Urinary Biomarkers for Pancreatic Cancer

Log Theme09 - Introduction Machine Learning

Lisa Hu
414264
Bio-Informatics
Hanzehogeschool Groningen, ILST

Dave Langers (LADR) & Bart Barnard (BABA)

November 12, 2022

Contents

1 Data description

The data can be found on kaggle.com: Urinary biomarkers for pancreatic cancer The files are saved as `~/data/Data.csv` and `~/data/Documentation.csv` for easier access.

The following packages were used:

- dplyr
- readr
- pander
- FSA
- ggplot2
- ggpubr
- ggbiplot

2 Reading the data

We first want to create an insight of our data:

```
dataset <- read.csv("../data/Data.csv")
codebook <- read_delim("../data/codebook.txt", delim = "|")
pander(codebook[1:4], booktabs = T, caption = "Data values", split.tables = 100)
```

Table 1: Data values

Name	Full Name	Type	Unit
sample_id	Sample ID	chr	-
patient_cohort	Patient's Cohort	chr	-
sample_origin	Sample Origin	chr	-
age	Age of subject	dbl	-
sex	Sex of subject	chr	-
diagnosis	Diagnosis	dbl	-
stage	Stage	chr	-
benign_sample_diagnosis	Benign Sample's Diagnosis	chr	-
plasma_CA19_9	Blood plasma CA19-9	dbl	U/ml
creatinine	Creatinine	dbl	mg/ml
LYVE1	LYVE1	dbl	ng/ml
REG1B	REG1B	dbl	ng/ml
TFF1	TFF1	dbl	ng/ml
REG1A	REG1A	dbl	ng/ml

```
pander(codebook[c(1,5)], booktabs = T, caption = "Description",
       justify = c("right", "left"), split.tables = 100)
```

Table 2: Description

Name	Description
sample_id	Unique string identifying each subject
patient_cohort	Cohort 1 = previously used samples; Cohort 2 = newly added samples
sample_origin	BPTB: Barts Pancreas Tissue Bank, London, UK; ESP: Spanish National Cancer Research Centre, Madrid, Spain; LIV: Liverpool University, UK; UCL: University College London, UK
age	Age in years
sex	M = male; F = female
diagnosis	1 = control (no cancer); 2 = benign hepatobiliary disease; 3 = PDA (pancreatic cancer)
stage	The stage of the disease (IA, IB, IIA, IIB, III, IV)
benign_sample_diagnosis	The diagnosis for those with a benign diagnosis

Name	Description
plasma_CA19_9	Blood plasma levels of CA19-9 monoclonal antibody, usually elevated when pancreatic cancer
creatinine	Urinary biomarker of kidney function
LYVE1	Urinary levels of Lymphatic Vessel Endothelial Hyaluronan receptor 1
REG1B	Urinary levels of Regenerating Family Member 1 Beta
TFF1	Urinary levels of Trefoil Factor 1
REG1A	Urinary levels of Regenerating Family Member 1 Alpha

The information given in the codebook originates from the `~/data/Documentation.csv`. This file was given with the data file and can be found on the website.

3 Manipulate the data

A lot of the rows contain empty strings instead of NA, which has to be fixed first. Besides that, the columns `sample_id` , `patient_cohort`, `sample_origin`, and `benign_sample_diagnosis` in the dataset significant value for the analysis and are therefor dropped. A column `diagnosis_group` was added for a comparison test.

```r
# Change the empty strings to NA
dataset[dataset == ""] <- NA

# Remove unnecessary columns
drop <- c("sample_id", "patient_cohort", "sample_origin", "benign_sample_diagnosis")
dataset <- dataset[,!(names(dataset) %in% drop)]

# Group the samples
dataset <- dataset %>%
  mutate(
    ## Factor for order of age
    diagnosis_group = factor(
      dplyr::case_when(
        diagnosis == 1 ~ "Control",
        diagnosis == 2 ~ "Benign",
        stage == "I" ~ "I-IIA",
        stage == "IA" ~ "I-IIA",
        stage == "IB" ~ "I-IIA",
        stage == "II" ~ "I-II",
        stage == "IIA" ~ "I-IIA",
        stage == "IIB" ~ "I-II",
        stage == "III" ~ "III-IV",
        stage == "IV" ~ "III-IV" ),
      level = c("Control", "Benign", "I-IIA", "I-II", "III-IV")
    ),
    ## Factor if there's a blood sample or not
    blood = factor(
      dplyr::case_when(
        plasma_CA19_9 >= 0 ~ "yes",
        TRUE ~ "no"),
      level = c("yes", "no")
    )
  )
```

3.1 REG1A vs. REG1B

```r
# Perform the tests
REG1A <- dunnTest(dataset$REG1A ~ dataset$diagnosis_group)
REG1B <- dunnTest(dataset$REG1B ~ dataset$diagnosis_group)

# Create a nice format to show the correct comparisons
comparison <- t(cbind(REG1A$res[c(2:5,7,8),c(1,4)], REG1B$res[c(2:5,7,8),4]))
colnames(comparison) <- comparison[1,]
comparison <- comparison[-1,]
comparison <- apply(comparison, 2, as.numeric)
rownames(comparison) <- c("REG1A", "REG1B")
comparison[comparison > 0.05] <- "ns"
pander(comparison[,c(2,4,6)], split.tables = 100, booktabs = T,
       caption = "\\label{tab:comp}Adjusted p-values of Kruskal-Wallis test,
       Dunn's multiple comparisons; ns - not significant.
       The header shows the groups that were compared.")
```

Table 3: Adjusted p-values of Kruskal-Wallis test, Dunn's multiple comparisons; ns - not significant. The header shows the groups that were compared.

	Control - I-II	Control - I-IIA	Control - III-IV
REG1A	1.928479e-05	ns	4.837915e-07
REG1B	3.864924e-15	0.0002123534	5.789369e-17

```r
pander(comparison[,c(1,3,5)], split.tables = 100, booktabs = T)
```

	Benign - I-II	Benign - I-IIA	Benign - III-IV
REG1A	0.000768779	ns	4.494778e-05
REG1B	1.200471e-12	0.001777207	3.927231e-14

```r
dataset <- dataset[,!(names(dataset) %in% "REG1A")]
```

Although performance between the two is similar, a Kruskal-Wallis test with Dunn's multiple comparisons shows that REG1B outperforms REG1A when the control and benign samples are compared to the I-IIA PDAC samples. Therefor, REG1B was used further on in the experiments and REG1A is dropped.

3.2 Log transformation

A summary of the data shows very high maximum values, but rather low medians. A log-transformation is applied to correct this.

```r
pander(summary(dataset), split.table = 100)
```

Table 5: Table continues below

age	sex	diagnosis	stage	plasma_CA19_9	creatinine	LYVE1	REG1B	TFF1	diagnosis_group	blood
Min. :26.00	Length:590	Min. :1.000	Length:590	Min. : 0.0	Min. :0.05655	Min. : 0.000129	Min. : 0.0011	Min. : 0.005	Control:183	yes:350
1st Qu.:50.00	Class :character	1st Qu.:1.000	Class :character	1st Qu.: 8.0	1st Qu.:0.37323	1st Qu.: 0.167179	1st Qu.: 10.7572	1st Qu.: 43.961	Benign :208	no :240
Median :60.00	Mode :character	Median :2.000	Mode :character	Median : 26.5	Median :0.72384	Median : 1.649862	Median : 34.3034	Median : 259.874	I-IIA : 27	NA
Mean :59.08	NA	Mean :2.027	NA	Mean : 654.0	Mean :0.85538	Mean : 3.063530	Mean : 111.7741	Mean : 597.869	I-II : 75	NA
3rd Qu.:69.00	NA	3rd Qu.:3.000	NA	3rd Qu.: 294.0	3rd Qu.:1.13948	3rd Qu.: 5.205037	3rd Qu.: 122.7410	3rd Qu.: 742.736	III-IV : 97	NA
Max. :89.00	NA	Max. :3.000	NA	Max. :31000.0	Max. :4.11684	Max. :23.890323	Max. :1403.8976	Max. :13344.300	NA	NA
NA	NA	NA	NA	NA's :240	NA	NA	NA	NA	NA	NA

```r
log.data <- log(dataset[5:9] +1)
dataset[5:9] <- log.data
```

The samples are then grouped by diagnosis for easier access of the different samples. Table 5 shows the different amounts of samples per diagnosis and the amount of which are also blood samples.

```
# Different diagnosis and blood groups
control <- subset(dataset, diagnosis == 1)
benign <- subset(dataset, diagnosis == 2)
pdac <- subset(dataset, diagnosis == 3)

# Demographics
demograph <- dataset %>%
  group_by(sex, diagnosis, stage) %>% tally()
demograph.blood <- dataset %>%
  group_by(sex, blood) %>% tally()

pander(demograph, booktabs = T, caption = "Demographic of the samples")
```

Table 7: Demographic of the samples

sex	diagnosis	stage	n
F	1	NA	115
F	2	NA	101
F	3	I	1
F	3	IB	6
F	3	II	3
F	3	IIA	6
F	3	IIB	33
F	3	III	27
F	3	IV	7
M	1	NA	68
M	2	NA	107
M	3	IA	3
M	3	IB	6
M	3	II	4
M	3	IIA	5
M	3	IIB	35
M	3	III	49
M	3	IV	14

```
pander(demograph.blood, booktabs = T, caption = "Demographic of the blood samples")
```

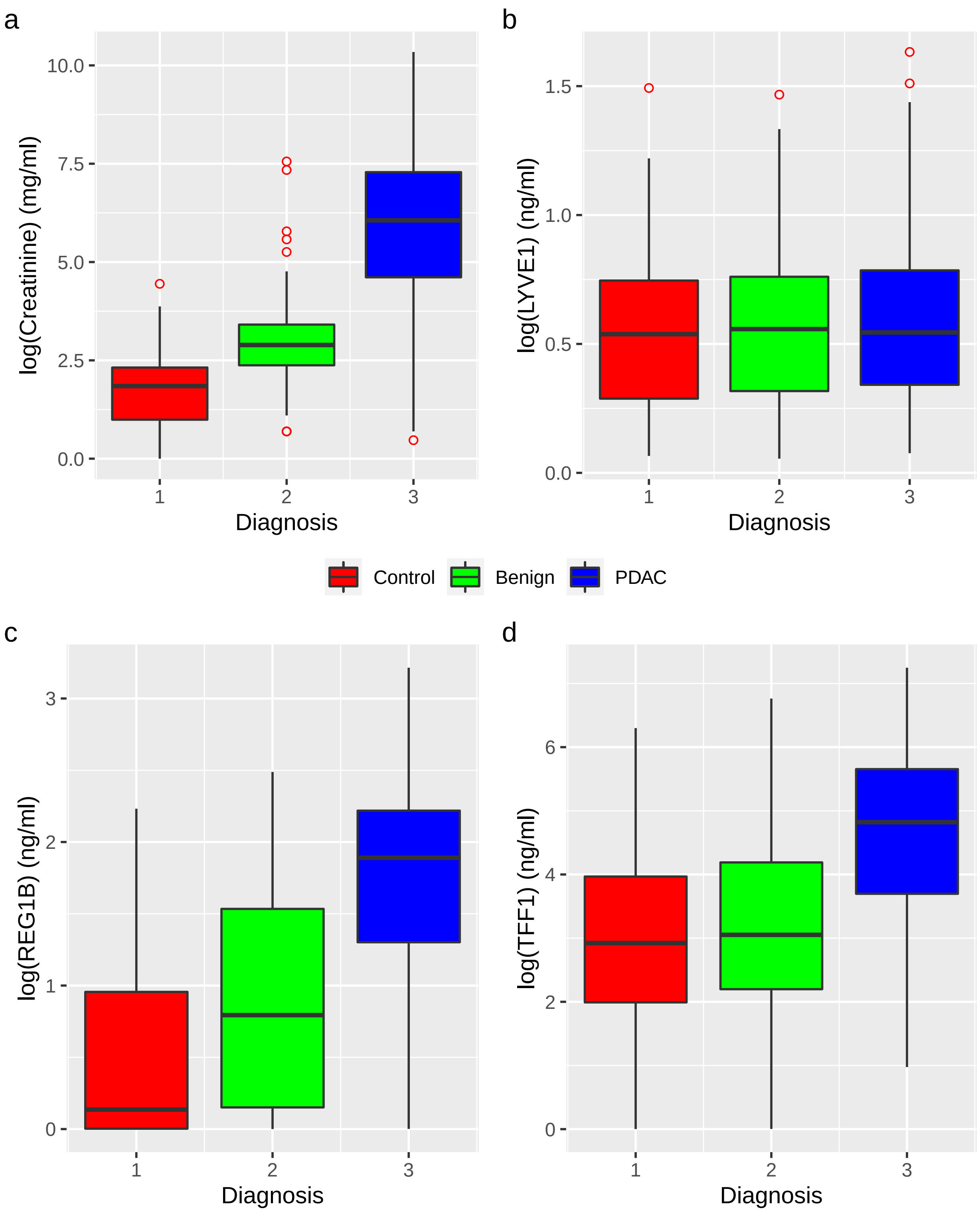
Table 8: Demographic of the blood samples

sex	blood	n
F	yes	179
F	no	120
M	yes	171
M	no	120

4 Analyse the data

4.1 Boxplots

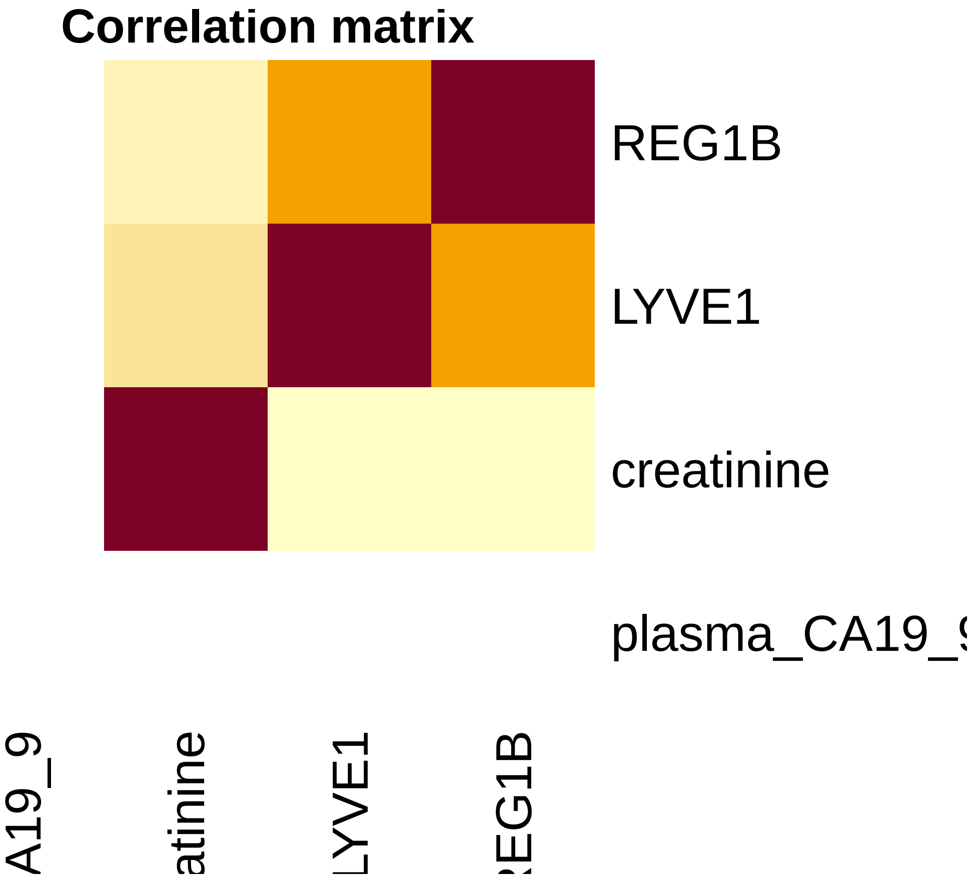
```r
# Create the boxplots for the different columns
y.values <- names(dataset[5:8])
y.labs <- c("log(Creatinine) (mg/ml)", "log(LYVE1) (ng/ml)", "log(REG1B) (ng/ml)",
            "log(TFF1) (ng/ml)")
plt.tag <- c("a", "b", "c", "d")
plts <- mapply(create.plots, y.values, y.labs, plt.tag)

# Grid and print the plots
p1 <- ggarrange(plotlist = plts[1:2], ncol = 2,
                common.legend = TRUE, legend = "bottom")
p2 <- ggarrange(plotlist = plts[3:4], ncol = 2,
                common.legend = TRUE, legend = "none")
my.grid <- ggarrange(p1, p2, nrow = 2)
print(annotate_figure(my.grid))
```



The outliers are not localized in a specific diagnosis group, but rather spread over the groups.

4.2 Correlation matrix

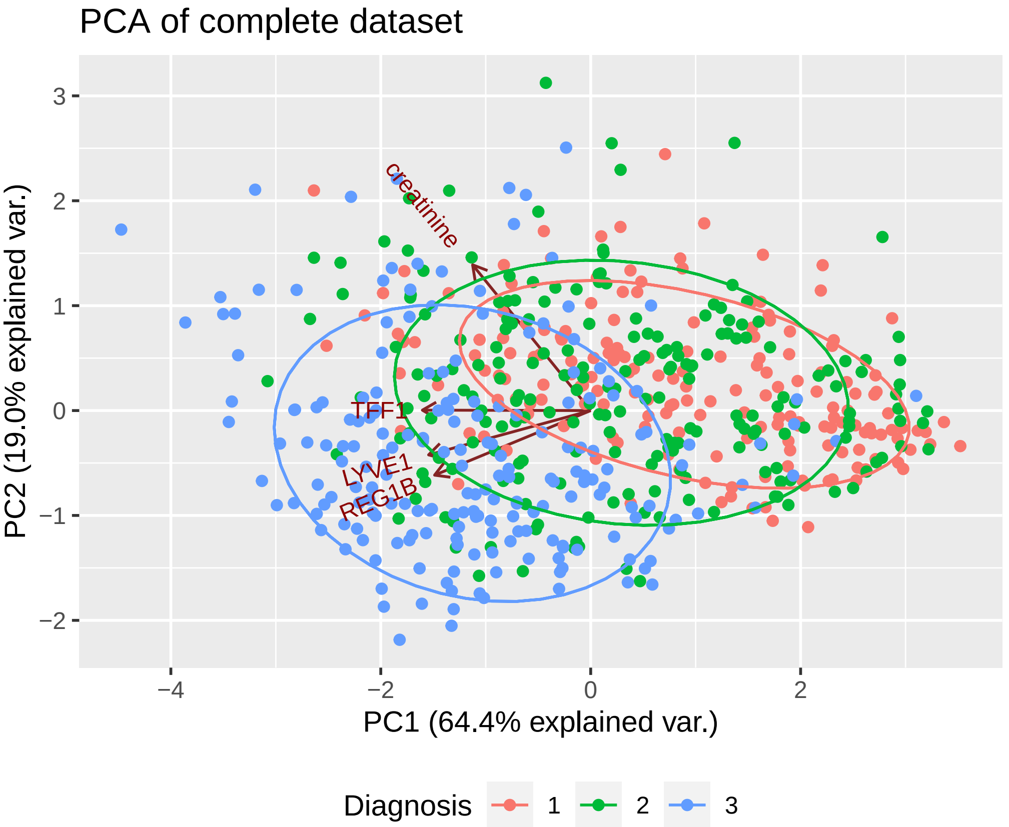
```
cor_matrix <- cor(dataset[,5:8])
heatmap(cor_matrix, scale = "column", Colv = NA, Rowv = NA, main = "Correlation matrix")
```



The heatmap shows that there is not much correlation between creatinine and the other variables. The other outstanding one has to be the TFF1 biomarker, being the most correlated variable to others.

4.3 PCA

```
pca <- prcomp(dataset[,6:9], center = TRUE, scale. = TRUE)
ggbiplot(pca, obs.scale = 1, var.scale = 1, groups = factor(dataset$diagnosis),
         ellipse = TRUE, circle = FALSE) +
  ggtitle("PCA of complete dataset") +
  guides(color = guide_legend(title = "Diagnosis")) +
  theme(legend.position = "bottom")
```



While the control and benign group show relative distance from the PDAC group, there is a still a lot of overlapping samples with the benign and PDAC groups. As earlier concluded from the heatmap, the creatinine biomarker does not show much relativeness with the other biomarkers. LYVE1 is nicely in between the TFF1 and REG1B biomarkers. Every point close tho the origin have values close to the mean for all variables.

5 Machine Learning

5.1 Quality Metrics

Accuracy is the most important quality metric to measure the performance of an algorithm. Though it is easy to choose an algorithm this way, there are always multiple algorithms one can take as final option. Hence, other quality metrics have to be taken in account to make the optimal choice.

For this project, the model is a finished product, trained and tested with the already collected data. New samples for this model are manually inserted by the acting physician, thus speed is not a relevant metric. Naturally, it is of more importance a patient with a malignant cancer should not be classified as benign, rather than a patient with a benign case being classified as malignant. These errors can be visualized in a confusion matrix, which almost all algorithms output.

5.1.1 Confusion Matrix

A standard confusion matrix is a 2x2 matrix which shows all the correct hits and rejections, and errors. In Weka, a confusion matrix looks a bit like this:

Table 9: Example of a confusion matrix

a	b	<- classified as
TP	FN	a = Control/Benign (healty)
FP	TN	b = Malignant

In this example, the correctly classified “healthy” instances are true positive (TP) and correctly classified “Malignant” instances are true negative (TN). The “healthy” instances that were classified as “Malignant” are false positive (FP) and malignant instances that were classified as benign are false negative (FN).

5.1.2 Sensitivity and False Positive Rate

Generally important for machine learning algorithms, but also for this project: Sensitivity and False Positive Rate (FPR). Also known as the true positive rate (TPR), sensitivity is calculated as $\frac{TP}{TP+FN}$. The FPR is calculated as $\frac{FP}{FP+TN}$. The FPR can also be derived from the True Negative Rate (TNR): $FPR = 1 - TNR$.

5.2 Weka: Model exploration

For the exploration of the model, the data is cleaned it contains only the biomarkers and the classification labels (Control, Benign or PDAC).

```
# Set working directory to this folder
setwd("../data")
# Read dataset
dataset <- read.csv("../data/Data.csv")
# Change the empty strings to NA
dataset[dataset == ""] <- NA

# Group the samples
dataset$sex <- factor(dataset$sex)
dataset <- dataset %>%
  mutate(
    diagnosis = factor(
      dplyr::case_when(
        diagnosis == 1 ~ "Control",
        diagnosis == 2 ~ "Benign",
        stage == "I" ~ "PDAC",
        stage == "IA" ~ "PDAC",
        stage == "IB" ~ "PDAC",
        stage == "II" ~ "PDAC",
        stage == "IIA" ~ "PDAC",
        stage == "IIB" ~ "PDAC",
        stage == "III" ~ "PDAC",
        stage == "IV" ~ "PDAC"),
      level = c("Control", "Benign", "PDAC")
    )
  )

# Drop unnecessary columns
drop <- c("sample_id", "patient_cohort", "sample_origin", "benign_sample_diagnosis",
          "REG1A", "stage")
dataset <- dataset[,!(names(dataset) %in% drop)]

# Move diagnosis and stage to last column
dataset <- dataset %>% select(-3, everything())

# Log transform and meann centering
log.data <- log(dataset[3:7] +1)
dataset[3:7] <- log.data
```

The `~/data/wekafiles/base.exp` file contains all the options used for a baseline run in Weka. The data is run through different types of algorithms with 10-fold cross validation:

```
# Read the results
result <- read_csv("../data/weka_out/base.csv")
# Make algorithm names readable
result <- algorithm.names(result)
# Results
x <- result %>%
  group_by(Key_Scheme) %>%
  summarise_at(vars(Percent_correct, True_positive_rate, False_positive_rate,
                    Area_under_ROC), list(mean = weighted.mean))
names(x) <- c("Algorithm", "Accuracy", "Sensitivity", "FPR", "AUROC")
cap <-  "\\label{tab:weka}Results of the different algorithms from Weka"
pander(x, booktabs = T, split.tables = 100, caption = cap)
```

Table 10: Results of the different algorithms from Weka

Algorithm	Accuracy	Sensitivity	FPR	AUROC
ZeroR	35.25	0	0	0.5
OneR	49.51	0.4612	0.2047	0.6283
NaiveBayes	60.07	0.5645	0.1982	0.8165
Logistic	65.2	0.528	0.1551	0.8173
SimpleLogistic	64.31	0.5296	0.1629	0.814
SMO	62.81	0.4258	0.1165	0.7721
IBk	52.37	0.3905	0.1085	0.641
J48	59.08	0.5536	0.1935	0.7674
RandomForest	65.61	0.6581	0.1601	0.8502

These results show a relative low accuracy and sensitivity. Some algorithms also have a low ROC value, putting the cutoff at 0.8: OneR, SMO, IBk and J48 will not be used. As for the remaining three: NaiveBayes has by far the lowest accuracy of them and is therefor also dropped. Leaving the options Logistic and RandomForest. Since earlier shown there is a linear correlation between the different variables, the Logistic algorithm would be more fitting for this type of data.

5.2.1 Data imbalance

To prepare the data for the optimization, the data needs to be split into groups: one file containing `Control` and `PDAC` samples, another file containing `Benign` and `PDAC` samples:

```
# Random split for training and test sets (50/50)
set.seed(391)
train.rows <- sort(sample(seq_len(nrow(dataset)), size = floor(0.7*nrow(dataset))))

training <- dataset[train.rows,]
test <- dataset[-train.rows,1:7]

control <- subset(training,
                  training$diagnosis == "Control" | training$diagnosis == "PDAC")
benign <- subset(training,
                       training$diagnosis == "Benign" | training$diagnosis == "PDAC")

# Export dataset
write.csv(dataset, "../data/wekafiles/cleaned_data.csv", row.names = F, quote = F, na="")
write.csv(control, "../data/wekafiles/control_train.csv", row.names = F, quote = F, na="")
write.csv(benign, "../data/wekafiles/benign_train.csv", row.names = F, quote = F, na="")
write.csv(test, "../data/wekafiles/test.csv", row.names = F, quote = F, na="")

# Set working directory back to log folder
setwd("../log")
```

5.2.2 Algorithm optimization

For the optimization of the algorithm, Weka's ThresholdSelector classifier will be used. This algorithm allows a threshold on the probability output of the given classifier. It is important that the FPR is as low as possible, since no patient wants to be diagnosed healthy when there is something serious swarming around. Again, these options can be imported from `~/data/wekafiles/optimization.exp`

```
# Read the optimization results
opt.res <- read_csv("../data/weka_out/optimization.csv")
# Make algorithm names readable
opt.res <- algorithm.names(opt.res)
opt.res <- opt.res %>%
  mutate(
    Options = factor(opt.res$Key_Scheme_options,
                     level = unique(opt.res$Key_Scheme_options),
                     labels = c("-", "0.550", "0.575", "0.600", "0.625", "0.65", "0.675"))
  )
opt.res$Options <- paste(opt.res$Key_Scheme, paste0("(", opt.res$Options, ")"))
# Call the wanted results
x <- opt.res %>%
  group_by(Key_Dataset, Options) %>%
  summarise_at(vars(Percent_correct, True_positive_rate, False_positive_rate),
               list(mean = weighted.mean))
names(x) <- c("Dataset", "Algorithm (threshold)", "Accuracy", "TPR", "FPR")
cap <-  "Results of the ThresholdSelector with different thresholds "
pander(x[8:14,2:5], booktabs = T, split.tables = 100,
       caption = paste0("\\label{tab:control}", cap, "(Control vs. PDAC)."))
```

Table 11: Results of the ThresholdSelector with different thresholds (Control vs. PDAC).

Algorithm (threshold)	Accuracy	TPR	FPR
Logistic (-)	85.23	0.8454	0.1412
ThresholdSelector (0.550)	86.39	0.8844	0.1558
ThresholdSelector (0.575)	87.1	0.9027	0.1594
ThresholdSelector (0.600)	87.33	0.9181	0.1695
ThresholdSelector (0.625)	87.43	0.9303	0.1792
ThresholdSelector (0.65)	87.18	0.941	0.1944
ThresholdSelector (0.675)	86.62	0.9464	0.2103

```r
pander(x[1:7,2:5], booktabs = T, split.tables = 100,
       caption = paste0("\\label{tab:benign}", cap, "(Benign vs. PDAC)."))
```

Table 12: Results of the ThresholdSelector with different thresholds (Benign vs. PDAC).

Algorithm (threshold)	Accuracy	TPR	FPR
Logistic (-)	79.23	0.794	0.2108
ThresholdSelector (0.550)	80.07	0.8264	0.2276
ThresholdSelector (0.575)	80.22	0.8443	0.2436
ThresholdSelector (0.600)	80.04	0.8581	0.2618
ThresholdSelector (0.625)	79.51	0.8636	0.2786
ThresholdSelector (0.65)	79.27	0.8775	0.2981
ThresholdSelector (0.675)	78.8	0.8893	0.3201

The results of the `Logistic` classifier shows the base algorithm without the optimization. Though the accuracy is high, the FPR is too. The FPR needs to be as low as possible without the expense of the accuracy and TPR, so the highest TPR and lowest FPR possible. Comparing with a significance of 0.001, the best option for the Control model is the one with a cutoff of 0.70, whereas the Benign model would be best with a cutoff of 0.60.