Thema09_Log

Contents

install kableExtra install knitr

Urinary biomarkers for pancreatic cancer

https://www.kaggle.com/johnjdavisiv/urinary-biomarkers-for-pancreatic- cancer

Research question: Is it possible to detect pancreatic cancer using values of the urinary biomarkers?

How can values of urinary biomarkers detect pancreatic cancer?

Introduction

Pancreatic ductal adenocarcinoma (PDAC) is one of the deadiest cancers. The chances of survival are increased when diagnosed in an early stage. However, PDAC shows symptoms when it already spread throughout the body. Most of the time, it's too late by then. There may be a way to detect PCAD in an early stage with a simple urine test, with the use of the following biomarkers: creatinine (Urinary biomarker of kidney function) LYVE1 (Urinary levels of Lymphatic vessel endothelial hyaluronan receptor 1, a protein that may play a role in tumor metastasis), REG1A and REG1B (Urinary levels of a protein that may be associated with pancreas regeneration.), and TFF1 (Urinary levels of Trefoil Factor 1, which may be related to regeneration and repair of the urinary tract)

the attributes in the data interesting for this research are the biomarker values mentioned before. There is also an attribute called diagnosis, in which the diagnosis of the sample is stated, where 1 means no PDAC, 2 means benign hepatobiliary disease (non cancerous, non harmful pancreatic condition), and 3 means that the sample has PDAC.

EDA

Codebook

```
myData <- read.csv("Data/Debernardi et al 2020 data.csv")

columns <- colnames(myData)
type <- c("character", "character", "character", "double", "character", "double", "logical", "logical",
unit <- c(NA, NA, NA, "years", "F/M", NA, NA, NA, "U/ml", "mg/ml", "ng/ml", "ng/ml", "ng/ml", "ng/ml")
descriptions = c("Unique string identifying each subject", "Cohort 1, previously used samples; Cohort 2
codebook <- data.frame(columns, type, unit, descriptions)
write.csv(codebook, "Codebook.csv", row.names = FALSE)
```

```
pander(codebook, booktabs = T, caption="The Codebook")
```

Table 1: The Codebook

columns	type	unit	descriptions
sample_id	character	NA	Unique string identifying each subject
patient_cohort	character	NA	Cohort 1, previously used samples; Cohort 2, newly added samples
sample_origin	character	NA	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	double	years	Age in years
sex	character	F/M	M = male, F = female
diagnosis	double	NA	1 = control (no pancreatic disease), 2 = benign hepatobiliary disease (119 of which are chronic pancreatitis); 3 = Pancreatic ductal adenocarcinoma, i.e. pancreatic cancer
stage	logical	NA	For those with pancratic cancer, what stage was it? One of IA, IB, IIA, IIIB, III, IV
benign_sample_diagnosis	logical	NA	For those with a benign, non-cancerous diagnosis, what was the diagnosis?
plasma_CA19_9	double	U/ml	Blood plasma levels of CA 19–9 monoclonal antibody that is often elevated in patients with pancreatic cancer. Only assessed in 350 patients (one goal of the study was to compare various CA 19-9 cutpoints from a blood sample to the model developed using urinary samples).
creatinine	double	mg/ml	Urinary biomarker of kidney function
LYVE1	double	ng/ml	Urinary levels of Lymphatic vessel endothelial hyaluronan receptor 1, a protein that may play a role in tumor metastasis
REG1B	double	ng/ml	Urinary levels of a protein that may be associated with pancreas regeneration.
TFF1	double	ng/ml	Urinary levels of Trefoil Factor 1, which may be related to regeneration and repair of the urinary tract
REG1A	double	ng/ml	Urinary levels of a protein that may be associated with pancreas regeneration. Only assessed in 306 patients (one goal of the study was to assess REG1B vs REG1A)

Data exploration

Visualization

To look at the data and any possible missing data, here's a table containing only the relevant columns for this research.

```
relevant <- myData[c(4, 6, 9:14)]
pander(head(relevant, n=15), booktabs = T, caption = "Values of biomarkers and the diagnosis")
```

Table 2: Values of biomarkers and the diagnosis

age	diagnosis	plasma_CA19_9	creatinine	LYVE1	REG1B	TFF1	REG1A
33	1	11.7	1.832	0.8932	52.95	654.3	1262
81	1	NA	0.9727	2.038	94.47	209.5	228.4
51	1	7	0.7804	0.1456	102.4	461.1	NA
61	1	8	0.7012	0.002805	60.58	142.9	NA
62	1	9	0.2149	0.0008596	65.54	41.09	NA
53	1	NA	0.8482	0.003393	62.13	59.79	NA
70	1	NA	0.622	0.1744	152.3	117.5	NA
58	1	11	0.8935	0.003574	3.73	40.29	NA
59	1	NA	0.4863	0.001945	7.021	26.78	NA
56	1	24	0.6107	0.2788	83.93	19.18	NA
77	1	NA	0.2941	0.001176	6.218	28.3	NA
71	1	23	1.052	0.8603	243.1	608.3	NA
49	1	NA	0.8596	1.416	151.8	74.19	505.6
53	1	7	1.911	1.517	150.9	590.7	NA
56	1	12	0.9161	0.5996	93.81	93.58	NA

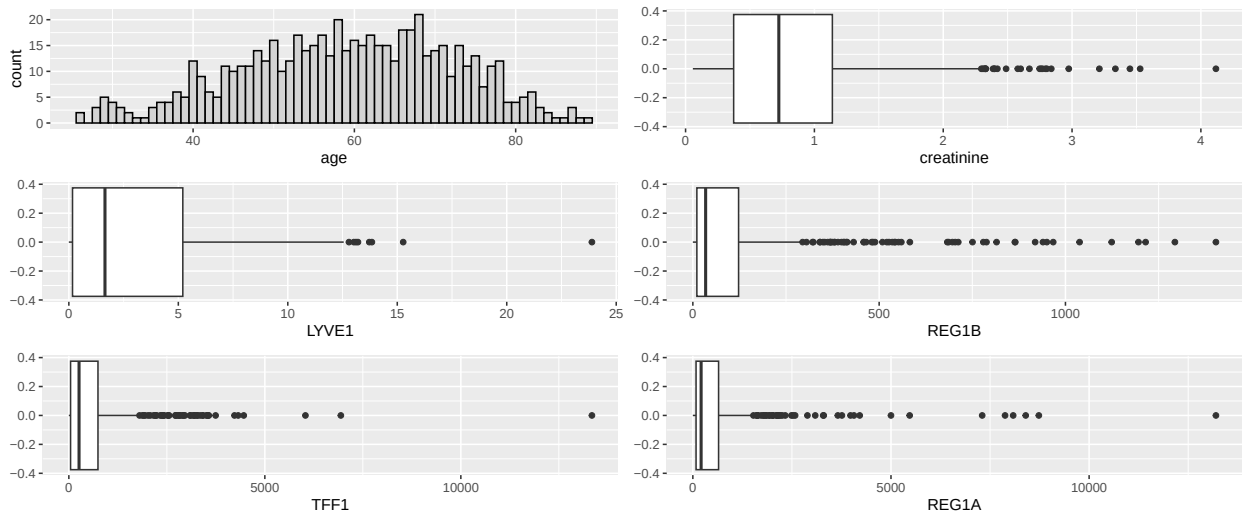
As is it obvious to see, the REG1A column contains a lot of missing values. The question is exactly how is this possible and what does it mean for the rest of the data? The first and most simple solution is that we don't need this column at all, since the REG1B is similar in function and containing all of the data. Another solution that's not very logical, is to use only the rows where there is a value in the REG1A column. The most logical solution in my opinion is to look at every column separately, and then remove the missing data. When the cleaned data then is plotted, it's possible to look at all the columns together and see if there is any correlation or trend to figure out.

knowing this, let's have a look at the important raw data with the use of boxplots.

```
p1 <- ggplot(myData, aes(x=age)) +
  geom_histogram(fill = "lightgrey", col = "black", binwidth = 1)

p2 <- ggplot(myData, aes(x=creatinine)) +
  geom_boxplot()
p3 <- ggplot(myData, aes(x=LYVE1)) +
  geom_boxplot()
p4 <- ggplot(myData, aes(x=REG1B)) +
  geom_boxplot()
p5 <- ggplot(myData, aes(x=TFF1)) +
  geom_boxplot()
p6 <- ggplot(myData, aes(x=REG1A)) +
  geom_boxplot()
```

```
grid.arrange(p1, p2, p3, p4, p5, p6, nrow = 3)
```

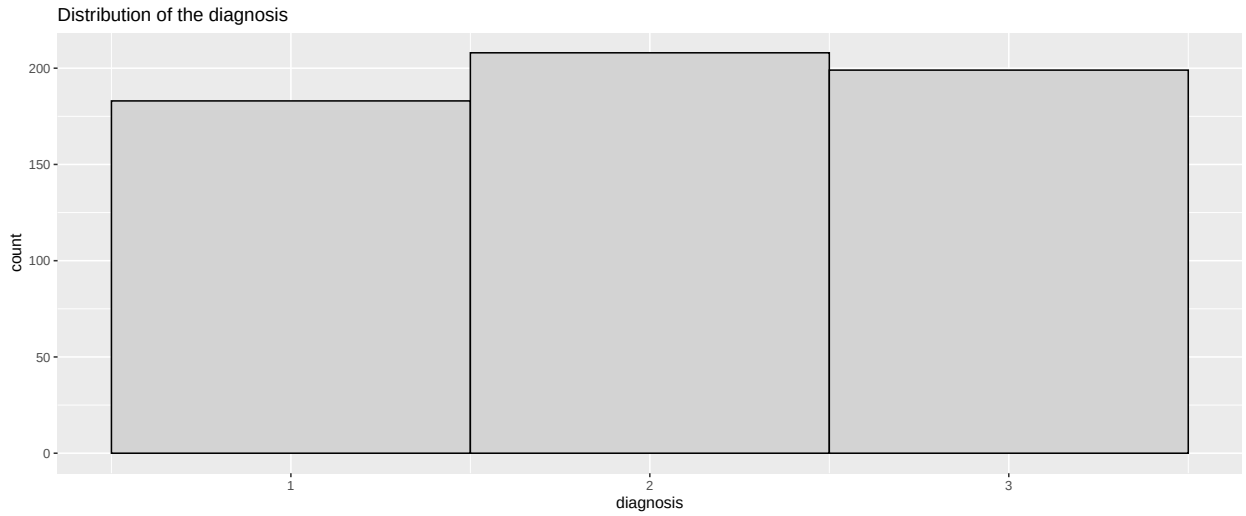


as you can see, there are also a lot of outliers which we need to check. We can normalize the data using a log transformation.

```
log <- log(myData[9:14] +1)
myData[9:14] <- log
```

Before proceeding, it is quite helpful to look at the distribution of the diagnosis column, to see if they're evenly distributed or not.

```
ggplot(myData, aes(x=diagnosis)) +
  geom_histogram(fill = "lightgrey", col = "black", binwidth = 1) +
  ggtitle("Distribution of the diagnosis")
```



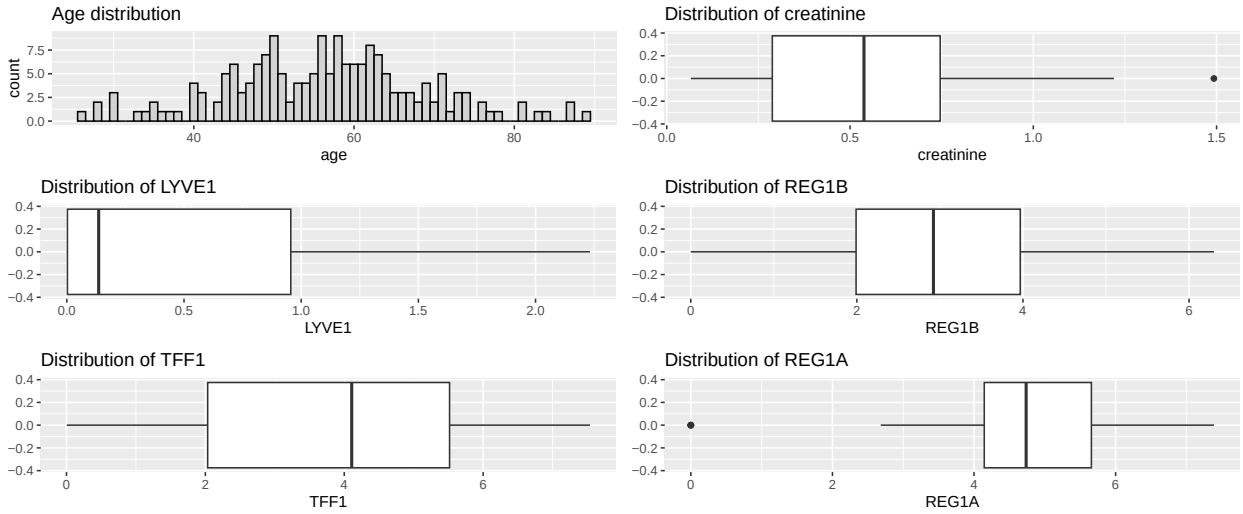
What we should do now, is look at the values of those urinary levels with each diagnosis and see if there are any obvious differences.
Let's start with the samples that do not have PDAC

```
myData1 <- subset(myData, myData$diagnosis == 1)

p1 <- ggplot(myData1, aes(x=age)) +
  geom_histogram(fill = "lightgrey", col = "black", binwidth = 1) +
  ggtitle("Age distribution")

p2 <- ggplot(myData1, aes(x=creatinine)) +
  geom_boxplot() +
  ggtitle("Distribution of creatinine")
p3 <- ggplot(myData1, aes(x=LYVE1)) +
  geom_boxplot() +
  ggtitle("Distribution of LYVE1")
p4 <- ggplot(myData1, aes(x=REG1B)) +
  geom_boxplot() +
  ggtitle("Distribution of REG1B")
p5 <- ggplot(myData1, aes(x=TFF1)) +
  geom_boxplot() +
  ggtitle("Distribution of TFF1")
p6 <- ggplot(myData1, aes(x=REG1A)) +
  geom_boxplot() +
  ggtitle("Distribution of REG1A")

grid.arrange(p1, p2, p3, p4, p5, p6, nrow = 3)
```



Here are the values of the samples with non-cancerous pancreatic conditions.

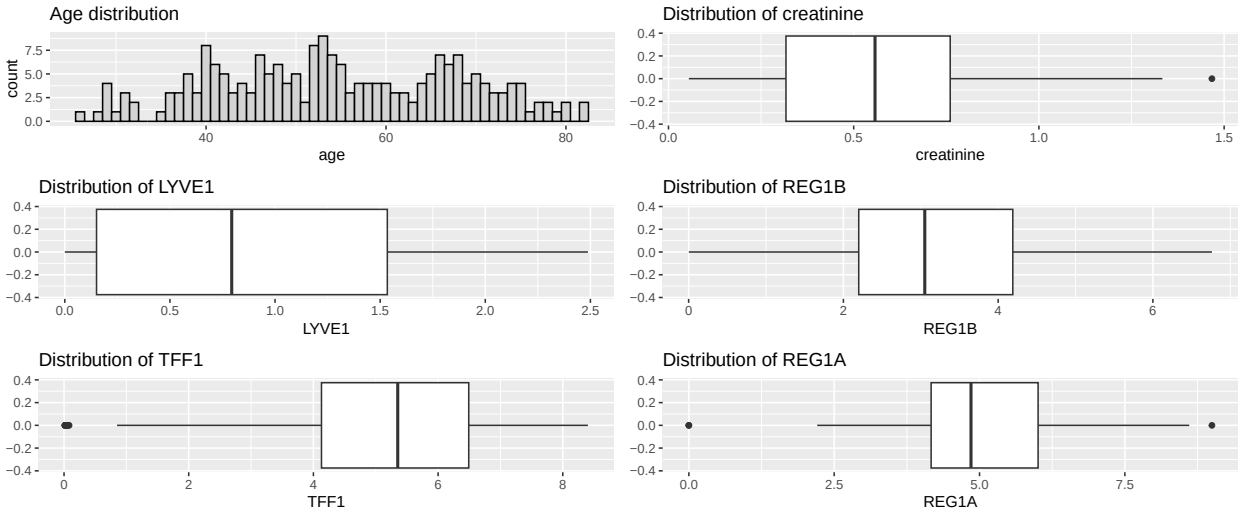
```
myData2 <- subset(myData, myData$diagnosis == 2)

p1 <- ggplot(myData2, aes(x=age)) +
  geom_histogram(fill = "lightgrey", col = "black", binwidth = 1) +
  ggtitle("Age distribution")

p2 <- ggplot(myData2, aes(x=creatinine)) +
  geom_boxplot() +
  ggtitle("Distribution of creatinine")
p3 <- ggplot(myData2, aes(x=LYVE1)) +
```

```
  geom_boxplot() +
  ggtitle("Distribution of LYVE1")
p4 <- ggplot(myData2, aes(x=REG1B)) +
  geom_boxplot() +
  ggtitle("Distribution of REG1B")
p5 <- ggplot(myData2, aes(x=TFF1)) +
  geom_boxplot() +
  ggtitle("Distribution of TFF1")
p6 <- ggplot(myData2, aes(x=REG1A)) +
  geom_boxplot() +
  ggtitle("Distribution of REG1A")

grid.arrange(p1, p2, p3, p4, p5, p6, nrow = 3)
```



And finally the samples that do have PDAC.

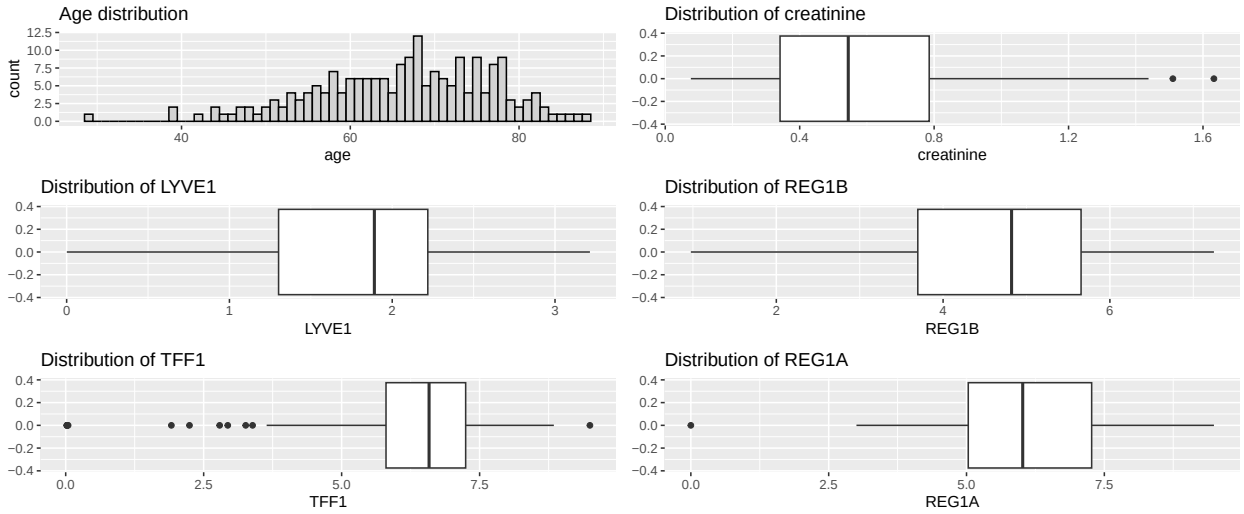
```
myData3 <- subset(myData, myData$diagnosis == 3)

p1 <- ggplot(myData3, aes(x=age)) +
  geom_histogram(fill = "lightgrey", col = "black", binwidth = 1) +
  ggtitle("Age distribution")

p2 <- ggplot(myData3, aes(x=creatinine)) +
  geom_boxplot() +
  ggtitle("Distribution of creatinine")
p3 <- ggplot(myData3, aes(x=LYVE1)) +
  geom_boxplot() +
  ggtitle("Distribution of LYVE1")
p4 <- ggplot(myData3, aes(x=REG1B)) +
  geom_boxplot() +
  ggtitle("Distribution of REG1B")
p5 <- ggplot(myData3, aes(x=TFF1)) +
  geom_boxplot() +
  ggtitle("Distribution of TFF1")
p6 <- ggplot(myData3, aes(x=REG1A)) +
  geom_boxplot() +
```

```
  ggtitle("Distribution of REG1A")

grid.arrange(p1, p2, p3, p4, p5, p6, nrow = 3)
```



At first glance, there are notable differences but we can further investigate this in future steps of this research.

```
write.csv(myData, "Data/DataCleaned.csv")
```

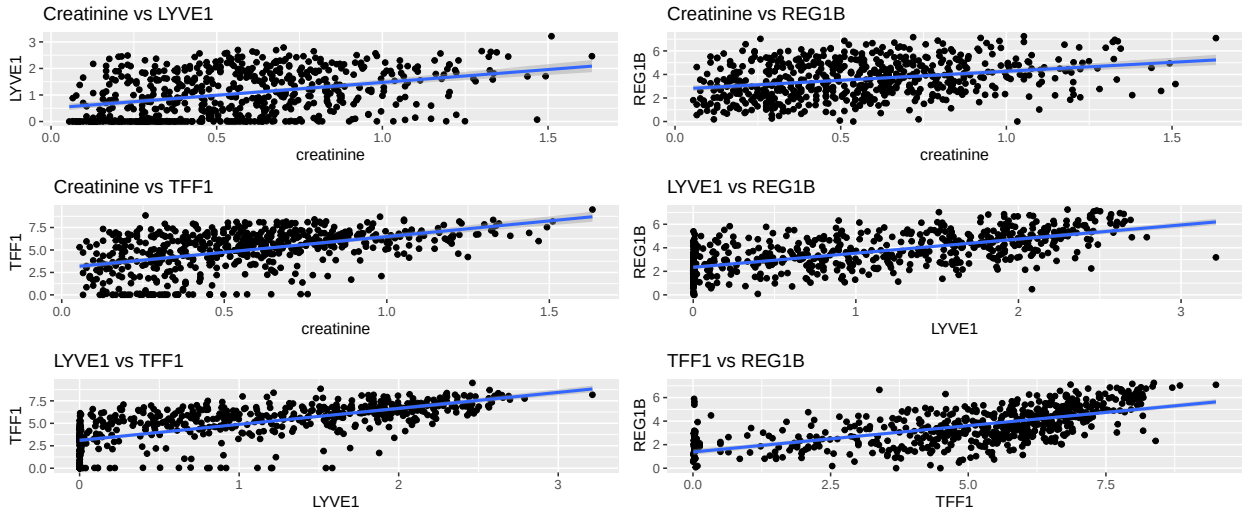
Correlation

Now we can check if the different biomarkers correlate with each other in any way.

```
p1 <- ggplot(myData, aes(x=creatinine, y=LYVE1)) +
  geom_point() +
  geom_smooth(method='lm', formula = y~x) +
  ggtitle("Creatinine vs LYVE1")
p2 <- ggplot(myData, aes(x=creatinine, y=REG1B)) +
  geom_point() +
  geom_smooth(method='lm', formula = y~x) +
  ggtitle("Creatinine vs REG1B")
p3 <- ggplot(myData, aes(x=creatinine, y=TFF1)) +
  geom_point() +
  geom_smooth(method='lm', formula = y~x) +
  ggtitle("Creatinine vs TFF1")
p4 <- ggplot(myData, aes(x=LYVE1, y=REG1B)) +
  geom_point() +
  geom_smooth(method='lm', formula = y~x) +
  ggtitle("LYVE1 vs REG1B")
p5 <- ggplot(myData, aes(x=LYVE1, y=TFF1)) +
  geom_point() +
  geom_smooth(method='lm', formula = y~x) +
  ggtitle("LYVE1 vs TFF1")
p6 <- ggplot(myData, aes(x=TFF1, y=REG1B)) +
  geom_point() +
  geom_smooth(method='lm', formula = y~x) +
```

```
  ggtitle("TFF1 vs REG1B")

grid.arrange(p1, p2, p3, p4, p5, p6, nrow=3)
```



In this visualization there is a bit of correlation visible

Now, let's prepare the data for machine learning. The first step will be to remove the columns that aren't really necessary and relevant to the research to make it more clear.

```
myData <- subset(myData, select = -c(patient_cohort, sample_origin, benign_sample_diagnosis))
pander(summary(myData), booktabs = T, caption = "Relevant data")
```

Table 3: Relevant data (continued below)

sample_id	age	sex	diagnosis
Length:590	Min. :26.00	Length:590	Min. :1.000
Class :character	1st Qu.:50.00	Class :character	1st Qu.:1.000
Mode :character	Median :60.00	Mode :character	Median :2.000
NA	Mean :59.08	NA	Mean :2.027
NA	3rd Qu.:69.00	NA	3rd Qu.:3.000
NA	Max. :89.00	NA	Max. :3.000
NA	NA	NA	NA

Table 4: Table continues below

stage	plasma_CA19_9	creatinine	LYVE1
Length:590	Min. : 0.000	Min. :0.05501	Min. :0.000129
Class :character	1st Qu.: 2.197	1st Qu.:0.31717	1st Qu.:0.154584
Mode :character	Median : 3.314	Median :0.54455	Median :0.974503
NA	Mean : 3.882	Mean :0.56738	Mean :1.054778
NA	3rd Qu.: 5.687	3rd Qu.:0.76056	3rd Qu.:1.825361
NA	Max. :10.342	Max. :1.63254	Max. :3.214479
NA	NA's :240	NA	NA

REG1B	TFF1	REG1A
Min. :0.001104	Min. :0.005279	Min. :0.000
1st Qu.:2.464467	1st Qu.:3.805664	1st Qu.:4.403
Median :3.563973	Median :5.563994	Median :5.345
Mean :3.607996	Mean :4.979594	Mean :5.411
3rd Qu.:4.818184	3rd Qu.:6.611675	3rd Qu.:6.477
Max. :7.247720	Max. :9.498920	Max. :9.488
NA	NA	NA's :284

Quality Metrics

There are 3 possible diagnoses in this data, those are 1: The patient has PDAC; 2: The patient has pancreatic condition, but not cancer; and 3, The patient doesn't have any form of pancreatic conditions or cancer.

It is important to determine the positive and negative diagnoses here, to find the true positives and true negatives. This research is about whether the sample has pancreatic cancer or not. So positive in this case will be diagnosis 1, and negative both 2 and 3, since they're both non-cancerous.

False negatives is the least wanted with research like this, because in that case the patient has PDAC, but the test says the patient doesn't, so there won't be any further treatment, while the patient needs it. it's better to have someone classified as PDAC, while the sample doesn't have it, because in further research to really test the person on PDAC, the test will come out negative.

In the machine learning process it's important to have the lowest possible amount of false negatives, so the false negative rate (FNR) should be as low as possible. The FNR tells of all the truly positives, what rate is

wrongly classified. In the Java wrapper later on, it is important to calculate the FNR to see if it is reasonably low. On the other hand it is important to have a high amount of true positives, so those patients get the right treatment in time. So the FNR must be as low as possible while the TPR must be as high as possible.

Machine learning algorithms

The next step is to determine which ML algorithm gives the best results. First let's look at the baseline accuracy with ZeroR. Weka correctly classifies 35.25% of the instances with 10-fold cross-validation. Using the experimenter in weka, it is possible to see the performance of every algorithm really fast. The experimenter uses 10-fold cross-validation and repeats it 10 times. So in the data there are 100 rows per ML algorithm. The code below reads per algorithm and averages the 100 rows to get a highly accurate representation of the results. In the table below there is a summary of every used cost-sensitive classifier and the resulting quality metrics. Remember it is important that the TPR is as high as possible, while the FNR is as low as possible.

```
ML_dataLoc <- "Data/ExperimentClean.csv"
ML_data <- read.csv(ML_dataLoc)
ML_ZeroR <- ML_data[c(1:100),]
ML_OneR <- ML_data[c(101:200),]
ML_J48 <- ML_data[c(201:300),]
ML_IBK <- ML_data[c(301:400),]
ML_NaiveBayes <- ML_data[c(401:500),]
ML_SimpleLog <- ML_data[c(501:600),]
ML_SMO <- ML_data[c(601:700),]
ML_RandomForest <- ML_data[c(701:800),]
algos <- list(ML_ZeroR, ML_OneR, ML_J48, ML_IBK, ML_NaiveBayes, ML_SimpleLog, ML_SMO, ML_RandomForest)
avgs_pc <- list()
avgs_pi <- list()
avgs_tp <- list()
avgs_fp <- list()
avgs_tn <- list()
avgs_fn <- list()
avgs_pr <- list()
avgs_rc <- list()
avgs_roc <- list()

for (algo in algos) {
  percent_correct <- algo[2]
  avg <- lapply(percent_correct, mean)
  avgs_pc[[length(avgs_pc) + 1]] <- avg
}
for (algo in algos) {
  percent_incorrect <- algo[3]
  avg <- lapply(percent_incorrect, mean)
  avgs_pi[[length(avgs_pi) + 1]] <- avg
}
for (algo in algos) {
  TP <- algo[4]
  avg <- lapply(TP, mean)
  avgs_tp[[length(avgs_tp) + 1]] <- avg
}
for (algo in algos) {
  FP <- algo[5]
```

```
  avg <- lapply(FP, mean)
  avgs_fp[[length(avgs_fp) + 1]] <- avg
}
for (algo in algos) {
  TN <- algo[6]
  avg <- lapply(TN, mean)
  avgs_tn[[length(avgs_tn) + 1]] <- avg
}
for (algo in algos) {
  FN <- algo[7]
  avg <- lapply(FN, mean)
  avgs_fn[[length(avgs_fn) + 1]] <- avg
}
for (algo in algos) {
  precision <- algo[8]
  avg <- lapply(precision, mean)
  avgs_pr[[length(avgs_pr) + 1]] <- avg
}
for (algo in algos) {
  recall <- algo[9]
  avg <- lapply(recall, mean)
  avgs_rc[[length(avgs_rc) + 1]] <- avg
}
for (algo in algos) {
  roc_area <- algo[10]
  avg <- lapply(roc_area, mean)
  avgs_roc[[length(avgs_roc) + 1]] <- avg
}
avgs_df <- data.frame(row.names = c("ZeroR", "OneR", "J48", "IBK", "NaiveBayes", "SimpleLogistics", "SM

vec_pc <- unlist(avgs_pc)
vec_pi <- unlist(avgs_pi)
vec_tp <- unlist(avgs_tp)
vec_fp <- unlist(avgs_fp)
vec_tn <- unlist(avgs_tn)
vec_fn <- unlist(avgs_fn)
vec_pr <- unlist(avgs_pr)
vec_rc <- unlist(avgs_rc)
vec_roc <- unlist(avgs_roc)

avgs_df$percent_correct_avgs <- vec_pc
avgs_df$percent_incorrect_avgs <- vec_pi
avgs_df$TP_avgs <- vec_tp
avgs_df$FP_avgs <- vec_fp
avgs_df$TN_avgs <- vec_tn
avgs_df$FN_avgs <- vec_fn
avgs_df$precision_avgs <- vec_pr
avgs_df$recall_avgs <- vec_rc
avgs_df$ROC_Area_avgs <- vec_roc

pander(avgs_df, booktabs = T, caption = "Machine Learning Summary")
```

Table 6: Machine Learning Summary (continued below)

	percent_correct_avgs	percent_incorrect_avgs	TP_avgs
ZeroR	31.02	68.98	1
OneR	48.93	51.07	0.7135
J48	57.15	42.85	0.7579
IBK	55.37	44.63	0.7975
NaiveBayes	52.36	47.64	0.7744
SimpleLogistics	54.1	45.9	0.8378
SMO	51.86	48.14	0.9143
RandomForest	59.86	40.14	0.7318

Table 7: Table continues below

	FP_avgs	TN_avgs	FN_avgs	precision_avgs
ZeroR	1	0	0	0.3102
OneR	0.4051	0.5949	0.2865	0.4455
J48	0.2842	0.7158	0.2421	0.5493
IBK	0.3759	0.6241	0.2025	0.492
NaiveBayes	0.394	0.606	0.2256	0.4698
SimpleLogistics	0.4029	0.5971	0.1622	0.4846
SMO	0.5269	0.4731	0.0857	0.4398
RandomForest	0.2395	0.7605	0.2682	0.5841

	recall_avgs	ROC_Area_avgs
ZeroR	1	0.5
OneR	0.7135	0.6542
J48	0.7579	0.7726
IBK	0.7975	0.7827
NaiveBayes	0.7744	0.763
SimpleLogistics	0.8378	0.801
SMO	0.9143	0.711
RandomForest	0.7318	0.8276

```
write.csv(avgs_df, file = "Data/derbernardi_summary.csv")
```