     0.00013             0.00069
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Area_under_ROC, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Area_under_ROC and by_run$Algorithm
##
##             No bagging/boosting AdaBoost M1
## AdaBoost M1 3.3e-09             -
## Bagging     0.00054             3.5e-08
```

```
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$F_2_score, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$F_2_score and by_run$Algorithm
##
##             No bagging/boosting AdaBoost M1
## AdaBoost M1 0.0032              -
## Bagging     6.9e-06             1.1e-05
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Num_false_negatives, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Num_false_negatives and by_run$Algorithm
##
##             No bagging/boosting AdaBoost M1
## AdaBoost M1 0.0046              -
## Bagging     4.3e-06             7.3e-06
##
## P value adjustment method: holm
```

Tweaking Naive Bayes

useSupervisedDiscretization As I mentioned in the **First results** I would like to see how Naive bayes does when the *useSupervisedDiscretization* setting is set to true. If this setting is set to true, the Naive Bayes algorithm does not depend on a normal distribution for calculating chances, which I think might improve the algorithm for the filtered Breast Cancer Wisconsin (Original) Data Set, since the distribution of the attributes does not follow a normal distribution. The results of this experiment are shown in *Table 62*. As expected the accuracy, AUC and F_2 score are better with *useSupervisedDiscretization = True*. In the t tests below the table it can be seen that these scores differ significantly, as well as the number of false negatives. So I think it is fair to say that the Naive Bayes algorithm performs better on the filtered Breast Cancer Wisconsin (Original) Data Set with the *useSupervisedDiscretization* setting set to true.

```
weka.experiment <- read.arff("data/weka_experiment_naive_bayes_useSupervisedDiscretization.arff")

weka.experiment$Algorithm <- factor(
  rep(c("Plain", "use Supervised Discretization"), each = 100),
  levels = c("Plain", "use Supervised Discretization")
)

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
```

```
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)


by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  summarize(
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 = format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                "ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of the Naive Bayes algorithm with the useSu
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
```

```
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 62: Results of experiment run in Weka Experimenter of the Naive Bayes algorithm with the useSupervisedDiscretization setting set different values, the algorithms were used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
Plain	0.0025	0.0011	96.429	224.90	17.40	382.60	5.10	0.978	0.957	0.986	0.967
use Supervised Discretization	0.0051	0.0008	97.444	225.60	11.70	388.30	4.40	0.981	0.971	0.994	0.975

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

```
pairwise.t.test(by_run$Percent_correct, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Percent_correct and by_run$Algorithm
##
##                               Plain
## use Supervised Discretization 1.1e-07
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Area_under_ROC, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Area_under_ROC and by_run$Algorithm
##
##                               Plain
## use Supervised Discretization 1.5e-08
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$F_2_score, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$F_2_score and by_run$Algorithm
##
##                               Plain
## use Supervised Discretization 4.1e-05
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Num_false_negatives, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Num_false_negatives and by_run$Algorithm
##
##                               Plain
## use Supervised Discretization 0.025
##
## P value adjustment method: holm
```

Attribute Selection Now I will see if attribute selection can improve the Naive Bayes algorithm (as opposed to the IBk algorithm). The results of the experiment that I have done is shown in *Table 63*. In this table you can see that the accuracy, the TPR, the TNR and the F_2 score are better for the plain algorithm than for the attribute selected algorithms. The AUC is the same. This time I will not test the CfsSubsetEval attribute selection method because it usually (if not always) selects all 9 attributes (can be seen in *Figure 7*). Based on these results I conclude that selecting a subset of attributes for Naive Bayes is not helpfull in classifying the filtered Breast Cancer Wisconsin (Original) Data Set.

```
weka.experiment <- read.arff("data/weka_experiment_naive_bayes_attributeSelect.arff")

weka.experiment$Algorithm <- factor(
  rep(c("No attribute selection", "Wrapper Best first", "Wrapper exhaustive"), each = 100),
  levels = c("No attribute selection", "Wrapper Best first", "Wrapper exhaustive")
)

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)


by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
```

```
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  summarize(
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 = format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                "ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of the Naive Bayes algorithm using differen
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 63: Results of experiment run in Weka Experimenter of the Naive Bayes algorithm using different methods for selecting attributes, the algorithms were used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
No attribute selection	0.0055	0.0012	97.444	225.60	11.70	388.30	4.40	0.981	0.971	0.994	0.975
Wrapper Best first	7.3821	0.0002	97.238	225.10	12.50	387.50	4.90	0.979	0.969	0.994	0.972
Wrapper exhaustive	36.1481	0.0010	97.063	224.20	12.70	387.30	5.80	0.975	0.968	0.994	0.969

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

Baggin and Boosting When using AdaBoostM1 and Bagging methods with Naive Bayes the accuracy, AUC and F_2 scores are decreased, while the training and testing time are increased. Therefore I will keep using the plain Naive Bayes Algorithm.

```
weka.experiment <- read.arff("data/weka_experiment_naive_bayes_boosting.arff")

weka.experiment$Algorithm <- factor(
  rep(c("No bagging/boosting", "AdaBoost M1", "Bagging"), each = 100),
  levels = c("No bagging/boosting", "AdaBoost M1", "Bagging")
)

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)


by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
```

```
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  summarize(
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 = format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                "ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of the Naive Bayes algorithm using plain Na
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 64: Results of experiment run in Weka Experimenter of the Naive Bayes algorithm using plain Naive Bayes as well as Naive Bayes in combination with AdaBoostM1 and a bagging method, the algorithms were used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
No bagging/boosting	0.0037	0.0004	97.444	225.60	11.70	388.30	4.40	0.981	0.971	0.994	0.975
AdaBoost M1	0.1716	0.0076	96.349	218.60	11.60	388.40	11.40	0.950	0.971	0.983	0.950
Bagging	0.0283	0.0108	96.444	225.30	17.70	382.30	4.70	0.980	0.956	0.989	0.969

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

Voting Now I will try to see if I can get a better performing algorithm by combining different algorithms. I have chosen to use the IBk (KNN = 2), Naive Bayes (useSupervisedDiscretization = True) and the Random Forest Algorithm for this. A comparison of the plain (optimized) algorithms for Naive Bayes, IBk and Random Forest and the voting combining these three algorithms is shown in *Table 65*. In the table you can see that the voting algorithm does not have the best score for accuracy, number of false negatives or F_2 score, but that the AUC score of the voting is the best among these 4 algorithms.

In the t test below the table yo can see that the accuracy of the voting algorithm is significantly different from the Naive Bayes and Random Forest algorithm, but not from the IBk algorithm. So the conclusion that I draw from the t tests in combination with the results table is that the accuracy of the voting algorithm is not significantly worse than any of the other algorithms.

Another thing that can be seen in the t tests below the table is that the AUC score differs significantly between the voting algorithm and the IBk algorithm, but not between voting and the other two algorithms. That combined with the data in the results table tells me that the Voting algorithm is at least significantly better than one of the other algorithms regarding the AUC score.

In respect to the F_2 score the voting algorithm only significantly differs from the random forest algorithm, so again I conclude that the the F_2 score of voting is not significantly worse than any of the other algorithms.

When looking at the the t tests for the number of false negatives the voting once again only significantly differs from the Random Forest, and once again in respect to the number of false negatives the voting algorithm does not perform significantly worse than any of the other three algorithms.

```
weka.experiment <- read.arff("data/weka_experiment_voting.arff")

weka.experiment$Algorithm <- factor(
  rep(c("IBk (KNN = 2)", "Naive Bayes (useSupervisedDiscretization = True)", "Random Forest", "Voting")
  levels = c("IBk (KNN = 2)", "Naive Bayes (useSupervisedDiscretization = True)", "Random Forest", "Voti
)

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
```

```
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)


by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  summarize(
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 = format(round(mean(F_2_score), digits = 3), nsmall = 3))

kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                "ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of Naive Bayes algorithm, the IBk algorith,
  booktabs = T,
  linesep = "",
  longtable = T,
```

```
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = tru
```

Table 65: Results of experiment run in Weka Experimenter of Naive Bayes algorithm, the IBk algorith, the random forest algorithm and a voting algorithm using all three algorithms. The algorithms were used on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
IBk (KNN = 2)	0.0005	0.0161	97.619	226.30	11.30	388.70	3.70	0.984	0.972	0.989	0.977
Naive Bayes (useSupervised-Discretization = True)	0.0072	0.0005	97.444	225.60	11.70	388.30	4.40	0.981	0.971	0.994	0.975
Random Forest	0.2815	0.0072	97.286	223.00	10.10	389.90	7.00	0.970	0.975	0.994	0.967
Voting	0.2778	0.0257	97.603	225.90	11.00	389.00	4.10	0.982	0.973	0.995	0.976

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

```
pairwise.t.test(by_run$Percent_correct, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Percent_correct and by_run$Algorithm
##
##                                                 IBk (KNN = 2)
## Naive Bayes (useSupervisedDiscretization = True) 0.073
## Random Forest                                   0.015
## Voting                                          0.758
##                                                 Naive Bayes (useSupervisedDiscretization = True)
## Naive Bayes (useSupervisedDiscretization = True) -
## Random Forest                                   0.191
## Voting                                          0.019
##                                                 Random Forest
## Naive Bayes (useSupervisedDiscretization = True) -
```

```
## Random Forest                                       -
## Voting                                              0.019
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Area_under_ROC, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Area_under_ROC and by_run$Algorithm
##
##                                                     IBk (KNN = 2)
## Naive Bayes (useSupervisedDiscretization = True) 9.3e-07
## Random Forest                                       4.7e-05
## Voting                                              1.1e-06
##                                                     Naive Bayes (useSupervisedDiscretization = True)
## Naive Bayes (useSupervisedDiscretization = True) -
## Random Forest                                       0.96
## Voting                                              0.96
##                                                     Random Forest
## Naive Bayes (useSupervisedDiscretization = True) -
## Random Forest                                       -
## Voting                                              0.71
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$F_2_score, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$F_2_score and by_run$Algorithm
##
##                                                     IBk (KNN = 2)
## Naive Bayes (useSupervisedDiscretization = True) 0.05082
## Random Forest                                       0.00062
## Voting                                              0.28333
##                                                     Naive Bayes (useSupervisedDiscretization = True)
## Naive Bayes (useSupervisedDiscretization = True) -
## Random Forest                                       0.00292
## Voting                                              0.11437
##                                                     Random Forest
## Naive Bayes (useSupervisedDiscretization = True) -
## Random Forest                                       -
## Voting                                              0.00039
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Num_false_negatives, by_run$Algorithm, paired = T)
```

```
##
```

```
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Num_false_negatives and by_run$Algorithm
##
##                                                 IBk (KNN = 2)
## Naive Bayes (useSupervisedDiscretization = True) 0.07454
## Random Forest                                   0.00046
## Voting                                          0.38684
##                                                 Naive Bayes (useSupervisedDiscretization = True)
## Naive Bayes (useSupervisedDiscretization = True) -
## Random Forest                                   0.00073
## Voting                                          0.38684
##                                                 Random Forest
## Naive Bayes (useSupervisedDiscretization = True) -
## Random Forest                                   -
## Voting                                          9.6e-05
##
## P value adjustment method: holm
```

Voting with cost Now I will try to lower the number of false negatives of the voting algorithm by using the cost sensitive classifier method in weka. I will use the plain voting compared to cost sensitive learning and cost sensitive classification both with a cost set to 1:5 (FP:FN). The results are shown in *Table 66*. In the table it can be seen that cost sensitive learning not only improves the number of false negatives, but does also increase the accuracy and F_2 score. In the t test below the table it can be seen that the difference in accuracy, AUC, number of false negative and F_2 score all differ significantly between the plain voting and the cost sensitive learning algorithm. Only the AUC score in favor of the plain voting algorithm.

```
weka.experiment <- read.arff("data/weka_experiment_voting_cost.arff")

weka.experiment$Algorithm <- factor(
  rep(c("Voting", "Voting, cost sensitive learning 1:5", "Voting, cost sensitive classifier 1:5"), each
  levels = c("Voting", "Voting, cost sensitive learning 1:5", "Voting, cost sensitive classifier 1:5")
)

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)


by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, Algorithm) %>%
  dplyr::summarise(
    .groups = "keep",
    Algorithm = first(Algorithm),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
```

```
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )


summarized <- by_run %>%
  dplyr::group_by(Algorithm) %>%
  dplyr::summarise(
    .groups = "keep",
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = format(round(mean(Percent_correct), digits = 3), nsmall = 3),
    TP = format(round(mean(Num_true_positives), digits = 2), nsmall = 2),
    FP = format(round(mean(Num_false_positives), digits = 2), nsmall = 2),
    TN = format(round(mean(Num_true_negatives), digits = 2), nsmall = 2),
    FN = format(round(mean(Num_false_negatives), digits = 2), nsmall = 2),
    TPR = format(round(mean(True_positive_rate), digits = 3), nsmall = 3),
    TNR = format(round(mean(True_negative_rate), digits = 3), nsmall = 3),
    AUC = format(round(mean(Area_under_ROC), digits = 3), nsmall = 3),
    F2 = format(round(mean(F_2_score), digits = 3), nsmall = 3))


kbl(
  subset(summarized),
  row.names = F,
  col.names = c("Settings", "Training", "Testing",
                "ACC", "TP", "FP", "TN", "FN",
                "TPR", "TNR", "AUC", "$F_2$"),
  caption = "Results of experiment run in Weka Experimenter of using the classifier on the filtered Brea
  booktabs = T,
  linesep = "",
  longtable = T,
  escape = FALSE
) %>%
  add_header_above(c(" " = 1, "Time" = 2, " " = 1, "Confusion matrix" = 4, " " = 4)) %>%
  kable_styling(latex_options = c("HOLD_position", "striped", "repeat_header")) %>%
  column_spec(1, width = "2.5cm") %>%
  column_spec(c(2:3, 8:11), width = "1cm") %>%
  footnote(general = c("ACC = accuracy (\\\\%)", "TP = true positive", "FP = false positive", "TN = true
```

Table 66: Results of experiment run in Weka Experimenter of using the classifier on the filtered Breast Cancer Wisconsin (Original) Data Set. The experiment was run using 10-fold cross validation and the iteration was set to 10 repetitions.

	Time			Confusion matrix							
Settings	Training	Testing	ACC	TP	FP	TN	FN	TPR	TNR	AUC	F_2
Voting	0.2850	0.0255	97.603	225.90	11.00	389.00	4.10	0.982	0.973	0.995	0.976
Voting, cost sensitive learning 1:5	0.2532	0.0243	98.048	229.00	11.30	388.70	1.00	0.996	0.972	0.994	0.987
Voting, cost sensitive classifier 1:5	0.2778	0.0257	97.508	229.00	14.70	385.30	1.00	0.996	0.963	0.979	0.984

Column explanation:
ACC = accuracy (%)
TP = true positive
FP = false positive
TN = true negative
FN = false negative
TPR = true positive rate = sensitivity = recall
TNR = true negative rate = specificity
AUC = area under the ROC curve
F_2 = the F_β score with $\beta = 2$

```
pairwise.t.test(by_run$Percent_correct, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Percent_correct and by_run$Algorithm
##
##                                       Voting
## Voting, cost sensitive learning 1:5   1.5e-05
## Voting, cost sensitive classifier 1:5 0.3
##                                       Voting, cost sensitive learning 1:5
## Voting, cost sensitive learning 1:5   -
## Voting, cost sensitive classifier 1:5 7.2e-05
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Area_under_ROC, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Area_under_ROC and by_run$Algorithm
##
##                                       Voting
## Voting, cost sensitive learning 1:5   0.0098
## Voting, cost sensitive classifier 1:5 7.3e-09
##                                       Voting, cost sensitive learning 1:5
```

```
## Voting, cost sensitive learning 1:5   -
## Voting, cost sensitive classifier 1:5 7.3e-09
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$F_2_score, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$F_2_score and by_run$Algorithm
##
##                                       Voting
## Voting, cost sensitive learning 1:5   1.5e-06
## Voting, cost sensitive classifier 1:5 4.6e-05
##                                       Voting, cost sensitive learning 1:5
## Voting, cost sensitive learning 1:5   -
## Voting, cost sensitive classifier 1:5 4.6e-05
##
## P value adjustment method: holm
```

```
pairwise.t.test(by_run$Num_false_negatives, by_run$Algorithm, paired = T)
```

```
##
##  Pairwise comparisons using paired t tests
##
## data:  by_run$Num_false_negatives and by_run$Algorithm
##
##                                       Voting
## Voting, cost sensitive learning 1:5   6.4e-07
## Voting, cost sensitive classifier 1:5 6.4e-07
##                                       Voting, cost sensitive learning 1:5
## Voting, cost sensitive learning 1:5   -
## Voting, cost sensitive classifier 1:5 -
##
## P value adjustment method: holm
```

ROC and Learning Curve Now I will make a plot showing the ROC curve of the most optimized classifier. In my case this is going to be the cost sensitive learning algorithm (cost 1:5 for FP:FN) wrapping the voting algorithm. The algorithms that are used for voting are: the IBk (KNN = 2) algorithm, the Naive Bayes (useSupervisedDiscretization = True) and th Random Forest algorithm. The ROC curve is shown in *Figure 8*. The data that this figure is based on is from running this algorithm in the Weka Explorer using 10-fold cross validation and saving the result to an arff file. I find this ROC curve quite satisfying. The curve gets pretty close to the (0, 1) coordinate which would be a perfect classifier. The place that is closest to the (0, 1) coordinate favors TPR over FPR. When looking at the figure it even looks like the TPR will be 1 when the false positive rate is around 0.2. If that is the case that would mean the algorithm could correctly classify all malignant samples, while misclassifying 20% of the benign samples.

```
ROC <- read.arff("data/ROC.arff")

colors <- colorblind_pal()(3);
```

```
ggplot(ROC, aes(x = `False Positive Rate`, y =`True Positive Rate`)) +
  geom_point(aes(color = Threshold)) +
  labs(title = "ROC curve") +
  scale_colour_gradient(
  low = colors[3],
  high = colors[2],
  space = "Lab",
  na.value = "grey50",
  guide = "colourbar",
  aesthetics = "colour"
)
```

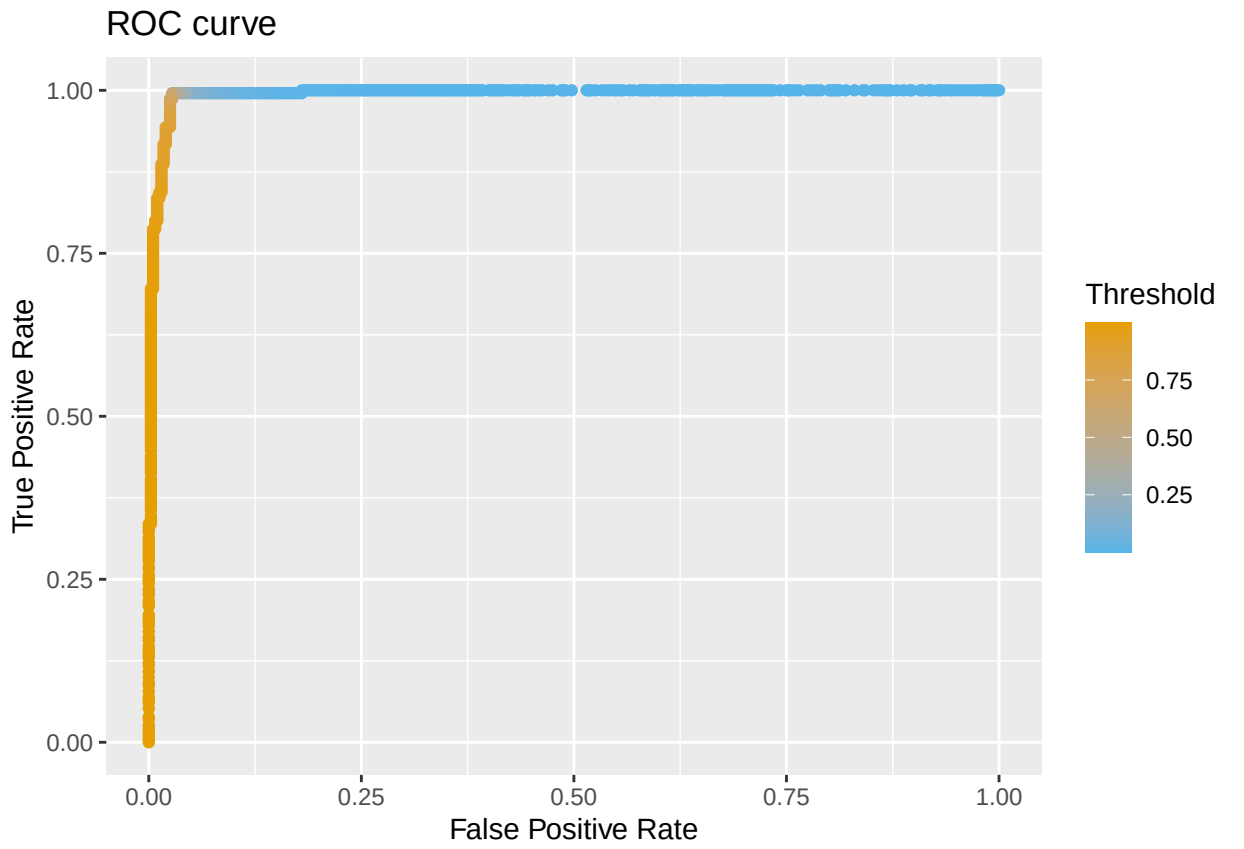

Figure 8: ROC curve of most optimized classifier algorithm use on the filtered Breast Cancer Wisconsin (Original) Data Set. The algorithm that is used is a classifier made and run in the Weka Explorer: the cost sensitive learning algorithm (cost 1:5 for FP:FN) wrapping the voting algorithm. The algorithms that are used for voting are: the IBk (KNN = 2) algorithm, the Naive Bayes (useSupervisedDiscretization = True) and th Random Forest algorithm.

Now I will run the algorithm while using different percentages of the training set. I will run this in the Weka Experimenter, wrapping the algorithm in FilteredClassifier with Resample as a filter. In Resample I will set the *noReplacement* setting to True, to disable replacing while sampling. I will set the sampleSizePercent to: 1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100. I will run these tests using 10-fold cross-validation and 10 repititions. The learning curves for accuracy, AUC, F_2 score and number of false negatives is shown in *Figure 9*. From these learning curves it becomes clear that only a couple of instances are needed for training the classifier to get a pretty high score. But then again OneR also had an accuracy

above 90%. But the number of false negatives definitely seems to go down until 80% mark. So I think it is reasonable to assume that we need at least 80% of the 630 instances that we have to get a classifier that avoids classifying malignant as benign.

```
weka.experiment <- read.arff("data/weka_experiment_learing_curve.arff")

weka.experiment$percent_used <- rep(c(seq(from = 100, to = 5, length.out = 20), 1), each = 100)

weka.experiment$F_2_score <- apply(weka.experiment, 1, function(row) {
  f.2.score(
    as.numeric(row["Num_true_positives"]),
    as.numeric(row["Num_false_positives"]),
    as.numeric(row["Num_false_negatives"]))
  }
)

by_run <- weka.experiment %>%
  dplyr::group_by(Key_Run, percent_used) %>%
  dplyr::summarise(
    .groups = "keep",
    percent_used = first(percent_used),
    Key_Run = first(Key_Run),
    Num_true_positives = sum(Num_true_positives),
    Num_false_positives = sum(Num_false_positives),
    Num_false_negatives = sum(Num_false_negatives),
    Num_true_negatives = sum(Num_true_negatives),
    N = sum(Num_true_positives, Num_false_positives, Num_false_negatives,Num_true_negatives),
    Percent_correct = ((Num_true_positives + Num_true_negatives) / N) * 100,
    True_positive_rate = (Num_true_positives / (Num_true_positives + Num_false_negatives)),
    True_negative_rate = (Num_true_negatives / (Num_true_negatives + Num_false_positives)),
    Elapsed_Time_training = sum(Elapsed_Time_training),
    Elapsed_Time_testing = sum(Elapsed_Time_testing),
    Area_under_ROC = mean(Area_under_ROC),
    F_2_score = f.2.score(Num_true_positives, Num_false_positives, Num_false_negatives)
  )

summarized <- weka.experiment %>%
  dplyr::group_by(percent_used) %>%
  dplyr::summarise(
    .groups = "keep",
    "Training time" = mean(Elapsed_Time_training),
    "Testing time" = mean(Elapsed_Time_testing),
    ACC = mean(Percent_correct),
    TP = mean(Num_true_positives),
    FP = mean(Num_false_positives),
    TN = mean(Num_true_negatives),
    FN = mean(Num_false_negatives),
    TPR = mean(True_positive_rate),
    TNR = mean(True_negative_rate),
    AUC = mean(Area_under_ROC),
    F2 = mean(F_2_score))


data <- subset(summarized, select = c(percent_used, ACC, AUC, F2, FN))
```

```r
long.data <- pivot_longer(data, 2:5, names_to = "quality_metric")

long.data$quality_metric <- factor(long.data$quality_metric)
levels(long.data$quality_metric) = c(ACC=TeX("Accuracy (%)"), AUC=TeX("AUC"), F2=TeX('$F_2$ score'), FN=

ggplot(long.data, aes(x = percent_used, y=value)) +
  geom_point() +
  geom_smooth(
    method = loess,
    se=FALSE,
     formula= y ~ x
  ) +
  expand_limits(x = 0, y = 0) +
  facet_wrap(
    quality_metric ~ .,
    ncol =2,
    scales = "free",
    labeller = label_parsed
  ) +
  labs(
    title = "Learning curves",
    x = "Percentage of training set used"
  )
```

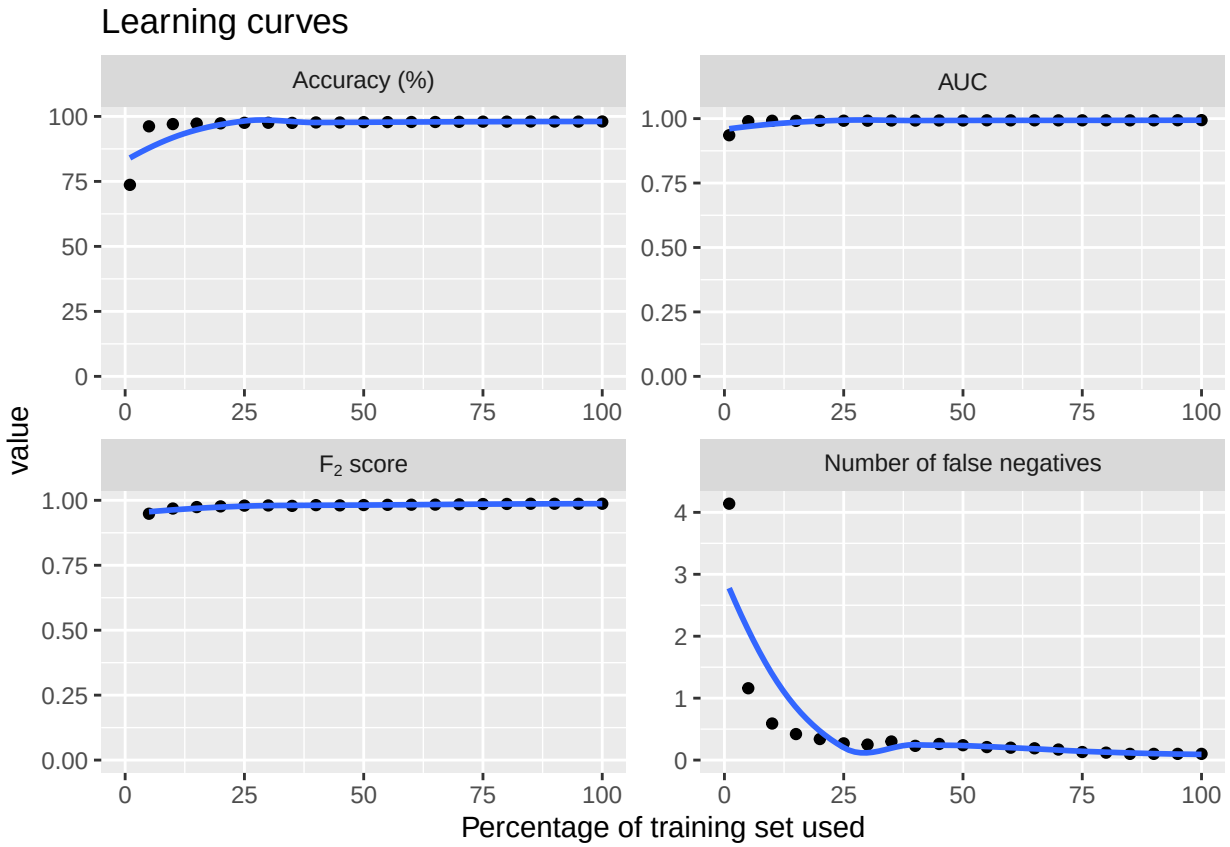

Figure 9: Different quality metrics scored for different percentages of the trainingset used. Results are from the optimal algorithm run in the Weka Experimenter using 10-fold cross validation and number of repititions set to 10