# Initial Data Wrangling and Modelling

*Igor Hut*

*26 april 2016*

This document contains exploratory data analysis and initial tryouts of different ML algorithms for classification of the probationary data set for my PhD thesis.

For the sake of transparency and reproducibility all the used R code will be included.

## Exploratory Data Analysis

Raw data set was provided by prof. Koruga and A. Dragicevic in the form of an MS Excel file `IGOR HUT TTRPSouthend2015.xlsx`, which contains tagged results obtained by OMS recording of tissue samples.

### Initial settings and data import

```
library(readxl)
library(ggplot2)
library(ggthemes)
library(dplyr)
library(tidyr)
library(matrixStats)
library(caret)
library(AppliedPredictiveModeling)
date()
```

```
## [1] "Tue Apr 26 14:43:35 2016"
```

```
setwd("~/GitHub/PhD")

initData<-read_excel("IGOR HUT TTRPSouthend2015.xlsx")

initData # Checking the tbl
```

```
## Source: local data frame [166 x 260]
##
##           Plate No Device (1-Edmund, 2-RasPi, 3-Canon)     Case   100
##    (dbl)     (chr)                                (chr)    (chr) (dbl)
## 1      1 G20706/14                  2015-11-17_10-49 Negative     0
## 2      2 G20739/14                  2015/11/17_10-51 Negative     0
## 3      3 G20786/14                  2015-11-17_10-55 Negative     0
## 4      4 G20798/14                  2015-11-17_10-57 Negative     0
## 5      5 G20849/14                  2015/11/17_10-58 Negative     0
## 6      6 G20880/14                  2015-11-17_11-00 Negative     0
## 7      7 G20815/14                  2015-11-17_11-02 Negative     0
## 8      8 G20781/14                  2015-11-17_11-03 Negative     0
## 9      9 G20883/14                  2015-11-17_11-05 Negative     0
## 10    10 G20694/14                  2015-11-17_11-15 Negative     0
```

```
## ..   ...       ...                                      ...      ...   ...
## Variables not shown: 100.125 (dbl), 100.31 (dbl), 100.435 (dbl), 100.563
##   (dbl), 100.754 (dbl), 100.947 (dbl), 101.139 (dbl), 101.331 (dbl),
##   101.52200000000001 (dbl), 101.715 (dbl), 101.907 (dbl), 102.099 (dbl),
##   102.288 (dbl), 102.477 (dbl), 102.67400000000001 (dbl),
##   102.86499999999999 (dbl), 103.054 (dbl), 103.251 (dbl), 103.443 (dbl),
##   103.634 (dbl), 103.82599999999999 (dbl), 104.018 (dbl), 104.209 (dbl),
##   104.398 (dbl), 104.593 (dbl), 104.785 (dbl), 104.976 (dbl),
##   105.16500000000001 (dbl), 105.354 (dbl), 105.547 (dbl), 105.745 (dbl),
##   105.937 (dbl), 106.129 (dbl), 106.319 (dbl), 106.512 (dbl), 106.705
##   (dbl), 106.89700000000001 (dbl), 107.086 (dbl), 107.27500000000001
##   (dbl), 107.47199999999999 (dbl), 107.663 (dbl), 107.852 (dbl), 108.048
##   (dbl), 108.24 (dbl), 108.432 (dbl), 108.624 (dbl), 108.816 (dbl),
##   109.00700000000001 (dbl), 109.196 (dbl), 109.39100000000001 (dbl),
##   109.583 (dbl), 109.773 (dbl), 109.962 (dbl), 110.158 (dbl), 110.348
##   (dbl), 110.54300000000001 (dbl), 110.735 (dbl), 110.92700000000001
##   (dbl), 111.117 (dbl), 111.31 (dbl), 111.502 (dbl), 111.694 (dbl),
##   111.864 (dbl), 112.07299999999999 (dbl), 112.27 (dbl), 112.461 (dbl),
##   112.65 (dbl), 112.846 (dbl), 113.038 (dbl), 113.229 (dbl),
##   113.42100000000001 (dbl), 113.614 (dbl), 113.80500000000001 (dbl),
##   113.994 (dbl), 114.18899999999999 (dbl), 114.381 (dbl), 114.571 (dbl),
##   114.76 (dbl), 114.949 (dbl), 115.142 (dbl), 115.34099999999999 (dbl),
##   115.533 (dbl), 115.72499999999999 (dbl), 115.86499999999999 (dbl),
##   116.036 (dbl), 116.241 (dbl), 116.423 (dbl), 116.637 (dbl), 116.87
##   (dbl), 117.07899999999999 (dbl), 117.33 (dbl), 117.52800000000001 (dbl),
##   117.73699999999999 (dbl), 117.99299999999999 (dbl), 118.199 (dbl),
##   118.455 (dbl), 118.712 (dbl), 118.967 (dbl), 119.22 (dbl), 119.48 (dbl),
##   119.736 (dbl), 119.991 (dbl), 120.245 (dbl), 120.498 (dbl), 120.708
##   (dbl), 120.968 (dbl), 121.224 (dbl), 121.48 (dbl), 121.735 (dbl),
##   121.992 (dbl), 122.249 (dbl), 122.505 (dbl), 122.764 (dbl), 122.973
##   (dbl), 123.343 (dbl), 123.57899999999999 (dbl), 123.83199999999999
##   (dbl), 124.161 (dbl), 124.43600000000001 (dbl), 124.70699999999999
##   (dbl), 124.98099999999999 (dbl), 125.26 (dbl), 125.497 (dbl), 125.73
##   (dbl), 126.10599999999999 (dbl), 126.32299999999999 (dbl),
##   126.55800000000001 (dbl), 126.779 (dbl), 127.10899999999999 (dbl),
##   127.32899999999999 (dbl), 127.60599999999999 (dbl), 127.886 (dbl),
##   128.10400000000001 (dbl), 128.37299999999999 (dbl), 128.715 (dbl),
##   129.07300000000001 (dbl), 129.36699999999999 (dbl), 129.66900000000001
##   (dbl), 129.96899999999999 (dbl), 130.267 (dbl), 130.46299999999999
##   (dbl), 130.76599999999999 (dbl), 131.065 (dbl), 131.393 (dbl),
##   131.70599999999999 (dbl), 131.88900000000001 (dbl), 132.26300000000001
##   (dbl), 132.55699999999999 (dbl), 133.01 (dbl), 133.28299999999999 (dbl),
##   133.67699999999999 (dbl), 134.06299999999999 (dbl), 134.464 (dbl),
##   134.85499999999999 (dbl), 135.24299999999999 (dbl), 135.61799999999999
##   (dbl), 136.024 (dbl), 136.43 (dbl), 136.797 (dbl), 137.221 (dbl),
##   137.61500000000001 (dbl), 138.01 (dbl), 138.40299999999999 (dbl),
##   138.797 (dbl), 139.19 (dbl), 139.584 (dbl), 139.97900000000001 (dbl),
##   140.37200000000001 (dbl), 140.76599999999999 (dbl), 141.16 (dbl),
##   141.51400000000001 (dbl), 141.90899999999999 (dbl), 142.30199999999999
##   (dbl), 142.696 (dbl), 143.089 (dbl), 143.483 (dbl), 143.875 (dbl),
##   144.27099999999999 (dbl), 144.66499999999999 (dbl), 145.05600000000001
##   (dbl), 145.453 (dbl), 145.84700000000001 (dbl), 146.203 (dbl),
##   146.79400000000001 (dbl), 147.142 (dbl), 147.65100000000001 (dbl),
##   148.15899999999999 (dbl), 148.667 (dbl), 149.101 (dbl), 149.613 (dbl),
```

```
##   150.285 (dbl), 150.791 (dbl), 151.328 (dbl), 151.827 (dbl),
##   152.29499999999999 (dbl), 152.78299999999999 (dbl), 153.31200000000001
##   (dbl), 153.91900000000001 (dbl), 154.45699999999999 (dbl), 154.995
##   (dbl), 155.70400000000001 (dbl), 156.24199999999999 (dbl),
##   156.74700000000001 (dbl), 157.233 (dbl), 157.786 (dbl),
##   158.40299999999999 (dbl), 158.994 (dbl), 159.608 (dbl),
##   160.19300000000001 (dbl), 160.845 (dbl), 161.554 (dbl),
##   162.14699999999999 (dbl), 162.80099999999999 (dbl), 163.44399999999999
##   (dbl), 164.16900000000001 (dbl), 164.971 (dbl), 165.63499999999999
##   (dbl), 166.34800000000001 (dbl), 167.16200000000001 (dbl), 167.886
##   (dbl), 168.411 (dbl), 169.29900000000001 (dbl), 169.89599999999999
##   (dbl), 170.624 (dbl), 171.43100000000001 (dbl), 172.16300000000001
##   (dbl), 173.113 (dbl), 174.05799999999999 (dbl), 174.82499999999999
##   (dbl), 176.12299999999999 (dbl), 177.00899999999999 (dbl), 178.005
##   (dbl), 179.256 (dbl), 180.43100000000001 (dbl), 181.83099999999999
##   (dbl), 183.22900000000001 (dbl), 184.648 (dbl), 186.36500000000001
##   (dbl), 187.822 (dbl), 189.40899999999999 (dbl), 191.21100000000001
##   (dbl), 193.21199999999999 (dbl), 195.43199999999999 (dbl),
##   197.32900000000001 (dbl), 200.30699999999999 (dbl), 202.774 (dbl),
##   205.60599999999999 (dbl), 209.608 (dbl), 215.30099999999999 (dbl),
##   223.333 (dbl), 236.666 (dbl), 250 (dbl), 265 (dbl), 280 (dbl), 300 (dbl)
```

```
useData<-initData[-(1:3)] # First three columns are not needed for initial analysis

useData[1:10] # Checking first columns of useData
```

```
## Source: local data frame [166 x 10]
##
##        Case   100 100.125 100.31 100.435 100.563 100.754 100.947 101.139
##       (chr) (dbl)   (dbl)  (dbl)   (dbl)   (dbl)   (dbl)   (dbl)   (dbl)
## 1  Negative     0       0      0       0       0       0       0       0
## 2  Negative     0       0      0       0       0       0       0       0
## 3  Negative     0       0      0       0       0       0       0       0
## 4  Negative     0       0      0       0       0       0       0       0
## 5  Negative     0       0      0       0       0       0       0       0
## 6  Negative     0       0      0       0       0       0       0       0
## 7  Negative     0       0      0       0       0       0       0       0
## 8  Negative     0       0      0       0       0       0       0       0
## 9  Negative     0       0      0       0       0       0       0       0
## 10 Negative     0       0      0       0       0       0       0       0
## ..      ...   ...     ...    ...     ...     ...     ...     ...     ...
## Variables not shown: 101.331 (dbl)
```

```
# Let's make factors out of chr markings

useData$Case<-as.factor(useData$Case)

# Check the outcome

levels(useData$Case)
```

```
##  [1] "BL"                          "CANCER STAGE 1A"
##  [3] "CANCER STAGE 1A "            "CANCER STAGE 1A broken glass"
```

```
##  [5] "CANCER STAGE 1B"              "CIN2&HG"
##  [7] "LG"                           "MD"
##  [9] "Negative"                     "SD"
```

```
# Putting all the CANCER* factor levels iside one CANCER wraper

levels(useData$Case)<-sub("CANCER.*", "CANCER", levels(useData$Case))

# Check the situation

levels(useData$Case)
```

```
## [1] "BL"       "CANCER"   "CIN2&HG"  "LG"       "MD"       "Negative"
## [7] "SD"
```

```
summary(useData$Case)
```

```
##       BL   CANCER  CIN2&HG       LG       MD Negative       SD
##       27       13       26       26       20       27       27
```

```
# Zeroes are a huge problem in terms of calculating mean, sd etc. so let's get rid of them
# Keeping only the columns with abs(colSum)>=0.5

# useData1<-useData[,(colSums(abs(useData[2:length(useData)])))>=0.5]

useData1<-useData[,(colSums(abs(useData[2:length(useData)])))>=5]

# Putting the first column back

useData1<-cbind(useData[1],useData1)

# Making a table again

useData1<-tbl_df(useData1)

useData1
```

```
## Source: local data frame [166 x 121]
##
##        Case  115.533 115.72499999999999 115.86499999999999  116.036
##      (fctr)    (dbl)              (dbl)              (dbl)    (dbl)
## 1  Negative  0.00000            0.00000            0.00000  0.00000
## 2  Negative -0.00043           -0.00057           -0.00057 -0.00043
## 3  Negative -0.00071           -0.00071           -0.00071 -0.00029
## 4  Negative -0.00071           -0.00014           -0.00071  0.00014
## 5  Negative -0.00029           -0.00071           -0.00029 -0.00071
## 6  Negative  0.00014            0.00014           -0.00014  0.00000
## 7  Negative -0.00071           -0.00043           -0.00043 -0.00214
## 8  Negative -0.00071           -0.00143           -0.00200 -0.00171
## 9  Negative -0.00057           -0.00043           -0.00029 -0.00071
## 10 Negative -0.00014           -0.00043           -0.00043 -0.00071
## ..      ...      ...                ...                ...      ...
```

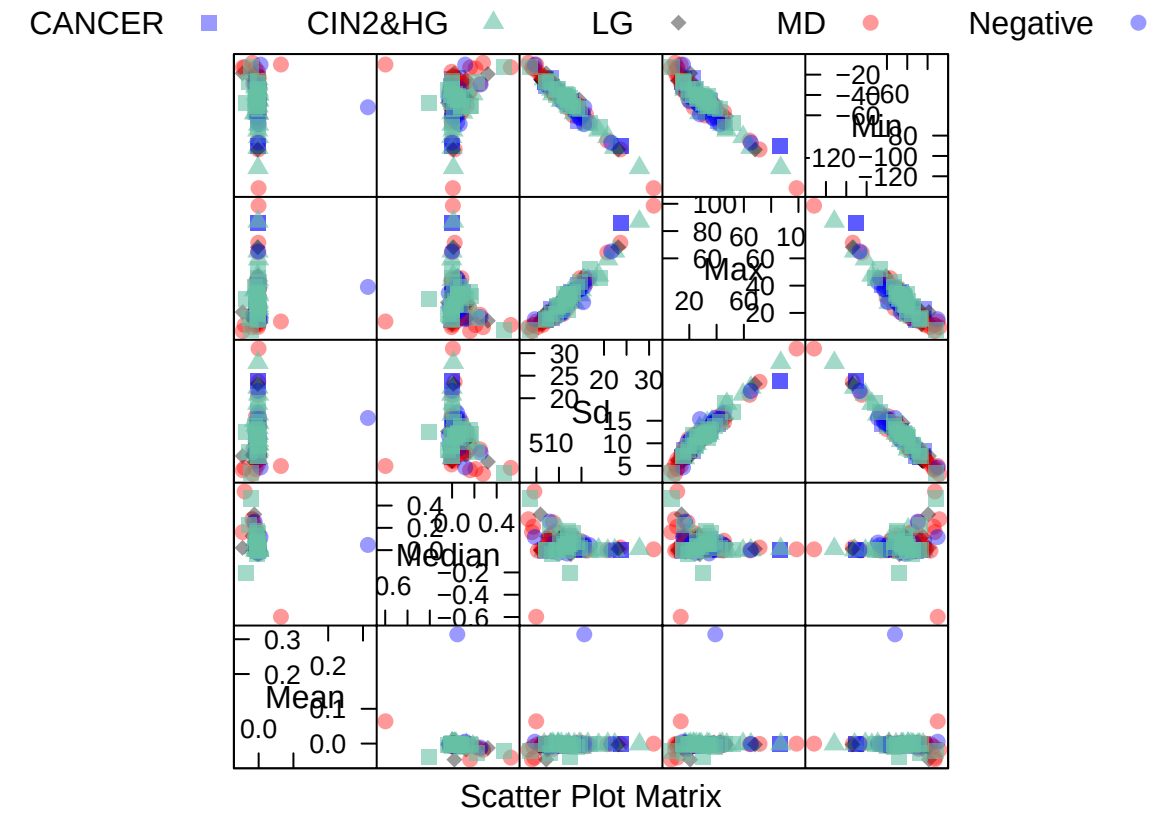
```
## Variables not shown: 116.241 (dbl), 116.423 (dbl), 116.637 (dbl), 116.87
##   (dbl), 117.07899999999999 (dbl), 117.33 (dbl), 117.52800000000001 (dbl),
##   117.73699999999999 (dbl), 117.99299999999999 (dbl), 118.199 (dbl),
##   118.455 (dbl), 118.712 (dbl), 118.967 (dbl), 119.22 (dbl), 119.48 (dbl),
##   119.736 (dbl), 119.991 (dbl), 120.245 (dbl), 120.498 (dbl), 120.708
##   (dbl), 120.968 (dbl), 121.224 (dbl), 121.48 (dbl), 121.735 (dbl),
##   121.992 (dbl), 122.249 (dbl), 122.505 (dbl), 122.764 (dbl), 122.973
##   (dbl), 123.343 (dbl), 123.57899999999999 (dbl), 123.83199999999999
##   (dbl), 124.161 (dbl), 124.43600000000001 (dbl), 124.70699999999999
##   (dbl), 124.98099999999999 (dbl), 125.26 (dbl), 125.497 (dbl), 125.73
##   (dbl), 126.10599999999999 (dbl), 126.32299999999999 (dbl),
##   126.55800000000001 (dbl), 126.779 (dbl), 127.10899999999999 (dbl),
##   127.32899999999999 (dbl), 127.60599999999999 (dbl), 127.886 (dbl),
##   128.10400000000001 (dbl), 128.37299999999999 (dbl), 128.715 (dbl),
##   129.07300000000001 (dbl), 129.36699999999999 (dbl), 129.66900000000001
##   (dbl), 129.96899999999999 (dbl), 130.267 (dbl), 130.46299999999999
##   (dbl), 130.76599999999999 (dbl), 131.065 (dbl), 131.393 (dbl),
##   131.70599999999999 (dbl), 131.88900000000001 (dbl), 132.26300000000001
##   (dbl), 132.55699999999999 (dbl), 133.01 (dbl), 133.28299999999999 (dbl),
##   133.67699999999999 (dbl), 134.06299999999999 (dbl), 134.464 (dbl),
##   134.85499999999999 (dbl), 135.24299999999999 (dbl), 135.61799999999999
##   (dbl), 136.024 (dbl), 136.43 (dbl), 136.797 (dbl), 137.221 (dbl),
##   137.61500000000001 (dbl), 138.01 (dbl), 138.40299999999999 (dbl),
##   138.797 (dbl), 139.19 (dbl), 139.584 (dbl), 139.97900000000001 (dbl),
##   140.37200000000001 (dbl), 140.76599999999999 (dbl), 141.16 (dbl),
##   141.51400000000001 (dbl), 141.90899999999999 (dbl), 142.30199999999999
##   (dbl), 142.696 (dbl), 143.089 (dbl), 143.483 (dbl), 143.875 (dbl),
##   144.27099999999999 (dbl), 144.66499999999999 (dbl), 145.05600000000001
##   (dbl), 145.453 (dbl), 145.84700000000001 (dbl), 146.203 (dbl),
##   146.79400000000001 (dbl), 147.142 (dbl), 147.65100000000001 (dbl),
##   148.15899999999999 (dbl), 148.667 (dbl), 149.101 (dbl), 149.613 (dbl),
##   150.285 (dbl), 150.791 (dbl), 151.328 (dbl), 151.827 (dbl),
##   152.29499999999999 (dbl), 152.78299999999999 (dbl), 153.31200000000001
##   (dbl), 153.91900000000001 (dbl), 154.45699999999999 (dbl), 154.995
##   (dbl), 155.70400000000001 (dbl)
```

**Let's try puting in few new varables that will serve as potential features for classification:**

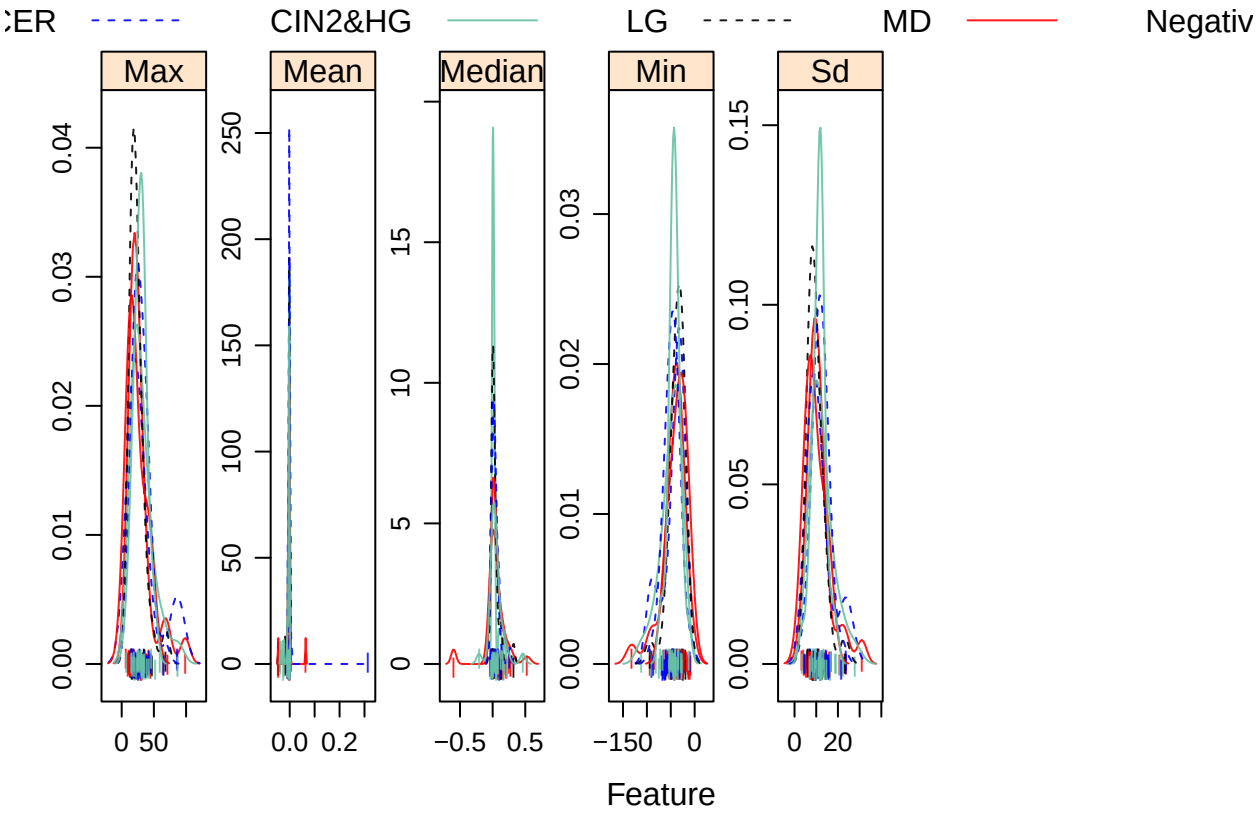
```
# as.matrix has to be used to transform df to matrix for matrixStats functions

useDataFeatures<-useData1%>%mutate(Mean=rowMeans(useData1[2:length(useData1)]),
                                   Median=rowMedians(as.matrix(useData1[2:length(useData1)])),
                                   Sd=rowSds(as.matrix(useData1[2:length(useData1)])),
                                   Max=rowMaxs(as.matrix(useData1[2:length(useData1)])),
                                   Min=rowMins(as.matrix(useData1[2:length(useData1)])))

useDataFeatures<-tbl_df(useDataFeatures)

# DF with just the newly calculated features

dataFeat<-select(useDataFeatures, Case, Mean, Median, Sd, Max, Min)
```

** Checking whether theese features make any sense:**

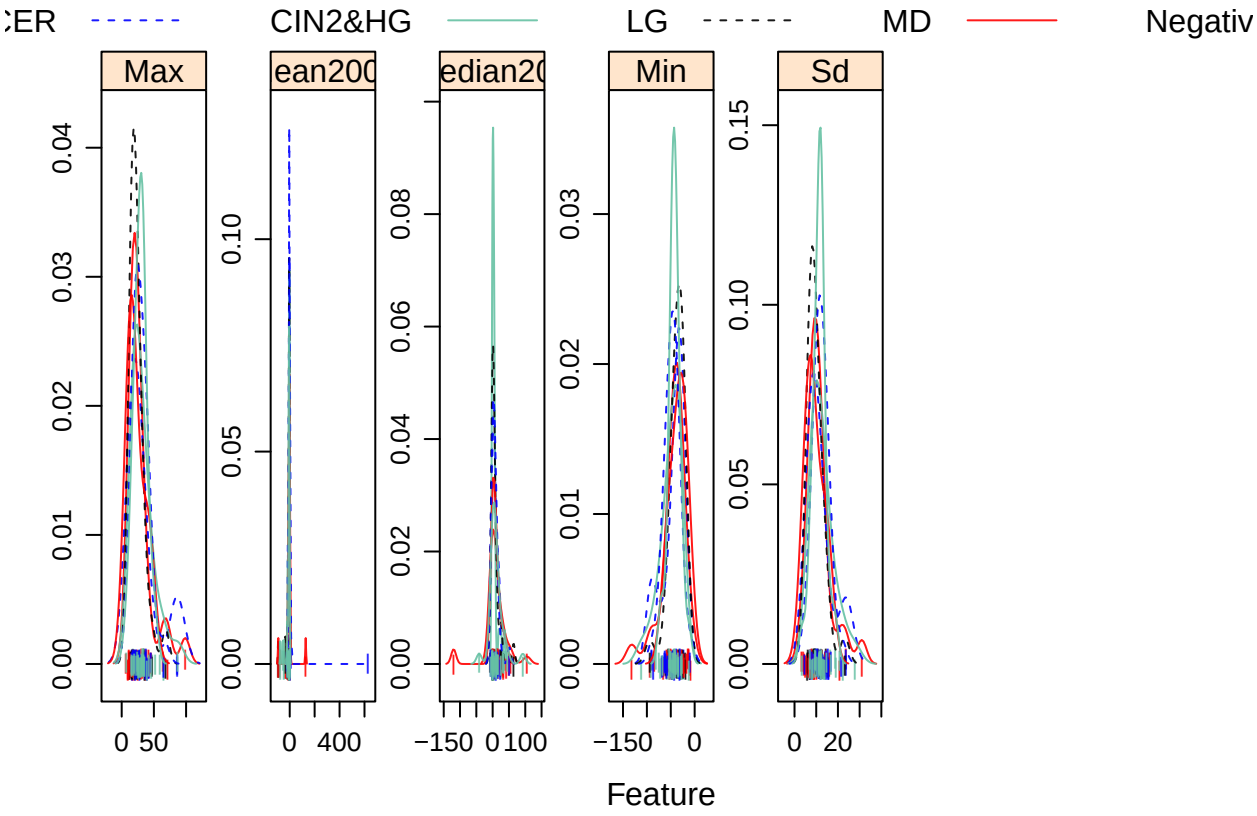
```
## Scatterplot Matrix
transparentTheme(trans = .4)

featurePlot(x = dataFeat[, 2:6],
            y = dataFeat$Case,
            plot = "pairs",
            ## Add a key at the top
            auto.key = list(columns = 7))
```




```
## Overlayed Density Plots
transparentTheme(trans = .9)
featurePlot(x = dataFeat[, 2:6],
            y = dataFeat$Case,
            plot = "density",
            ## Pass in options to xyplot() to
            ## make it prettier
            scales = list(x = list(relation="free"),
                          y = list(relation="free")),
            adjust = 1.5,
            pch = "|",
            layout = c(7, 1),
            auto.key = list(columns = 7))
```




```
## A bit of scaling for the "Mean" and "Median" features

dataFeat2<-mutate(dataFeat,Mean2000=Mean*2000, Median200=Median*200)

dataFeat2<-dataFeat2[-c(2:3)]

## Let's try overlayed density plots again

transparentTheme(trans = .9)
featurePlot(x = dataFeat2[,2:6],
            y = dataFeat2$Case,
            plot = "density",
            ## Pass in options to xyplot() to
            ## make it prettier
            scales = list(x = list(relation="free"),
                          y = list(relation="free")),
            adjust = 1.5,
            pch = "|",
            layout = c(7, 1),
            auto.key = list(columns = 7))
```




**Let's check how `min` and `max` values behave according to groups:**

```
dataFeat %>% group_by(Case) %>% summarise(groupAvgMax=mean(Max), groupAvgMin=mean(Min))
```

```
## Source: local data frame [7 x 3]
##
##       Case groupAvgMax groupAvgMin
##     (fctr)       (dbl)       (dbl)
## 1       BL    28.28284   -44.85641
## 2   CANCER    32.43411   -44.38103
## 3  CIN2&HG    31.88631   -50.02398
## 4       LG    23.91768   -37.36281
## 5       MD    22.14470   -34.25711
## 6 Negative    29.27853   -47.60861
## 7       SD    28.82962   -42.66793
```

## ML applications

**First we will try clustering the data set into 7 clusters to check whether any meaningful clustering can be performed, based on the chosen features. Though it is obvious that these features are not adequate at all...**

```
# k-means perforemed on dataFeat2 to check whether any meaningful clustering
# can be performed based on these features
set.seed(333)
```

```
clusterData<-dataFeat2[-1]
clusters<-dataFeat2$Case

kMeansClusters<-kmeans(clusterData,7) # clustring into 7 categories

table(clusters,kMeansClusters$cluster) # check does it make any sense
```

```
##
## clusters    1  2  3  4  5  6  7
##   BL        7  0  0  7  3  7  3
##   CANCER    7  0  0  1  2  3  0
##   CIN2&HG  11  0  0 10  3  2  0
##   LG       15  0  1  5  1  3  1
##   MD       11  0  1  6  0  1  1
##   Negative  9  1  0 14  1  1  1
##   SD        9  0  1 11  0  4  2
```

```
# Nope :(
```

**Things to be done: clustering with data normalization and afterwards with the complete data set, also try k-medoids and k-medians...**

### Classification with Random Forests and Stochastic Gradient Boosting

We'll try RF on both useData1 and dataFeat2 to see is any classification, based on the given features, feasible at all...

```
# Let's form the training and test sets, based on 75% and 25% of the total data (useData1), respectfuly
set.seed(333)

inTrain <- createDataPartition(y=useData1$Case,p=0.75, list=FALSE)

training <- useData1[inTrain,]
testing <- useData1[-inTrain,]

# Running RF

modFit <- train(Case~ ., data=training, method="rf", prox=TRUE)

# Check the model

modFit # Uzas
```

```
## Random Forest
##
## 128 samples
## 120 predictors
##   7 classes: 'BL', 'CANCER', 'CIN2&HG', 'LG', 'MD', 'Negative', 'SD'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
```

```
## Summary of sample sizes: 128, 128, 128, 128, 128, 128, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa       Accuracy SD  Kappa SD
##     2   0.2151340  0.08330785  0.05288310   0.05940351
##    61   0.2290432  0.09729526  0.05263333   0.06152901
##   120   0.2322679  0.10318555  0.04962339   0.05668596
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 120.
```

```
testPred <- predict(modFit, testing)

confusionMatrix(testPred, testing$Case) # Za plakanje
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction BL CANCER CIN2&HG LG MD Negative SD
##   BL        2      0       2  1  0        0  1
##   CANCER    0      2       0  0  0        0  0
##   CIN2&HG   1      0       1  0  0        1  1
##   LG        1      0       0  2  1        1  1
##   MD        0      1       2  0  2        0  0
##   Negative  1      0       0  1  0        4  0
##   SD        1      0       1  2  2        0  3
##
## Overall Statistics
##
##                Accuracy : 0.4211
##                  95% CI : (0.2631, 0.5918)
##     No Information Rate : 0.1579
##     P-Value [Acc > NIR] : 9.904e-05
##
##                   Kappa : 0.3187
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: BL Class: CANCER Class: CIN2&HG Class: LG
## Sensitivity            0.33333       0.66667        0.16667   0.33333
## Specificity            0.87500       1.00000        0.90625   0.87500
## Pos Pred Value         0.33333       1.00000        0.25000   0.33333
## Neg Pred Value         0.87500       0.97222        0.85294   0.87500
## Prevalence             0.15789       0.07895        0.15789   0.15789
## Detection Rate         0.05263       0.05263        0.02632   0.05263
## Detection Prevalence   0.15789       0.05263        0.10526   0.15789
## Balanced Accuracy      0.60417       0.83333        0.53646   0.60417
##                      Class: MD Class: Negative Class: SD
## Sensitivity            0.40000          0.6667   0.50000
## Specificity            0.90909          0.9375   0.81250
## Pos Pred Value         0.40000          0.6667   0.33333
## Neg Pred Value         0.90909          0.9375   0.89655
## Prevalence             0.13158          0.1579   0.15789
```

```
## Detection Rate         0.05263          0.1053   0.07895
## Detection Prevalence   0.13158          0.1579   0.23684
## Balanced Accuracy      0.65455          0.8021   0.65625
```

```
# And now with dataFeat2

# Running RF

modFit <- train(Case~., data=training, method="rf", prox=TRUE)

# Check the model

modFit # Uzas
```

```
## Random Forest
##
## 128 samples
## 120 predictors
##   7 classes: 'BL', 'CANCER', 'CIN2&HG', 'LG', 'MD', 'Negative', 'SD'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 128, 128, 128, 128, 128, 128, ...
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa       Accuracy SD  Kappa SD
##     2   0.2276898  0.09852165  0.07273416   0.08141272
##    61   0.2576418  0.13313920  0.06533171   0.07408408
##   120   0.2495140  0.12307112  0.06817150   0.07668988
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 61.
```

```
# Prediction on test data

testPred <- predict(modFit, testing)

# Results check

confusionMatrix(testPred, testing$Case) # Ocaj
```

```
## Confusion Matrix and Statistics
##
##            Reference
## Prediction BL CANCER CIN2&HG LG MD Negative SD
##   BL        2      0       2  1  0        0  2
##   CANCER    0      2       1  0  0        0  0
##   CIN2&HG   0      0       1  0  0        2  1
##   LG        1      0       0  1  2        1  0
##   MD        0      1       1  0  2        0  0
##   Negative  2      0       0  2  0        3  0
##   SD        1      0       1  2  1        0  3
##
```

```
## Overall Statistics
##
##                Accuracy : 0.3684
##                  95% CI : (0.2181, 0.5401)
##     No Information Rate : 0.1579
##     P-Value [Acc > NIR] : 0.001316
##
##                   Kappa : 0.2579
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: BL Class: CANCER Class: CIN2&HG Class: LG
## Sensitivity            0.33333       0.66667        0.16667   0.16667
## Specificity            0.84375       0.97143        0.90625   0.87500
## Pos Pred Value         0.28571       0.66667        0.25000   0.20000
## Neg Pred Value         0.87097       0.97143        0.85294   0.84848
## Prevalence             0.15789       0.07895        0.15789   0.15789
## Detection Rate         0.05263       0.05263        0.02632   0.02632
## Detection Prevalence   0.18421       0.07895        0.10526   0.13158
## Balanced Accuracy      0.58854       0.81905        0.53646   0.52083
##                      Class: MD Class: Negative Class: SD
## Sensitivity            0.40000         0.50000   0.50000
## Specificity            0.93939         0.87500   0.84375
## Pos Pred Value         0.50000         0.42857   0.37500
## Neg Pred Value         0.91176         0.90323   0.90000
## Prevalence             0.13158         0.15789   0.15789
## Detection Rate         0.05263         0.07895   0.07895
## Detection Prevalence   0.10526         0.18421   0.21053
## Balanced Accuracy      0.66970         0.68750   0.67188
```

```
# Let's try with boosting algh.

modFit <- train(Case~., method="gbm", data=training, verbose=FALSE)

# Check the model

print(modFit) #
```

```
## Stochastic Gradient Boosting
##
## 128 samples
## 120 predictors
##   7 classes: 'BL', 'CANCER', 'CIN2&HG', 'LG', 'MD', 'Negative', 'SD'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 128, 128, 128, 128, 128, 128, ...
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa       Accuracy SD
##   1                   50      0.2413343  0.11116766  0.05809439
##   1                  100      0.2307369  0.09987460  0.04899414
##   1                  150      0.2285456  0.09614802  0.05103002
```

```
##   2                   50      0.2374680  0.10680620  0.05202366
##   2                  100      0.2476053  0.12061600  0.06386658
##   2                  150      0.2361023  0.10604919  0.05385393
##   3                   50      0.2372298  0.10570380  0.05653890
##   3                  100      0.2480788  0.11886008  0.07073946
##   3                  150      0.2382309  0.10675256  0.06114377
##   Kappa SD
##   0.06435072
##   0.05332484
##   0.05598754
##   0.05699530
##   0.06818010
##   0.06050256
##   0.06320510
##   0.07823380
##   0.06791720
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using  the largest value.
## The final values used for the model were n.trees = 100,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```r
# Prediction on test data

testPred <- predict(modFit, testing)

# Results check

confusionMatrix(testPred, testing$Case) # Ocaj
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction BL CANCER CIN2&HG LG MD Negative SD
##   BL        0      0       0  1  1        0  1
##   CANCER    0      2       1  0  0        0  0
##   CIN2&HG   0      0       1  1  0        1  1
##   LG        2      1       2  2  2        0  0
##   MD        1      0       2  0  2        0  1
##   Negative  2      0       0  1  0        4  0
##   SD        1      0       0  1  0        1  3
##
## Overall Statistics
##
##                Accuracy : 0.3684
##                  95% CI : (0.2181, 0.5401)
##     No Information Rate : 0.1579
##     P-Value [Acc > NIR] : 0.001316
##
##                   Kappa : 0.2591
##  Mcnemar's Test P-Value : NA
##
```

```
## Statistics by Class:
##
##                      Class: BL Class: CANCER Class: CIN2&HG Class: LG
## Sensitivity            0.00000       0.66667        0.16667   0.33333
## Specificity            0.90625       0.97143        0.90625   0.78125
## Pos Pred Value         0.00000       0.66667        0.25000   0.22222
## Neg Pred Value         0.82857       0.97143        0.85294   0.86207
## Prevalence             0.15789       0.07895        0.15789   0.15789
## Detection Rate         0.00000       0.05263        0.02632   0.05263
## Detection Prevalence   0.07895       0.07895        0.10526   0.23684
## Balanced Accuracy      0.45312       0.81905        0.53646   0.55729
##                      Class: MD Class: Negative Class: SD
## Sensitivity            0.40000          0.6667   0.50000
## Specificity            0.87879          0.9062   0.90625
## Pos Pred Value         0.33333          0.5714   0.50000
## Neg Pred Value         0.90625          0.9355   0.90625
## Prevalence             0.13158          0.1579   0.15789
## Detection Rate         0.05263          0.1053   0.07895
## Detection Prevalence   0.15789          0.1842   0.15789
## Balanced Accuracy      0.63939          0.7865   0.70312
```

**As can be seen, it is almost impossible to obtain any meaningful classification model based on the existing data.**

**Two conditions have to be met for any further development:**

1. *larger data set - conditio sine qua non*
2. *defining meaningful features (covariates) that will be used for analysis and modeling - this has to be done through consultations with prof. Koruga and Sanja.*

**Also, we can try with feature extraction directly from raw data, i.e. images.**