EDA/machine learning breast cancer proteomes

Pascal Visser

14-9-2021

Contents

```
# Used libraries
library(ggplot2)
library(reshape2)
library(dplyr)
library(tidyr)
library(pander)
library(ggrepel)
library(forcats)
library(scales)
library(knitr)
library(gridExtra)
library(grid)
```

1. Intro

1.1 Background

In a research of breast cancer proteomes, 77 breast cancer samples are generated by the NCI/NIH (Clinical Proteomic Tumor Analysis Consortium) generating 3 datasets. A dataset with expression values, a dataset with genes and proteins used by the PAM50 classification system and a dataset with clinical annotations of the patients.

The 77 breast cancer proteome expression values are log2 iTRAQ ratios for each sample. It contains ~12.000 proteins for each individual sample. This database also contains three samples from healthy individuals.

1.2 Research question

Is it possible to predict a breast cancer stage with machine learning, based of the protein expression data from 77 cancer proteomes?

2. Data

As named before, there are three datasets available. For this research, only two the the three will be used. The expression value data and the clinical patient data. There are 105 clinical patient samples and 77 proteome samples. 28 proteomes were cut of the final data due quality issues.

The goal of this research is to see if it is possible to predict the tumor stage based on the protein expression levels of the patients. The clinical database contain useful annotations of the patients such as:

- ER status (estrogen receptors)

- PR status (progesterone receptors)
- HER2 status (HER2 presence)
- Tumor stage (T1 t/m t4)
- Node stage (N0 t/m N3)
- Metastasis (M0 or M1)
- ACCJ status (summary of above annotations)

All these factors can indicate the cancerous state of the patient. In this case, tumor stage is the most important factor. Because the prediction of this tumor stage will be based of the expression level of the samples.

The tumor stage information runs from T1 to T4, with T1 as the least malicious. With the following classification:

- T1: The tumor in the breast is 20 millimetres (mm) or smaller in size at its widest area
- T2: The tumor is larger than 20 mm but not larger than 50 mm.
- T3: The tumor is larger than 50 mm.
- T4: The tumor has grown into the chest wall and/or skin. Metastasis is likely.

The tumor stage is a good indication of how worse the cancer situation is. Thus, a good variable to predict with the protein expression data.

2.1 load data

The data will be loaded from the Data folder in the repository.

```
# load proteomes data
proteomes <- as.data.frame(read.csv(file = 'Data/cancer_proteomes.csv'))
kable(head(proteomes[, 1:6]), caption = "Proteomes dataset")
```

Table 1: Proteomes dataset

RefSeq_accession_number	gene_symbol	gene_name	AO.A12D.01TCGA	8.A131.01TCGA	AO.A12B.01TCGA
NP_958782	PLEC	plectin isoform 1	1.096131	2.609943	-0.6598280
NP_958785	NA	plectin isoform 1g	1.111370	2.650422	-0.6487422
NP_958786	PLEC	plectin isoform 1a	1.111370	2.650422	-0.6542851
NP_000436	NA	plectin isoform 1c	1.107561	2.646374	-0.6321133
NP_958781	NA	plectin isoform 1e	1.115180	2.646374	-0.6404277
NP_958780	PLEC	plectin isoform 1f	1.107561	2.646374	-0.6542851

The proteomes file contain 86 columns: the first 3 columns with gene information, 80 columns of expression data with TCGA.ID (with 3 duplicates) and 3 columns of expression data of healthy persons.

Also the clinical patient data is loaded in. Containing the tumor stage annotation that will be used.

```
# load clinical patient data
patient_data <- as.data.frame(read.csv(file = "Data/data_patients.csv"))
kable(head(patient_data[, 1:7]), caption = "Clinical patient data")
```

Table 2: Clinical patient data

Complete.TCGA.ID	Gender	Age.at.Initial.Pathologic.Diagnosis	ER.Status	PR.Status	HER2.Final.Status	Tumor
TCGA-A2-A0T2	FEMALE	66	Negative	Negative	Negative	T3
TCGA-A2-A0CM	FEMALE	40	Negative	Negative	Negative	T2
TCGA-BH-A18V	FEMALE	48	Negative	Negative	Negative	T2
TCGA-BH-A18Q	FEMALE	56	Negative	Negative	Negative	T2
TCGA-BH-A0E0	FEMALE	38	Negative	Negative	Negative	T3
TCGA-A7-A0CE	FEMALE	57	Negative	Negative	Negative	T2

2.2 Transform data

To make the proteomes file more workable, the column name with the TCGA.ID will become the row name. Also the first 3 and the last 3 row will be excluded. The first 3 columns contains RefSeq number, gene_symbol and gene name. The RefSeq number will be saved as column name. The last 3 columns contains expression data of healthy people with no ID match to the patients, so this information is not needed, for now.

```
# Transform colnames and rownames

# Save the RefSeq numbers
nm <- proteomes$RefSeq_accession_number

# Exclude col 1-3 and 84 -86
proteomes <- as.data.frame(t(proteomes[,4:86]))
colnames(proteomes) <- nm

# Bind the colnames to the rows
proteomes <- cbind(rownames(proteomes), data.frame(proteomes, row.names=NULL))
colnames(proteomes)[1] <- "Complete.TCGA.ID"
kable(head(proteomes[, 1:6]), caption = "Clinical patient data")
```

Table 3: Clinical patient data

Complete.TCGA.ID	NP_958782	NP_958785	NP_958786	NP_000436	NP_958781
AO.A12D.01TCGA	1.0961312	1.1113704	1.1113704	1.1075606	1.1151802
C8.A131.01TCGA	2.6099430	2.6504218	2.6504218	2.6463739	2.6463739
AO.A12B.01TCGA	-0.6598280	-0.6487422	-0.6542851	-0.6321133	-0.6404277
BH.A18Q.02TCGA	0.1953407	0.2154129	0.2154129	0.2053768	0.2154129
C8.A130.02TCGA	-0.4940596	-0.5038992	-0.5006193	-0.5104589	-0.5038992
C8.A138.03TCGA	2.7650807	2.7797092	2.7797092	2.7979949	2.7870235

Now the dataframe is more workable throughout the analysis.

With 77 proteomes and 105 patients data, ID's must be matched into 1 dataframe. The final product going to be a dataframe with matching tumor stages of the patients with corresponds to the proteomes ID. To

match the ID's of the clinical patient data, the ID of the proteomes data will be transformed. Also the 3 duplicates will be removed.

```
# Make function to transform TGCA.id
clinical.id <- function(proteome.id) {
  x = substr(proteome.id, 4, 7)
  y = substr(proteome.id, 0, 2)
  paste("TCGA",y,x,sep="-")
}

#Sapply to id column in proteomes
proteomes$Complete.TCGA.ID <- sapply(proteomes$Complete.TCGA.ID, clinical.id)
proteomes_new <- proteomes

# Remove duplicates
proteomes_new <- proteomes_new[!duplicated(proteomes_new$Complete.TCGA.ID), ]
kable(head(proteomes_new[, 1:6]), caption = "Transformed proteomes dataset")
```

Table 4: Transformed proteomes dataset

Complete.TCGA.ID	NP_958782	NP_958785	NP_958786	NP_000436	NP_958781
TCGA-AO-A12D	1.0961312	1.1113704	1.1113704	1.1075606	1.1151802
TCGA-C8-A131	2.6099430	2.6504218	2.6504218	2.6463739	2.6463739
TCGA-AO-A12B	-0.6598280	-0.6487422	-0.6542851	-0.6321133	-0.6404277
TCGA-BH-A18Q	0.1953407	0.2154129	0.2154129	0.2053768	0.2154129
TCGA-C8-A130	-0.4940596	-0.5038992	-0.5006193	-0.5104589	-0.5038992
TCGA-C8-A138	2.7650807	2.7797092	2.7797092	2.7979949	2.7870235

Now the proteomes file has the correct ID in the first column, which can be match with the clinical patient data. The next 12.554 columns are the protein expression data of the patients.

2.3 Adding tumor data

Next the tumor stage annotation for the clinical patient data must merge into the proteomes data frame.

```
# Join data frames by ID
proteomes_complete <- merge(patient_data, proteomes_new, BY = 'Complete.TCGA.ID')

# Drop unwanted columns with clinical annotations
proteomes_complete <- proteomes_complete[-c(2:6,8:30)]

# Rename column
colnames(proteomes_complete)[2] <- 'Tumor_Stage'

kable(head(proteomes_complete[, 1:6]), caption = "Added tumor annotation")
```

Table 5: Added tumor annotation

Complete.TCGA.ID	Tumor_Stage	NP_958782	NP_958785	NP_958786	NP_000436
TCGA-A2-A0CM	T2	0.6834035	0.6944241	0.6980976	0.6870771
TCGA-A2-A0D2	T2	0.1074909	0.1041645	0.1074909	0.0975117
TCGA-A2-A0EQ	T2	-0.9126703	-0.9279787	-0.9279787	-0.9318058
TCGA-A2-A0EV	T1	0.4529859	0.4725901	0.4725901	0.4585871
TCGA-A2-A0EX	T3	1.1851082	1.1926120	1.1888601	1.1851082
TCGA-A2-A0EY	T2	1.1748810	1.1832088	1.1832088	1.1748810

Now the Tumor stage annotation is added to the data frame at the second column. Which is necessary for machine learning classification.

2.4 Wide to long format

For many functions R expects data to be in a long format, rather than a wide format. Thus the dataset will be converted to a long format

```
# wide to long format with reshape2 package
pro_long <- melt(proteomes_complete, id.vars = c("Complete.TCGA.ID", "Tumor_Stage"),
variable.name = "Protein_RefSeq")
kable(head(pro_long, 8), caption = "proteomes data in long format")
```

Table 6: proteomes data in long format

Complete.TCGA.ID	Tumor_Stage	Protein_RefSeq	value
TCGA-A2-A0CM	T2	NP_958782	0.6834035
TCGA-A2-A0D2	T2	NP_958782	0.1074909
TCGA-A2-A0EQ	T2	NP_958782	-0.9126703
TCGA-A2-A0EV	T1	NP_958782	0.4529859
TCGA-A2-A0EX	T3	NP_958782	1.1851082
TCGA-A2-A0EY	T2	NP_958782	1.1748810
TCGA-A2-A0SW	T2	NP_958782	-0.4877725
TCGA-A2-A0SX	T1	NP_958782	-0.3985598

Now the data is in a workable long format, that is better to be handled by the package ggplot. This will complete the data formatting.

3. Exploratory data analysis

For a good understanding of the data, is a EDA necessary. EDA stands for exploratory data analysis. In this part of the journal, the formatted data is visualised and explored. By neat and annotated tables and figures, the data will be much more clear and shows new insights.

- Summary
- Missing data
- Distribution
- Tumor stage distribution

3.1 Summary

First, let's look at a summary of all the expression values from the dataset:

```
# Summary all
pander(summary(pro_long$value), caption = 'Total summary')
```

Table 7: Total summary

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
-24.55	-0.922	-0.07428	-0.2092	0.6989	17.62	100622

The median and mean of all the expression values are negative, which means that most proteins are synthesised less. There is a high maximum and a low minimum. These are probably outliers. Also there are 100622 NA's out of 966,581 expressions values, which equals to 10,41%. Next there will be a summary per Tumor stage to see if there are notable differences.

```
# apply a summary per Tumor stage
pander(tapply(pro_long$value, pro_long$Tumor_Stage, summary))
```

- **T1**:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
-19.63	-0.9878	-0.1006	-0.2717	0.6981	17.62	13403

- **T2**:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
-24.55	-0.9046	-0.06929	-0.1869	0.7019	15.44	66912

- **T3**:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
-22.68	-0.9804	-0.09064	-0.3098	0.6694	11.77	13174

- **T4**:

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
-16.36	-0.8491	-0.03962	-0.08701	0.7275	11.93	7133

Same story here, negative means and medians, couple of outliers at the max. and min. No noticeable differences between the stages.

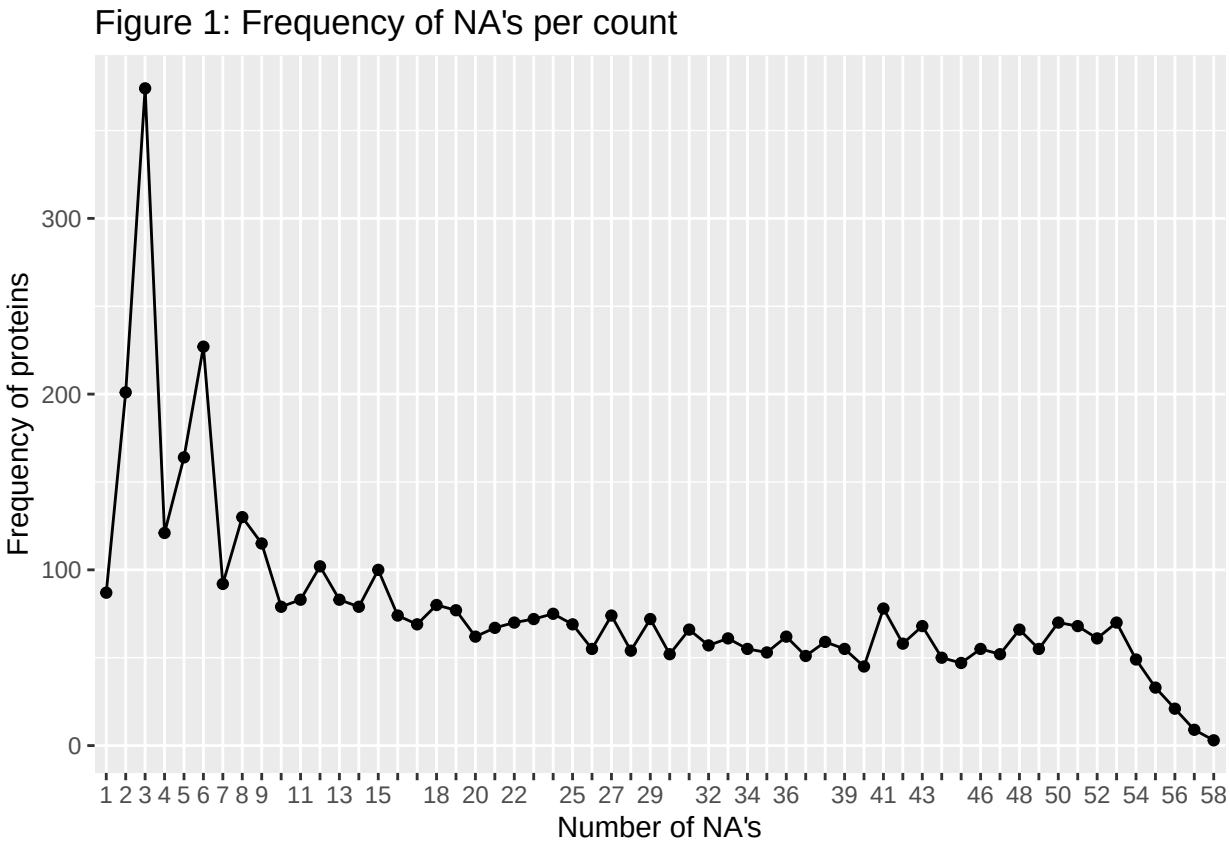
3.2 Missing data

With all big datasets, it is never 100% complete. Also in this dataset, multiple NA's are present. But how much? And which proteins has the most NA's?

```r
#total na's in dataset
tot.na <- sum(is.na(pro_long))
cat("Total number of NA's in dataset: ", tot.na)
```

```
## Total number of NA's in dataset:  100622
```

```r
# calculate na's per protein
na_per_protein <- aggregate(value ~ Protein_RefSeq, data=pro_long, function(x) {sum(is.na(x))}, na.acti
nas <- as.data.frame(table(na_per_protein$value[na_per_protein$value>=1]))


# plot
ggplot(nas, aes(Var1, Freq, group = 1)) + geom_line() + geom_point() +
labs(x = "Number of NA's", y = 'Frequency of proteins', title = "Figure 1: Frequency of NA's per count"
scale_x_discrete(guide = guide_axis(check.overlap = TRUE))
```

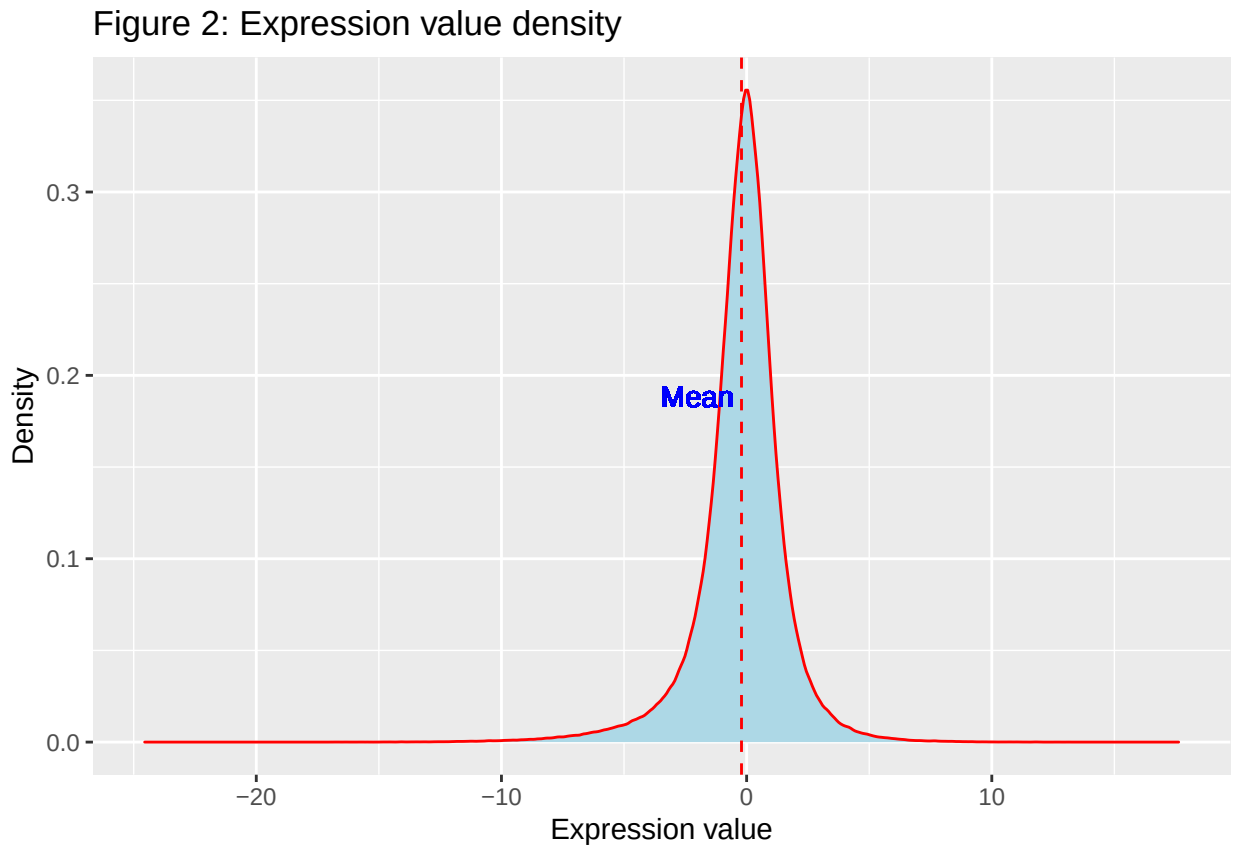


Out of the total 12554 protein in the dataset, 8017 proteins contain zero Na's, which means that 4536 protein have one or more NA's. In the graph is to see that low to medium counts of NA's are no uncommon sight. The graph need to be read as, there are 350+ proteins with 2 NA's. And there are 95 proteins with 7 NA's.

3.3 Distribution

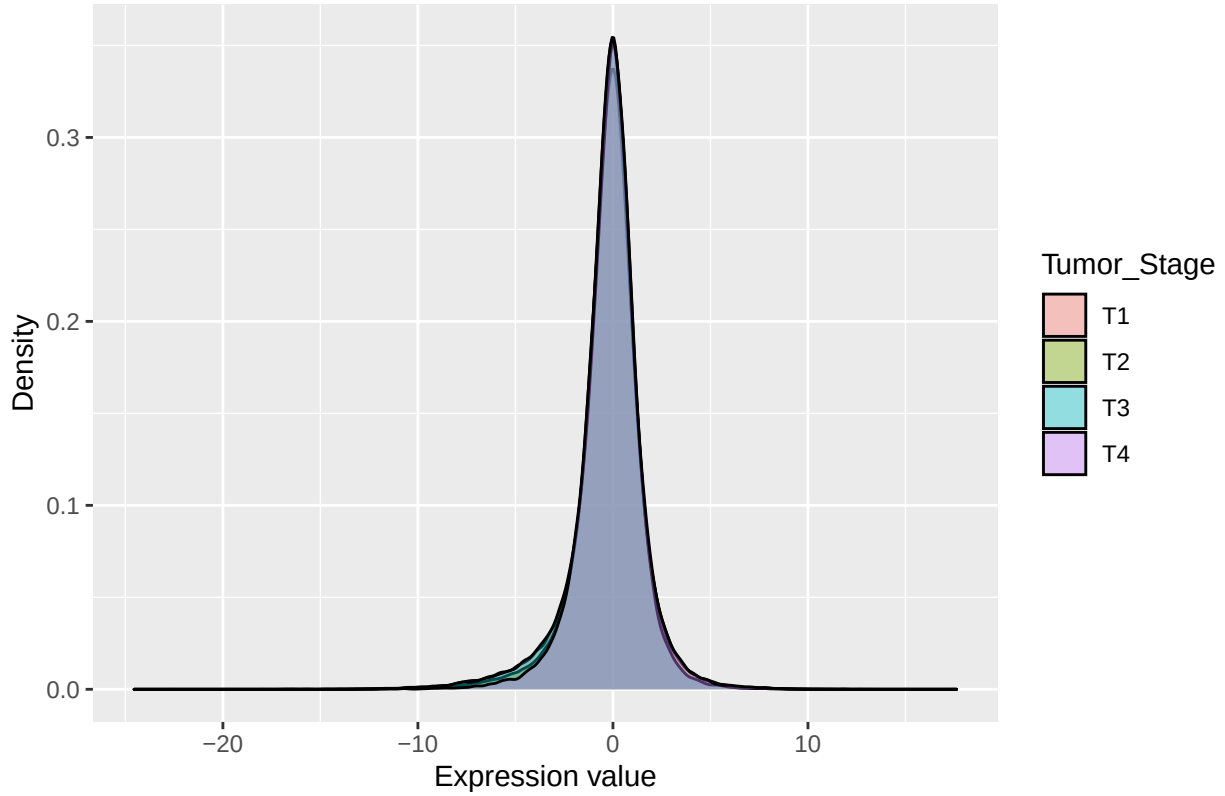
The distribution shows the spread of the data. The distribution can be shown by expression value and per tumor stage.

```r
# plot density expression value
plot1 <- ggplot(pro_long, aes(x = value)) +
         geom_density(color = 'red', fill = 'lightblue') +
         geom_vline(aes(xintercept=mean(value, na.rm=T)),color="red", linetype="dashed", size=0.5) +
         geom_text(aes(x=-2, label="\nMean", y=0.2), colour="blue", angle=0, text=element_text(size=11))
         labs(x = 'Expression value', y = 'Density', title = 'Figure 2: Expression value density')

# plot density by tumor stage
plot2 <- ggplot(pro_long, aes(x = value, group = Tumor_Stage, fill = Tumor_Stage)) +
         geom_density(adjust=1.5, alpha = .4) +
         labs(x = 'Expression value', y = 'Density', title = 'Figure 3: Expression density by tumor stag

plot1
```




```r
plot2
```

Figure 3: Expression density by tumor stage



The above distribution graphs shows very little variation in the data. In the general expression value plot, the peak lay around the 0, with a little more distribution towards the negative.

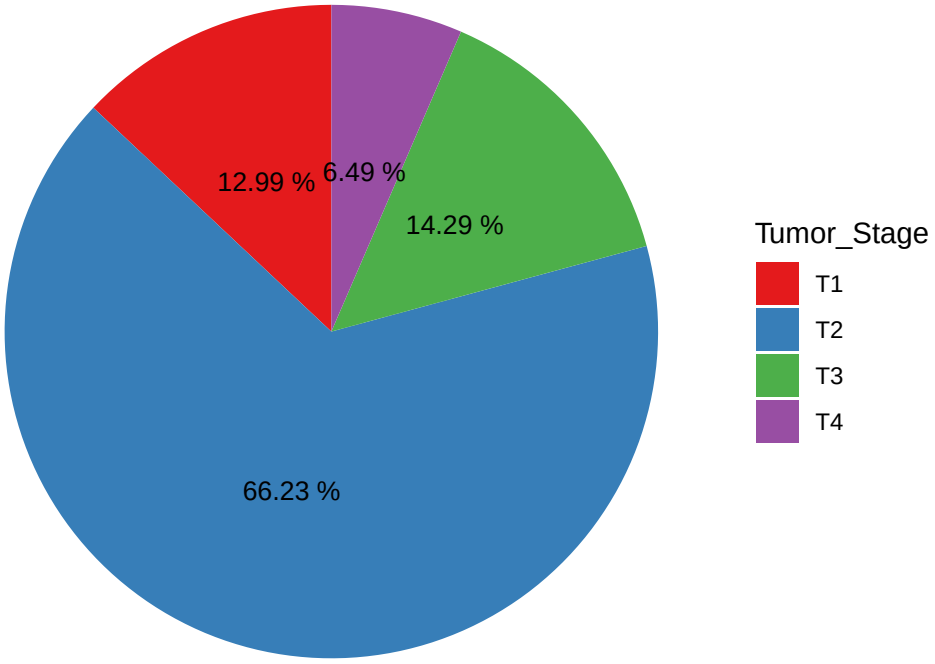
In the tumor stage density, the 4 stages overlap in the plot. But the difference is so little, that the overlap barely can be seen. Which means that there is no notable difference in the total sum of the expression values per tumor stage.

3.4 Tumor stage distribution

Each of the 77 patient is categorised in a tumor group, based of the tumor size and malicious state. But how many patient are in each stage? and how is this divided?

```
# count frequency of tumor groups
stage_count <- proteomes_complete %>%
            select(Tumor_Stage) %>%
            group_by(Tumor_Stage) %>%
            count()

prtage <- paste(round(100*stage_count$n/sum(stage_count$n), 2), "%")

# plot pie chart
ggplot(stage_count, aes(x = '', y = n, fill = Tumor_Stage)) + geom_bar(stat = 'identity', width = 2) + c
theme_void() +  geom_text(aes(label = prtage), position = position_stack(vjust = 0.50), size = 3.5) + sc
```

Figure 4: Percentage of Tumor stages found in patients



Tumor stage T2 is obvious the most common stage. 66% of the patients are in this stage. T1 and T3 share the about the same percentage. T4 is the rare group , with only 6,5% of the patients. Patients in stage T4 have a high mortality rate, so they die early. that's why the T4 percentage probably is so small.

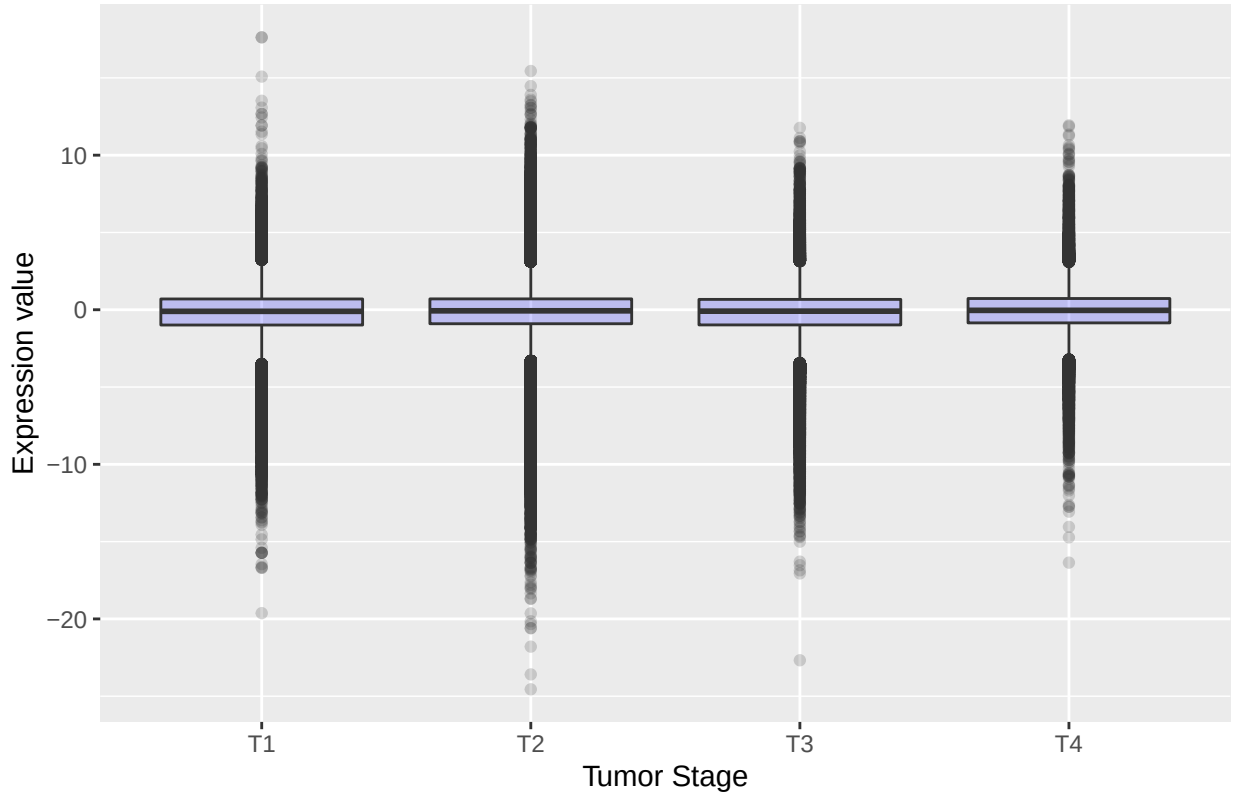
For T1, the tumor is 20 millimetres or smaller in size. thus it is difficult to detect. By stage T2 the tumor is a must more detectable size. therefore that's why T2 is probably more common.

3.5 Outliers

The summary of the tumor stages (table 7) showed nothing but the total spread of the data in table form. With a boxplot the data outliers can be spotted easier.

```
ggplot(pro_long, aes(Tumor_Stage, value)) +
geom_boxplot(fill='blue', alpha = 0.2, na.rm = T) +
labs(x = 'Tumor Stage', y = 'Expression value',title = 'Figure 5: Boxplot summary per Tumor group')
```

Figure 5: Boxplot summary per Tumor group



The spread of the data per group is very wide. In above boxplot, the outliers are clearly visible. This amount of outliers is normal for a dataset like this. As shown in the tumor stage distribution (figure 2/3), the biggest part of the values lies around the means of the group.

4. EDA Result

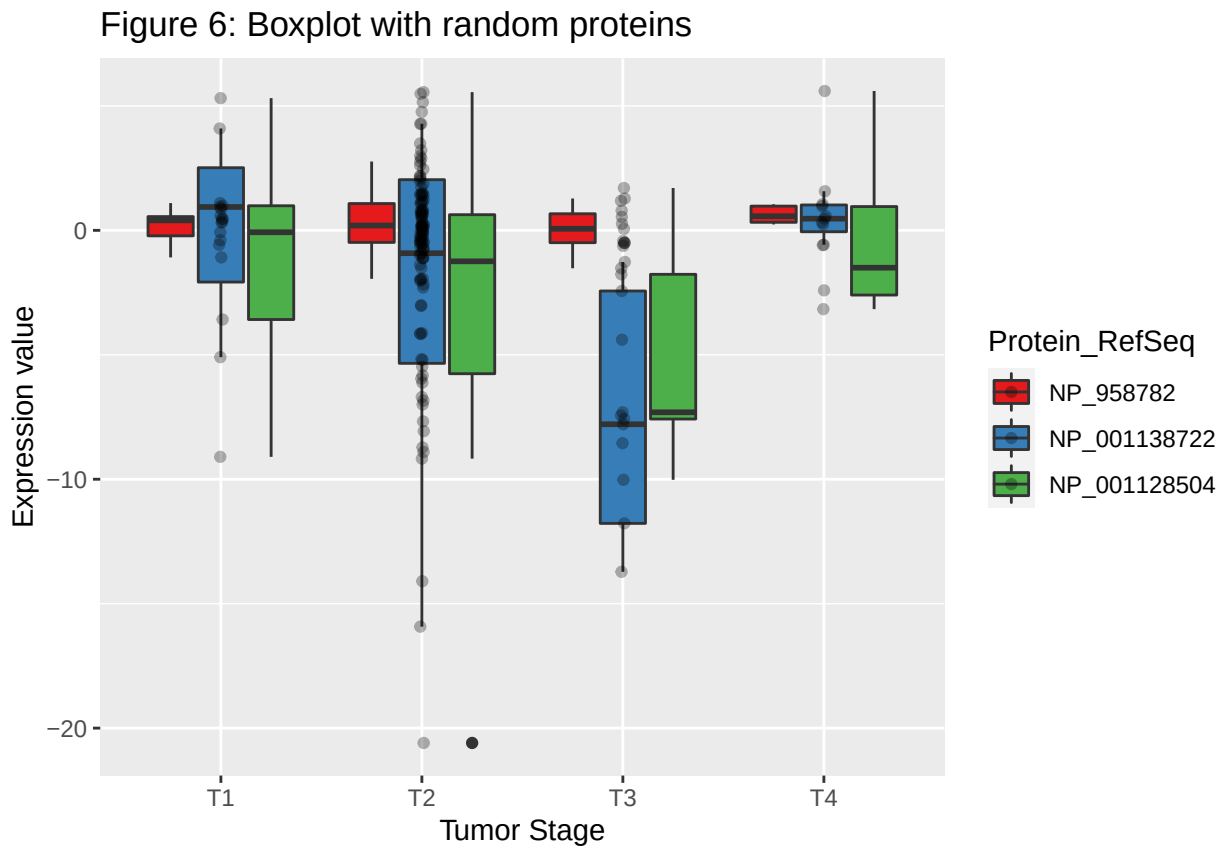
After the EDA, the results of the figures going to be discussed and concluded. In the EDA it was clear that there was not a lot of difference between the Tumor groups based on all the samples. Therefore it is not expected that the research question can be answered with a positive answer based of the EDA only. Further analyse and discussion is necessary for a better conclusion.

4.1 Boxplots per protein

In the figures of the tumor stage distribution, all the proteins were used in the distribution per stage. This gives a overall view of the possible differences, but not per protein. For example a few random protein will be plotted per tumor group.

```
p0 <- pro_long %>%
  filter(Protein_RefSeq %in% c('NP_958782', 'NP_001138722', 'NP_001128504'))

p0 %>%
  ggplot(aes(x = Tumor_Stage, y = value, fill = Protein_RefSeq)) +
  geom_boxplot(na.rm = T) + geom_jitter(width = 0.01, alpha = 0.3) +
```

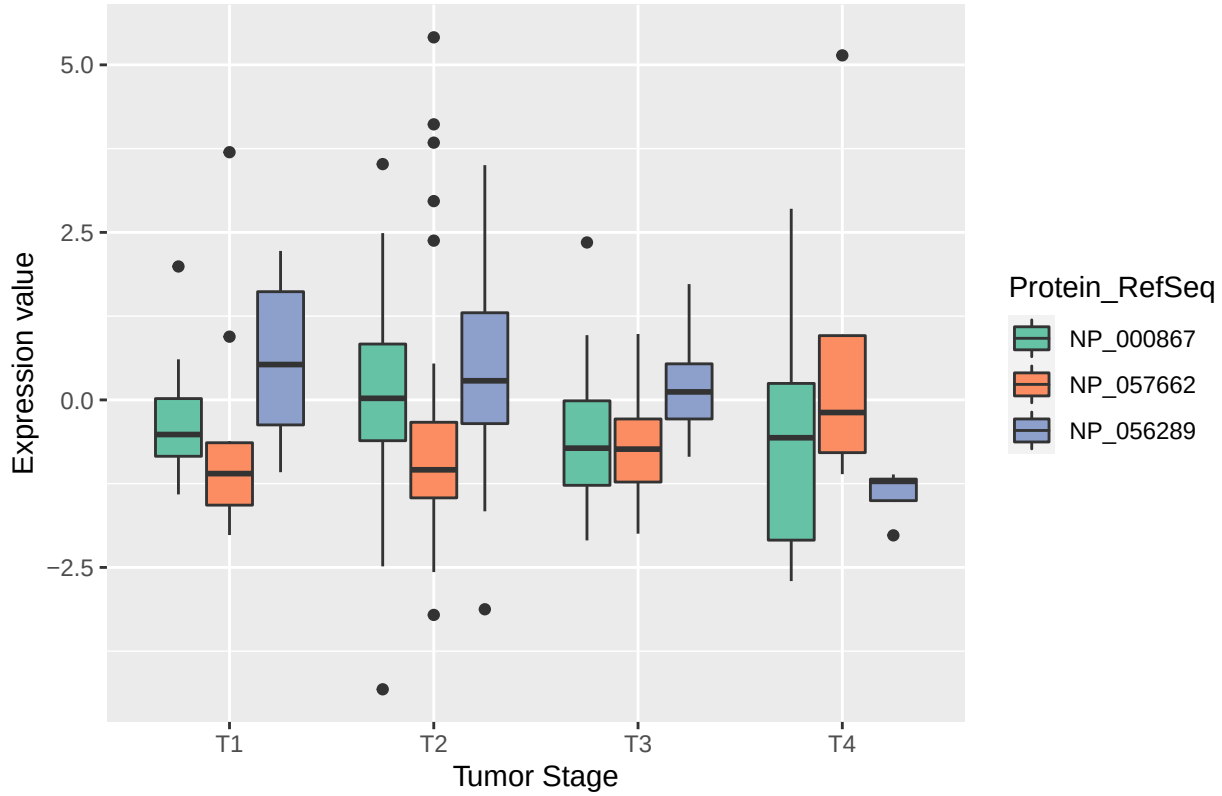
```
  labs(x = 'Tumor Stage', y = 'Expression value', title = 'Figure 6: Boxplot with random proteins') +
  scale_fill_brewer(palette="Set1")
```



As result, with three random protein, three very different distributions. There is not a pattern to find between de groups. Only that they tend to go more towards the negative values, In this example. with three other protein the results can be very different. As demonstrated below.

```
p1 <- pro_long %>%
  filter(Protein_RefSeq %in% c('NP_056289', 'NP_000867', 'NP_057662'))

p1 %>%
  ggplot(aes(x = Tumor_Stage, y = value, fill = Protein_RefSeq)) +
  geom_boxplot(na.rm = T) +
  labs(x = 'Tumor Stage', y = 'Expression value', title = 'Figure 7: Boxplot with random proteins #2') +
  scale_fill_brewer(palette="Set2")
```

Figure 7: Boxplot with random proteins #2



Same here, very different distributions of values. But this time more centred around the mean. Yet there can not be a clear conclusion made based of these boxplot. For a good and accurate answer, there must be a boxplot per tumor group per protein be made and compared to each other. which is maybe not statistic responsible. And if there are protein with clear differences per tumor level, the biological relevance plays also a part.

4.2 Overall result

Overall is the data loaded, transformed and analysed. with mixed results. There was a expectation that the groups overall showed a difference, but they didn't, at least not on the total scale. By analysing per protein, there are more notable differences.

In the EDA is a good view made of the data. By different plots and tables, the data is better understandable. The total summary and summary per tumor group is shown. Total count of NA's and NA count per protein is plotted. The overall distribution and distribution per tumor group. And also the tumor groups per patients is plotted.

5. EDA Discusssion and conclusion

The overall quality of the data is good and the data is broad. But with the research question: 'Is it possible to predict a breast cancer stage, based of the expression data from 77 cancer proteomes?' It is doubtful that it can be answered with a simple 'yes'. Mainly through earlier named reasons.

On the positive side, the data is wide and of good quality. with clear classes and not corrupted, only a few NA's per 100 instances. Without the NA's it is not likely that the results would be different, because there is already much data.

The difficult part with this data (in combination with the research question) is the many protein. there is not a clear trend to be seen is the data, based of the overall analysis. As seen in the results section (4.1), with random proteins the results are more divergent. But based on the data, there is no way to tell which protein is more important that the other. The biological relevance of the proteins is not clear.

As solution for this problem, a literature study of common breast cancer protein could be very useful. By knowing the 100 - 200 most related protein. The research could be narrowed to these proteins. With a new EDA on these filtered data, maybe differences will come on top.

6. Machine learning

The Goal in the machine learning is to predict a Tumor stage based of the protein expression data. By classifying tumor stage with expression values, predictions can be made in which category these values belong. This part covers the cleaning of the data, determine quality metrics and investigate performance of machine learning algorithms.

For the machine learning (ML) a program named Weka will be used. Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. In this analysis Weka version 3.8.5 is used.

6.1 cleaning data

Weka demand files to be a specific file type in order to be accepted. In this project, the data will be cast to a csv file type. To function properly, Weka wants the classifier at the last column for better processing. So the tumor stage will be set to the last column.

Also the NA's will be converted to the mean of that column. Because otherwise Weka will label the columns with NA's as string. The transformed data will be written to a csv file.

The dataset what will be used is the proteomes complete dataset. Before transforming it to the long format.

```
# Relocate Tumor stage to the end
tumTolast <- proteomes_complete %>%
               relocate(Tumor_Stage, .after = last_col())

#Remove NA's
for(i in 1:ncol(tumTolast)){
  tumTolast[is.na(tumTolast[,i]), i] <- mean(tumTolast[,i], na.rm = TRUE)
  }

# write to .csv
write.csv(tumTolast, 'Data/proteome_ML_data.csv')
```

Now the data is properly formatted for Machine learning uses in Weka.

6.2 Determine quality metrics

Beforehand it is useful to know what the quality metrics are for a success. So what quality metrics are important and how are they impact the desired result?

Example of different quality metrics:

- accuracy

- speed
- size of tree
- Confusion matrix

Obviously a high accuracy is desired by all means. Then the question is, how accurate is good enough? As base, a accuracy of 95% is considered good. with deadly diseases like cancers, 80% is not good enough. Which means that one in five is classified in the wrong Tumor stage, which can have serious consequences. By 95% one in twenty is classified wrong.

Speed of the algorithm is important by big usage. Implementing the model in a local environment, 1 to 2 minutes is okay. As it would not calculate more than 10 times a day. But in a large application, an fast algorithm is a demand.

The size of a Decision tree is a important factor. With a large tree with many leave, overfitting could be possible. As the algorithm is to specific on the trainings set. A small, but accurate tree is desirable.

A confusion matrix, is a specific table that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class. This indicates also the rate of true positives, true negatives, false positives and false negatives. Identifying T2 as T4 can cause serious problems. The errors in the matrix will weight heavy on the model performance.

In conclusion: accuracy, confusion matrix and size of tree will weight the heaviest in selecting a good algorithm.

6.3 ML algorithms

With a proper dataset and determined quality metrics,the .csv file is loaded into Weka. The first two rows with ID's are removed, the blank attribute and Complete.TGCA.ID. In the edit viewer, the Attributes are quickly visual scanned, to see if all the labels says numeric.

In The test options, cross-validation is set to 10 Folds.

TP = True positive FP = false positive

Cancer stage distribution:

-T1 = 10 instances
-T2 = 51 instances
-T3 = 11 instances
-T4 = 5 instances

Because of the dominance of T2, many algorithms will tend to choose T2. Because this gives automatically 66,23% accuracy.

6.3.1 Baseline

First ZeroR and OneR are used to measure baseline performance.

```
base <- read.csv('Data/zeroR_oneR_weka.csv', sep = ';', header = T)
pander(base, caption = "Baseline accuracy")
```

Table 12: Baseline accuracy

X.	ZeroR	OneR
Accuracy (%)	66,23	55,8
Speed (sec)	0,01	0,11
T1 TP (%)	0	0
T1 FP (%)	0	13,4
T2 TP (%)	100	84,3
T2 FP (%)	100	84,6
T3 TP (%)	0	0
T3 FP (%)	0	4,5
T4 TP (%)	0	0
T4 FP (%)	0	0

As base, the ZeroR has a accuracy of 66,23%. Which is logical, because zeroR select the most frequent classifier. In this case T2 represents 66,23% of the instances. so ZeroR select T2 as right.

OneR has a accuracy of 55,8%. OneR selects the tumor stage based of single protein value, using cut-off points in the value.

Both simple algorithms gives a good baseline of what the minimum expected accuracy can be. Speed of these algorithms is excellent.

6.3.2 Advanced algoritmhs

With a set baseline of 66,23%, more advanced algorithms are used. These advanced algorithms consists of: decisions trees, Bayes and logics.

Advanced algorithms:

- J48
- Random Forest
- Naïve Bayes
- Simple Logistic
- SVM (SMO)
- K-Nearest Neighbour (IBk)

All those algorithms will be recorded, same as the baseline. With the default options.

```
advan_tbl <- read.csv('Data/advanced_algorithms_weka.csv', sep = ';')
colnames(advan_tbl)[1] <- '.'
pander(advan_tbl, caption = "Advanced algorithms")
```

Table 13: Advanced algorithms

.	J48	Random.Fortest	Naïve.Bayes	Simple.Log.	SMO	IB1
Accuracy (%)	33,7	66,23	64,93	66,23	66,33	51,4
Speed (sec)	0,47	0,87	0,1	2,35	0,33	0,04
T1 TP (%)	0	0	0	0	0	20
T1 FP (%)	16,4	0	3	0	6	14,9
T2 TP (%)	47,1	100	98	100	92	64,7

.	J48	Random.Fortest	Naïve.Bayes	Simple.Log.	SMO	IB1
T2 FP (%)	50	100	92,3	100	76,9	46,2
T3 TP (%)	18,2	0	0	0	9,1	18,2
T3 FP (%)	31,8	0	1,5	0	7,6	18,2
T4 TP (%)	0	0	0	0	0	51,9
T4 FP (%)	40,3	0	0	0	0	35,4

In above table, six algorithms are used on the dataset. with mixed results.

looking at accuracy, Random Forest, simple logistic and SMO preform equally. With naïve bayes not fall far behind. They all around the 66% mark, same as ZeroR. But they all acquire it on a different way. Random forest and simple logistic have the exact same accuracy and TP/FP rate. only a speed difference, with is not big.

Naïve bayes is close to the baseline, and identifies the most as T2, true and false. but with a little classifying spread. SMO has the bes accuracy with 66,33% and identifies also a small part of t3 correct. both of these algorithms are very fast.

J48 has only a accuracy of 33,7%, which is disappointing. But, J48 has a large spread between the Tumor stages. only 47,1% of T2 is correct, yet other stages are identified positive or negative.

Same for IB1 as J48, low accuracy, big spread between tumor stages. And IB1 correctly identifies more Stages then J48. IB1 as in all 4 Tumor stages a true positive rate of ~20%. Which is the best out of the six algorithms.

Considering the results of all these algorithms, not one stands really out. Looking at accuracy: Random forest, Naïve bayes, simple logistic and SMO preform somewhat equally. All the algorithms are fast, with simple logistic as slowest. And IB1 has the most spread between the tumor stages.

7. Preforming protein filtering

With so many proteins, not many difference are able to be seen. In the EDA there was not much variation in the whole data set or between the tumor stages. In the results, random proteins showed difference, so there is difference in the dataset. By selecting a algorithm for the machine learning, 66% was the highest possible. Manley of all the many proteins.

7.1 Different expressed proteins

Not a these proteins are significant different form each other. Probably a few 1000 proteins show different expressions. These Different expressed proteins are good targets for a better dataset, because many proteins are irrelevant if they show little or no variation. A Different expressed protein is a protein that shows a significant different expression level relative to the other values across the tumor stages.

To highlight and filter out these Different expressed proteins, a couple of coding steps will be made. For reference the data of the healthy patients will be added by the dataset.

```
# select the row of the healthy patients and assign tumor stage 'T0'to them
healthy <- proteomes_new[78:80,]
stages <- c('T0', 'T0', 'T0')
healthy$Tumor_Stage <- stages
healthy <- healthy %>%
  relocate(Tumor_Stage, .after = 1)
```

```r
# bind the original and new rows together
proteomes_with_healthy <- rbind(proteomes_complete, healthy)
kable(head(proteomes_with_healthy[75:80,1:6]), caption = "Added healthy persons")
```

Table 14: Added healthy persons

	Complete.TCGA.ID	Tumor_Stage	NP_958782	NP_958785	NP_958786	NP_000436
75	TCGA-E2-A154	T1	0.8626593	0.8701860	0.8701860	0.8664226
76	TCGA-E2-A158	T1	-1.0865291	-1.0954924	-1.0954924	-1.0954924
77	TCGA-E2-A15A	T2	2.1801233	2.1801233	2.1801233	2.1801233
81	TCGA-X2-3d3f	T0	0.5985845	0.6066975	0.6039931	0.6039931
82	TCGA-bl-db9.	T0	-0.1912845	-0.1839177	-0.1860225	-0.1860225
83	TCGA-c4-55b.	T0	0.5669753	0.5787017	0.5767473	0.5767473

The healthy patients are labelled as T0, because they didn't have a tumor. The dataset ought to be 'prepared' by removing the NA's and transform the Tumor stage to a factor.

```r
# Remove the NA's and replace them with the mean of that column
for(i in which(sapply(proteomes_with_healthy, is.numeric))){
  proteomes_with_healthy[is.na(proteomes_with_healthy[,i]), i] <- mean(proteomes_with_healthy[,i], na.r
}

# Cast Tumor stage as factor and relocate to last column
proteomes_with_healthy <- proteomes_with_healthy %>%
                          mutate(Tumor_Stage = factor(Tumor_Stage)) %>%
                          relocate(Tumor_Stage, .after=last_col())
```

With a ready dataset, the protein with significant different expressions will be filtered out. A more strict filtering with p-values is chosen, Instead of the standard 0.05, because this results in a more select group of proteins.

```r
# Filter out the different proteins with a kruskal-Wallis test and  save the P-value to a list
res <- list()
for(name in colnames(proteomes_with_healthy)[2:(ncol(proteomes_with_healthy)-1)]) {
  f <- paste0(name, ' ~ Tumor_Stage')
  pval <- kruskal.test(formula = formula(f), data = proteomes_with_healthy)$p.value
  if(pval < 0.01){
    res[[name]] <- p.adjust(pval, method = 'bonferroni')
  }
}

# filter out the different proteins
p.values <- t(unlist(res))
cat("Proteins left = ",length(p.values))
```

```
## Proteins left =  210
```

```r
filtered_proteomes <- proteomes_with_healthy[,c(colnames(p.values))]
```

With a p-value of 0.01 210 protein will be left, which is still many. With a small p-value of 0.01, 210 proteins are still in the dataset. 210 is enough and strict.

As last, the Id and tumor stage are put back in place.

```
# add ID back
ID <- proteomes_with_healthy[,1]
filtered_proteomes <- cbind(filtered_proteomes, ID)
colnames(filtered_proteomes)[ncol(filtered_proteomes)] <- "Complete.TCGA.ID"
filtered_proteomes <-  filtered_proteomes %>%
                          relocate(Complete.TCGA.ID, .before = 1)

# Add Tumor stage back
Tum <- proteomes_with_healthy[,ncol(proteomes_with_healthy)]
filtered_proteomes <- cbind(filtered_proteomes, Tum)
colnames(filtered_proteomes)[ncol(filtered_proteomes)] <- "Tumor_Stage"
filtered_proteomes <- filtered_proteomes %>%
                          mutate(Tumor_Stage = factor(Tumor_Stage))
```

The Data now only consist of 210 proteins instead of 12.554. With these significant different proteins the algorithms will be run again. The filtered dataset is written to a csv file

```
# Write to csv, removes healthy patients for classifying
write.csv(filtered_proteomes[1:77,], 'Data/filtered_proteome_ML_data.csv')
```

7.2 ML algorithms with filtered dataset

With a filtered dataset the same algorithms will be used to see the difference.

7.2.1 Baseline

As base, ZeroR and OneR are used again

```
# load algorithm results
base_filtered <- read.csv('Data/zeroR_oneR_weka_filtered.csv', sep = ';', header = T)

grid.arrange(
  tableGrob(base),
  tableGrob(base_filtered),
  nrow = 1,
  top = textGrob('Normal vs filtered')
)
```

Normal vs filtered

	X.	ZeroR	OneR
1	Accuracy (%)	66,23	55,8
2	Speed (sec)	0,01	0,11
3	T1 TP (%)	0	0
4	T1 FP (%)	0	13,4
5	T2 TP (%)	100	84,3
6	T2 FP (%)	100	84,6
7	T3 TP (%)	0	0
8	T3 FP (%)	0	4,5
9	T4 TP (%)	0	0
10	T4 FP (%)	0	0

	X.	ZeroR	OneR
1	Accuracy (%)	66,23	57,14
2	Speed (sec)	0.01	0.02
3	T1 TP (%)	0	0
4	T1 FP (%)	0	0
5	T2 TP (%)	100	86,3
6	T2 FP (%)	100	100
7	T3 TP (%)	0	0
8	T3 FP (%)	0	7,6
9	T4 TP (%)	0	0
10	T4 FP (%)	0	0

ZeroR don't preform better then before, with the exact same statistics. OneR does it slightly better with a ~2% accuracy increase. Therefore it looks likes a improvement, at a base level.

7.2.2 Advanced algorithms

Same story here, same algorithms as with the unfiltered dataset.

```
# load algorithm results
advanced_filtered <- read.csv('Data/advanced_algorithms_weka_filtered.csv', sep = ';')
colnames(advanced_filtered)[1] <- '.'


pander(advan_tbl, caption = 'original')
```

Table 15: original

.	J48	Random.Fortest	Naïve.Bayes	Simple.Log.	SMO	IB1
Accuracy (%)	33,7	66,23	64,93	66,23	66,33	51,4
Speed (sec)	0,47	0,87	0,1	2,35	0,33	0,04
T1 TP (%)	0	0	0	0	0	20
T1 FP (%)	16,4	0	3	0	6	14,9
T2 TP (%)	47,1	100	98	100	92	64,7
T2 FP (%)	50	100	92,3	100	76,9	46,2
T3 TP (%)	18,2	0	0	0	9,1	18,2

.	J48	Random.Fortest	Naïve.Bayes	Simple.Log.	SMO	IB1
T3 FP (%)	31,8	0	1,5	0	7,6	18,2
T4 TP (%)	0	0	0	0	0	51,9
T4 FP (%)	40,3	0	0	0	0	35,4

```
pander(advanced_filtered, caption = 'filtered')
```

Table 16: filtered

.	J48	Random.Fortest	Naïve.Bayes	Simple.Log.	SMO	Ibk
Accuracy (%)	49,35	70,12	64,93	74,02	81,81	75,32
Speed (sec)	0.03	0.09	0.01	0.12	0.02	0.1
T1 TP (%)	10	30	30	50	50	30
T1 FP (%)	13,4	1,5	10,4	6	3	3
T2 TP (%)	62,7	100	90,2	90,2	96,1	94,1
T2 FP (%)	50	84,6	65,4	50	38,5	53,8
T3 TP (%)	27,3	0	9,1	27,3	54,5	27,3
T3 FP (%)	18,2	0	4,5	4,5	3	3
T4 TP (%)	40	0	0	60	60	80
T4 FP (%)	37,9	0	0	0	0	1,4

In general the accuracy is strongly improved, with the most difference in SMO. SMO preforms good with a accuracy of 81%. which means that it preforms well also in other stages than T2. looking at the spread of the TP and FP between the stages, SMO also wins relative to IBK and simple Logistic. SMO is the clear winner here.

8. Final result

After the modification of the dataset, EDA, algorithm testing, data cleaning and algorithm testing with cleaned data. SMO is the final outcome of the machine learning phases. This algorithm can predict out of 4 cancerous stages the correct stage with a 81% accuracy, based of 210 breast cancer related proteins of 77 patients. with references of 3 healthy patients.

But, as said in the determine quality metrics: 'with deadly diseases like cancers, 80% is not good enough. Which means that one in five is classified in the wrong Tumor stage, which can have serious consequences'. This is true, but with optimisation 81% is the highest accuracy possible.

The research question: 'Is it possible to predict a breast cancer stage with machine learning, based of the protein expression data from 77 cancer proteomes?. Yes and No. It is possible to predict a tumor stage with machine learning. But, it can only predicted with 81% accuracy. Thus, in serious applications, it is useless. Because of the low accuracy. But, with a general practitioner, it can be useful for a prediction. That a patient is possible in a certain tumor stage.

The answer of the research question depends on the application of the model.