Logbook parkinson dysphonia

L T Stein

2023-11-13

Packages

```
# loading knitting package
library("pander")
panderOptions("table.continues", "")
# loading plotting packages
library("ggplot2")
library("ggcorrplot")
# loading plot grid package
library("gridExtra")
# load package needed for pairs plot
library("GGally")
```

Introduction

The data is extracted from Kaggle and its article is published on ncbi by Max A. Little, Patrick E. McSharry, Eric J. Hunter, Lorraine O. Ramig (2008). 'Suitability of dysphonia measurements for telemonitoring of Parkinson's disease', *IEEE Transactions on Biomedical Engineering (to appear)*.

Background information

The article has found out a certain combination of measurements to predict the difference between a person with Parkinson's disease (PD) and without PD. Patients exhibit symptoms of loss in function of certain voice features which can be brought to light using different methods or tests.

These results can differentiate the states of PD or disease's presence at all. The numerical attributes of this data set represent the measurements created with different tools or tests accompanying with a variety of methods for identifying voice disorders.

The research paper's goal is to find out if healthy people can be discerned from people with Parkinson's disease using different sorts of voice recordings. Based on the data that the article gives as appendix, the purpose of this project is to find out what attribute or attributes are most important to give a diagnosis of possible sickness.

Therefore, given that goal the project's research question states: "Which type of voice measurements can be used to discern a Parkinson's patient from a healthy person?"

Observe data

First a look of what the data looks like from a .csv file. The data is divided by header and is comma separated.

```r
patient.data <- read.table("parkinsons_data.csv", header = T, sep = ",")
pander::pander(head(patient.data), caption = "First look of datastructure")
```

First look of datastructure (continued below)

name	MDVP.Fo.Hz.	MDVP.Fhi.Hz.	MDVP.Flo.Hz.	MDVP.Jitter...
phon_R01_S01_1	120	157.3	75	0.00784
phon_R01_S01_2	122.4	148.7	113.8	0.00968
phon_R01_S01_3	116.7	131.1	111.6	0.0105
phon_R01_S01_4	116.7	137.9	111.4	0.00997
phon_R01_S01_5	116	141.8	110.7	0.01284
phon_R01_S01_6	120.6	131.2	113.8	0.00968

MDVP.Jitter.Abs.	MDVP.RAP	MDVP.PPQ	Jitter.DDP	MDVP.Shimmer
7e-05	0.0037	0.00554	0.01109	0.04374
8e-05	0.00465	0.00696	0.01394	0.06134
9e-05	0.00544	0.00781	0.01633	0.05233
9e-05	0.00502	0.00698	0.01505	0.05492
0.00011	0.00655	0.00908	0.01966	0.06425
8e-05	0.00463	0.0075	0.01388	0.04701

MDVP.Shimmer.dB.	Shimmer.APQ3	Shimmer.APQ5	MDVP.APQ	Shimmer.DDA
0.426	0.02182	0.0313	0.02971	0.06545
0.626	0.03134	0.04518	0.04368	0.09403
0.482	0.02757	0.03858	0.0359	0.0827
0.517	0.02924	0.04005	0.03772	0.08771
0.584	0.0349	0.04825	0.04465	0.1047
0.456	0.02328	0.03526	0.03243	0.06985

NHR	HNR	status	RPDE	DFA	spread1	spread2	D2	PPE
0.02211	21.03	1	0.4148	0.8153	-4.813	0.2665	2.301	0.2847

NHR	HNR	status	RPDE	DFA	spread1	spread2	D2	PPE
0.01929	19.09	1	0.4584	0.8195	-4.075	0.3356	2.487	0.3687
0.01309	20.65	1	0.4299	0.8253	-4.443	0.3112	2.342	0.3326
0.01353	20.64	1	0.435	0.8192	-4.118	0.3341	2.406	0.369
0.01767	19.65	1	0.4174	0.8235	-3.748	0.2345	2.332	0.4103
0.01222	21.38	1	0.4156	0.8251	-4.243	0.2991	2.188	0.3578

Verify if everything is read correctly.

```
cat(ncol(patient.data), "columns and", nrow(patient.data), "rows")

## 24 columns and 195 rows
```

There are 2 nominal and 22 numeric columns.

Showing the amount of recordings of healthy people and sick people. The original labels of one and zero are replaced respectfully with sick and healthy. The new dataframe is written to a csv file for model training later on.

```
# Adding labels instead of "1" and "0"
patient.data$status[patient.data$status == "1"] <- "sick"
patient.data$status[patient.data$status == "0"] <- "healthy"

# preparing factor of levels
levels.of.status <- factor(patient.data$status)

amount.sick.recordings <- sum(levels.of.status == "sick")
amount.healthy.recordings <- sum(levels.of.status == "healthy")
cat(amount.sick.recordings, amount.healthy.recordings, amount.sick.recordings +
amount.healthy.recordings)

## 147 48 195
```

There are 147 recordings of Parkinson patient and 48 recordings of healthy people. In total there are: 195 recordings.

In order to make it more clear what the data means, the columns are put into a table.

```
columns <- colnames(patient.data)
definitions <- read.csv("definitions.csv", header = FALSE, sep = ",")
attributes <- data.frame(Attributes = c(columns),
                         Unit = c("-", rep("Hz", 3), "%", "ms", rep("-", 4), "dB", rep("-", 13)),
                        Datatype = c(sapply(patient.data, class)[1]),
                        Definitions = c(definitions[1]))
colnames(attributes) <- sub("V1", "Description", colnames(attributes))
pander::pander(attributes, style = "rmarkdown", caption = "Discriptions of attributes")
```

Discriptions of attributes (continued below)

Attributes	Unit	Datatype
name	-	character
MDVP.Fo.Hz.	Hz	character
MDVP.Fhi.Hz.	Hz	character
MDVP.Flo.Hz.	Hz	character
MDVP.Jitter...	%	character
MDVP.Jitter.Abs.	ms	character
MDVP.RAP	-	character
MDVP.PPQ	-	character
Jitter.DDP	-	character
MDVP.Shimmer	-	character
MDVP.Shimmer.dB.	dB	character
Shimmer.APQ3	-	character
Shimmer.APQ5	-	character
MDVP.APQ	-	character
Shimmer.DDA	-	character
NHR	-	character
HNR	-	character
status	-	character
RPDE	-	character
DFA	-	character
spread1	-	character
spread2	-	character
D2	-	character
PPE	-	character

Description
Name of recorded participant in experiment
The average vocal frequency in Hertz
The maximum vocal frequency in Hertz
The minimum vocal frequency in Hertz
Amount of jitter in percentage
Absolute jitter values in ms
Relative amplitude perturbation
Five-point period perturbation quotient
Absolute average difference of dissimilarities in jitter cycles
Local shimmer parameter
Local shimmer parameter in dB
Three-point amplitude perturbation quotient
Five-point amplitude perturbation quotient
MDVP 11-point amplitude perturbation quotient
Average absolute differences between the amplitudes of consecutive periods
Noise-to-harmonics ratio
Harmonics-to-noise ratio
State of health
Recurrence period density entropy measure
Signal fractal scaling exponent of detrended fluctuation analysis
Two nonlinear measures of fundamental
Frequency variation
Correlation dimension
Pitch period entropy

Retrospective All the data is read correctly with the corresponding amount of rows and columns as the article suggests. Noise-to-harmonics ratio and harmonics-to-noise ratio suggest they're the opposites of each other, it is interesting to observe this later on.

Explorative Data Analysis

It’s important to check if there are any NA or missing values.

```
cat("There are", sum(is.na(patient.data)), "missing values")

## There are 0 missing values
```

Retrospective No NA values are found.

By applying a summary to the dataset’s dataframe, its statistical meanings are returned.

```
pander::pander(head(summary(patient.data)), style = "rmarkdown", caption = "Summary of dataset")
```

Summary of dataset (continued below)

name	MDVP.Fo.Hz.	MDVP.Fhi.Hz.	MDVP.Flo.Hz.
Length:195	Min. : 88.33	Min. :102.1	Min. : 65.48
Class :character	1st Qu.:117.57	1st Qu.:134.9	1st Qu.: 84.29
Mode :character	Median :148.79	Median :175.8	Median :104.31
NA	Mean :154.23	Mean :197.1	Mean :116.32
NA	3rd Qu.:182.77	3rd Qu.:224.2	3rd Qu.:140.02
NA	Max. :260.11	Max. :592.0	Max. :239.17

MDVP.Jitter...	MDVP.Jitter.Abs.	MDVP.RAP	MDVP.PPQ
Min. :0.001680	Min. :7.000e-06	Min. :0.000680	Min. :0.000920
1st Qu.:0.003460	1st Qu.:2.000e-05	1st Qu.:0.001660	1st Qu.:0.001860
Median :0.004940	Median :3.000e-05	Median :0.002500	Median :0.002690
Mean :0.006220	Mean :4.396e-05	Mean :0.003306	Mean :0.003446
3rd Qu.:0.007365	3rd Qu.:6.000e-05	3rd Qu.:0.003835	3rd Qu.:0.003955
Max. :0.033160	Max. :2.600e-04	Max. :0.021440	Max. :0.019580

Jitter.DDP	MDVP.Shimmer	MDVP.Shimmer.dB.	Shimmer.APQ3
Min. :0.002040	Min. :0.00954	Min. :0.0850	Min. :0.004550
1st Qu.:0.004985	1st Qu.:0.01650	1st Qu.:0.1485	1st Qu.:0.008245
Median :0.007490	Median :0.02297	Median :0.2210	Median :0.012790
Mean :0.009920	Mean :0.02971	Mean :0.2823	Mean :0.015664
3rd Qu.:0.011505	3rd Qu.:0.03789	3rd Qu.:0.3500	3rd Qu.:0.020265
Max. :0.064330	Max. :0.11908	Max. :1.3020	Max. :0.056470

Shimmer.APQ5	MDVP.APQ	Shimmer.DDA	NHR
Min. :0.00570	Min. :0.00719	Min. :0.01364	Min. :0.000650
1st Qu.:0.00958	1st Qu.:0.01308	1st Qu.:0.02474	1st Qu.:0.005925
Median :0.01347	Median :0.01826	Median :0.03836	Median :0.011660
Mean :0.01788	Mean :0.02408	Mean :0.04699	Mean :0.024847
3rd Qu.:0.02238	3rd Qu.:0.02940	3rd Qu.:0.06080	3rd Qu.:0.025640
Max. :0.07940	Max. :0.13778	Max. :0.16942	Max. :0.314820

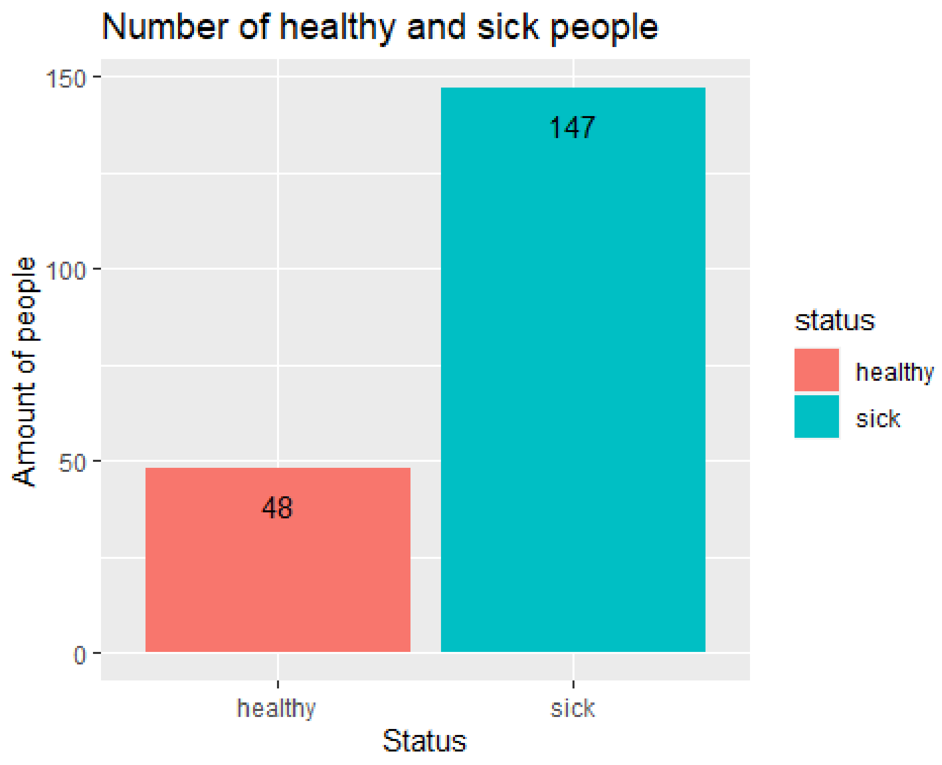
HNR	status	RPDE	DFA
Min. : 8.441	Length:195	Min. :0.2566	Min. :0.5743
1st Qu.:19.198	Class :character	1st Qu.:0.4213	1st Qu.:0.6748
Median :22.085	Mode :character	Median :0.4960	Median :0.7223
Mean :21.886	NA	Mean :0.4985	Mean :0.7181
3rd Qu.:25.076	NA	3rd Qu.:0.5876	3rd Qu.:0.7619
Max. :33.047	NA	Max. :0.6852	Max. :0.8253

spread1	spread2	D2	PPE
Min. :-7.965	Min. :0.006274	Min. :1.423	Min. :0.04454
1st Qu.:-6.450	1st Qu.:0.174350	1st Qu.:2.099	1st Qu.:0.13745
Median :-5.721	Median :0.218885	Median :2.362	Median :0.19405
Mean :-5.684	Mean :0.226510	Mean :2.382	Mean :0.20655
3rd Qu.:-5.046	3rd Qu.:0.279234	3rd Qu.:2.636	3rd Qu.:0.25298
Max. :-2.434	Max. :0.450493	Max. :3.671	Max. :0.52737

From observing the data above it becomes clear from the means that the values between attributes can differ highly. Also, the data of NHR (Noise-to-harmonics ration) and HRN (Harmonics-to-noise ratio) seem to pose that they are opposite attributes of each other.

Bar plot for showing amount of sick and healthy people

```
ggplot(patient.data, aes(x = status, fill = status)) + geom_bar() + xlab("Status") +
  ylab("Amount of people") + labs(title = "Number of healthy and sick people") + geom_text(stat='count',
aes(label=after_stat(count)), vjust=2)
```



Retrospective

In this dataset there are nearly thrice as many people recorded whom are sick opposing to healthy people. This means that values of the healthy status that are measured by coincidence, deal more impact to its dataset.

Normally it's preferred to have a dataset with lots of values for each group. In further research the sum of false positives therefore might be more than desired. In this context false positives are values that are measured by coincidence.

If alpha is 0.05 then the amount of false positives per status equals to the following results.

```
alpha <- 0.05
false.healthy.positives <- alpha * amount.healthy.recordings
false.sick.positives <- alpha * amount.sick.recordings
cat(false.healthy.positives, false.sick.positives)

## 2.4 7.35
```

This estimates that of the 48 healthy instances two values are measured by coincidence and of the 147 sick instances seven values are measured by chance.

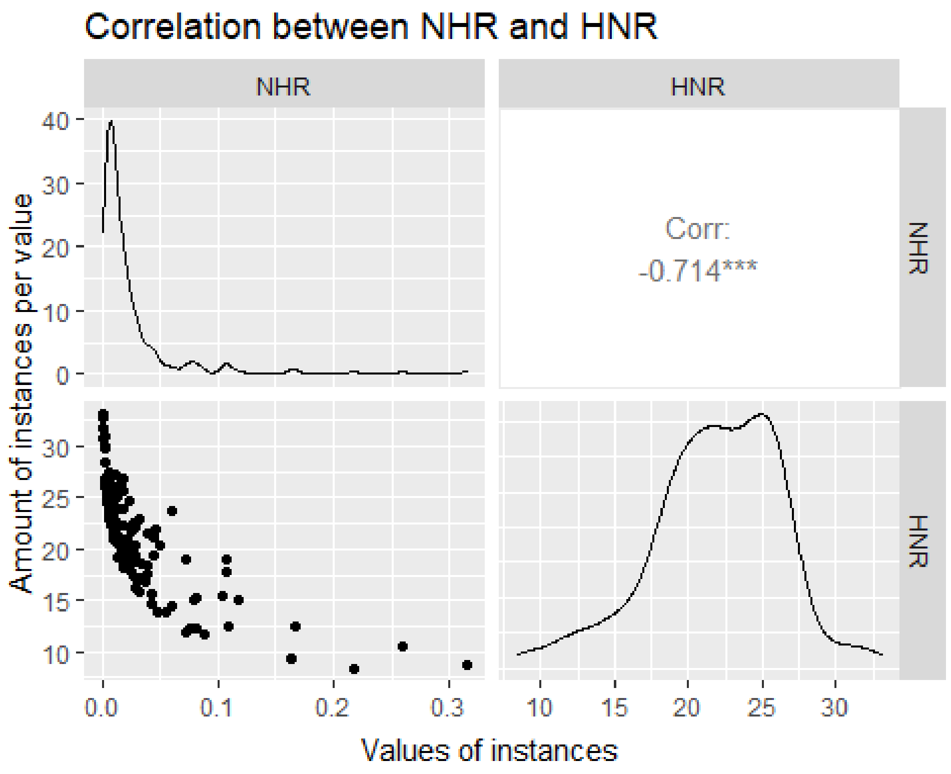
Retrospective There are approximately ten false positives that probably deviate from the instances in its attribute.

Based on the names different groups are formed for visualisation

```
MDVP.large <- 2:4
MDVP.small <- 5:9
MDVP.shimmer <- 10:15
NHR.HNR <- 16:17
RPDE.DEA <- 19:20
spread <- 21:22
D2.PPE <- 23:24
```

To find out if NHR and HNR correlate oppositely to one another a pairwise plot is made.

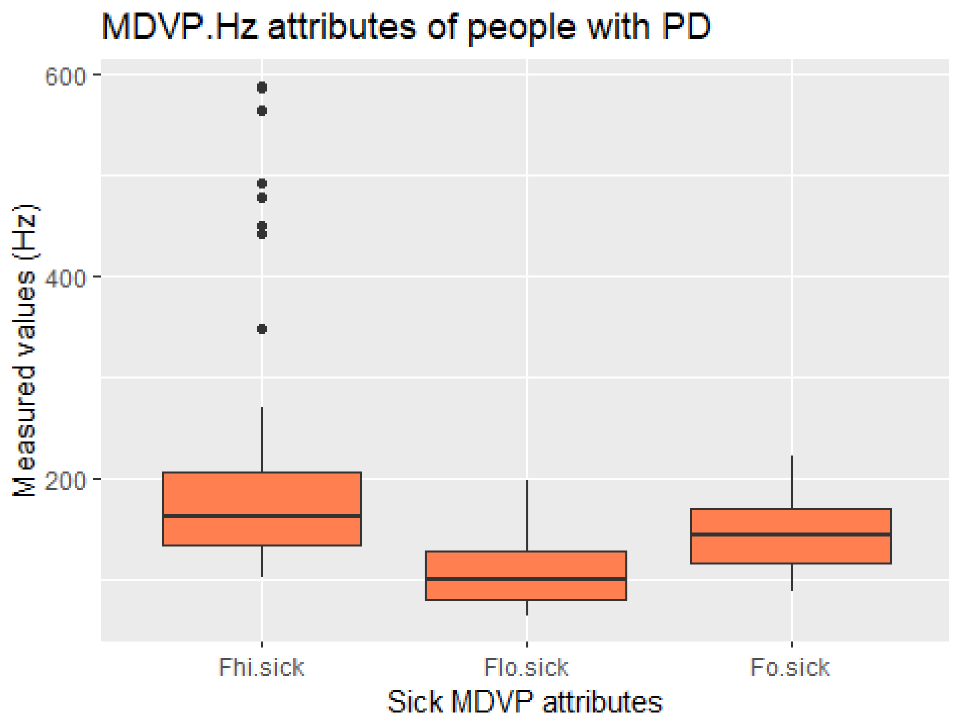
```
ggpairs(patient.data[NHR.HNR]) + labs(title = "Correlation between NHR and HNR") + xlab("Values of instances") +
                          ylab("Amount of instances per value") +
               theme(axis.title.x = element_text(vjust=-0.5), axis.title.y = element_text(vjust=1))
```



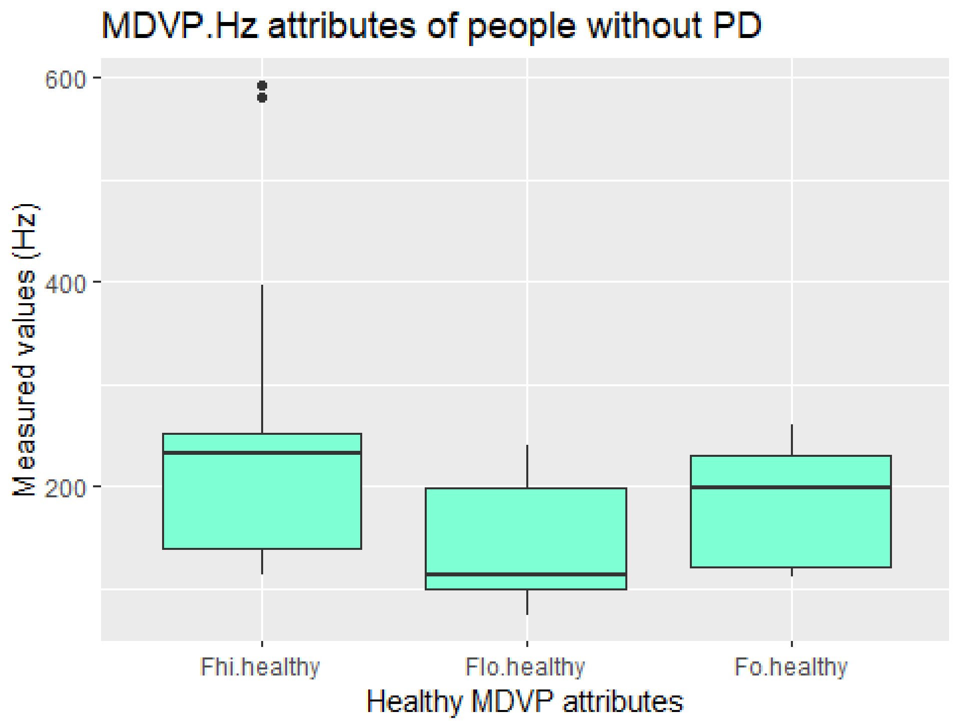
The two attributes correlate to each other oppositely, though their values are not the exact opposites. There are too many points that deviate from the downwards curve that the data moves in. An explanation might be that the formulas to calculate the two attributes differ from each other, thus not result in exact opposite values.

Boxplotting the data to show the differences for sick people and healthy people in MDVP measured attributes containing large numbers. These values are expressed in Hertz measuring the vocal frequencies (Fo=average, Fhi=maximum and Flo=minimum). It would be very curious to see if vocal frequencies are enough to distinguish sick from healthy people.

```
Fo <- patient.data$MDVP.Fo.Hz.[levels.of.status == "sick"]
Fhi <- patient.data$MDVP.Fhi.Hz.[levels.of.status == "sick"]
Flo <- patient.data$MDVP.Flo.Hz.[levels.of.status == "sick"]

Fo.healthy <- patient.data$MDVP.Fo.Hz.[levels.of.status == "healthy"]
Fhi.healthy <- patient.data$MDVP.Fhi.Hz.[levels.of.status == "healthy"]
Flo.healthy <- patient.data$MDVP.Flo.Hz.[levels.of.status == "healthy"]

difference.in.nums.sick <- data.frame(
                    labs = c(rep("Fo.sick", length(Fo)),
                         rep("Fhi.sick", length(Fhi)),
                         rep("Flo.sick", length(Flo))),
                    values = c(Fo, Fhi, Flo))

difference.in.nums.healthy <- data.frame(
                    labs = c(rep("Fo.healthy", length(Fo.healthy)),
                         rep("Fhi.healthy", length(Fhi.healthy)),
                         rep("Flo.healthy", length(Flo.healthy))),
                    values = c(Fo.healthy, Fhi.healthy, Flo.healthy))



ggplot(difference.in.nums.sick,aes(x=labs, y=values)) +
  geom_boxplot(fill = "coral") + ggtitle("MDVP.Hz attributes of people with PD") +
  labs(caption = "Boxplot of three attributes of MDVP.Hz for people with PD") +
  xlab("Sick MDVP attributes") + ylab("Measured values (Hz)")

ggplot(difference.in.nums.healthy,aes(x=labs, y=values)) + geom_boxplot(fill = "aquamarine") +
  ggtitle("MDVP.Hz attributes of people without PD") + labs(caption = "Boxplot of three attributes of
MDVP.Hz for people without PD") +
  xlab("Healthy MDVP attributes") + ylab("Measured values (Hz)")
```



Boxplot of three attributes of MDVP.Hz for people with PD



Boxplot of three attributes of MDVP.Hz for people without PD

Between the MDVP.Hz boxplots it becomes clear that the MDVP.HZ attribute means are similar to each other for sick people.

From only boxplotting the three MDVP.HZ attributes it becomes clear that sick people result in frequencies around 150 and healthy people are measured between 200-250. The frequencies (Hz) of the healthy labels per box are all higher than for the boxplot with sick people.

With ANOVA tests these differences can be determined per attribute.

```
aov.Fo.Hz <- summary(aov(MDVP.Fo.Hz. ~ status, data = patient.data))
aov.Fhi.Hz <- summary(aov(MDVP.Fhi.Hz. ~ status, data = patient.data))
aov.Flo.Hz <- summary(aov(MDVP.Flo.Hz. ~ status, data = patient.data))
cat("Fo.Hz = ", aov.Fo.Hz[[1]]$`Pr(>F)`[1], "\nFhi.Hz = ",
  aov.Fhi.Hz[[1]]$`Pr(>F)`[1], "\nFlo.Hz = ", aov.Flo.Hz[[1]]$`Pr(>F)`[1])
```

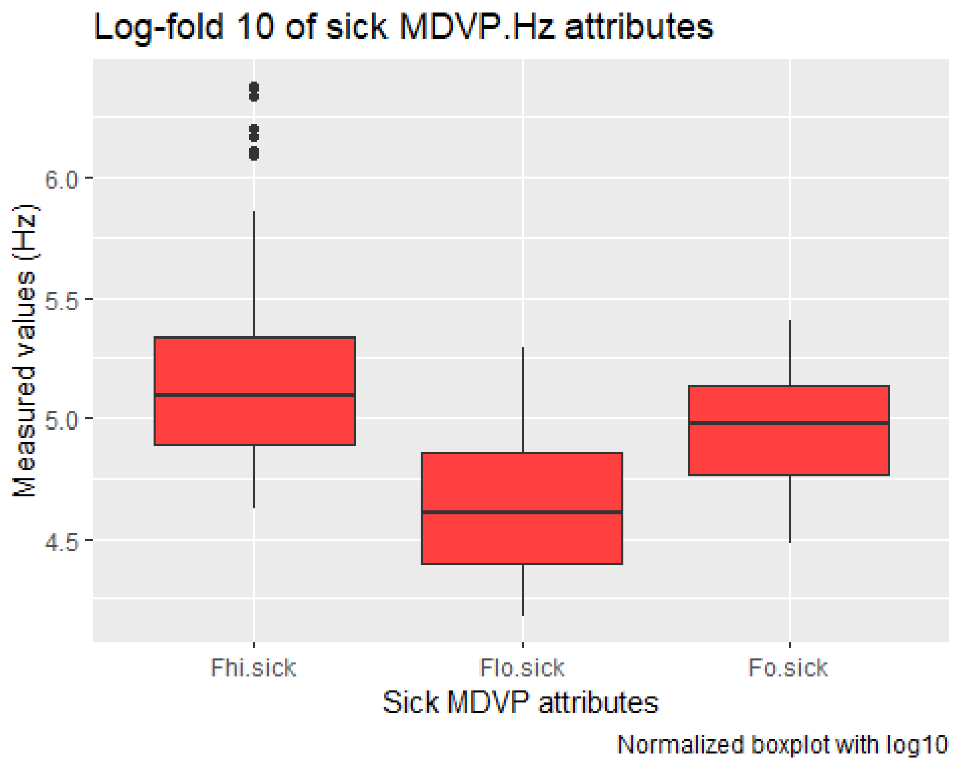
Attributes	P-values
Fo.Hz	3.121919e-08
Fhi.Hz	0.02027567
Flo.Hz	4.197004e

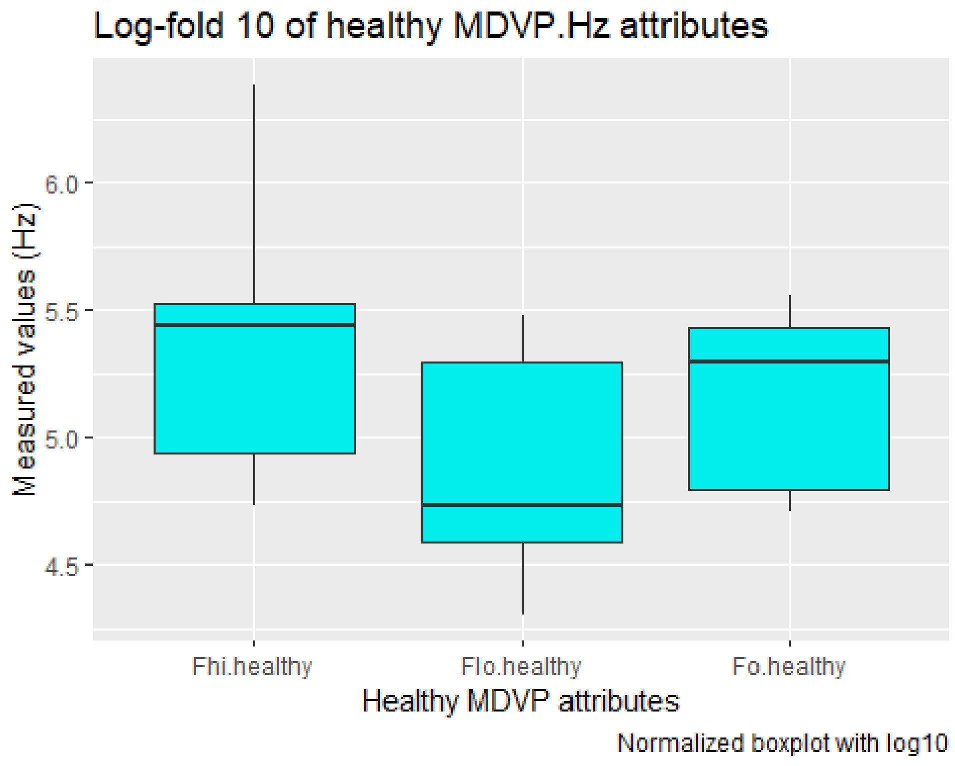
With all p-values smaller than alpha = 0.05 there's a significant difference between the statuses of each attribute.

Retrospective There are significant differences between the levels “healthy“ and “sick“ in the attributes MDVP.Fo.Hz, MDVP.Fhi.Hz, MDVP.Flo.Hz. These attributes might be interesting to use in differentiating people of PD with healthy people.

In order to find out if a log10-transformation skewers the amount of outliers that are present above, the boxplots from above are log-transformed.

```
difference.in.nums.sick.log10 <- data.frame(
                        labs = c(rep("Fo.sick", length(Fo)),
                               rep("Fhi.sick", length(Fhi)),
                               rep("Flo.sick", length(Flo))),
                        values = c(log(Fo), log(Fhi), log(Flo)))

difference.in.nums.healthy.log10 <- data.frame(
                        labs = c(rep("Fo.healthy", length(Fo.healthy)),
                               rep("Fhi.healthy", length(Fhi.healthy)),
                               rep("Flo.healthy", length(Flo.healthy))),
                        values = c(log(Fo.healthy), log(Fhi.healthy), log(Flo.healthy)))



ggplot(difference.in.nums.sick.log10,aes(x=labs, y=values)) +
  geom_boxplot(fill = "brown1") + ggtitle("Log-fold 10 of sick MDVP.Hz attributes") +
  labs(caption = "Normalized boxplot with log10")  +
  xlab("Sick MDVP attributes") + ylab("Measured values (Hz)")

ggplot(difference.in.nums.healthy.log10,aes(x=labs, y=values)) +
  geom_boxplot(fill = "cyan2") + ggtitle("Log-fold 10 of healthy MDVP.Hz attributes") + labs(caption =
"Normalized boxplot with log10") +
  xlab("Healthy MDVP attributes") + ylab("Measured values (Hz)")
```





After log-transforming the majority of outliers disappear. The healthy instance frequencies remain higher than the sick instances.

An ANOVA test is done to see if after normalizing the data, its differences between the levels remain as significant.

```
aov.Fo.Hz.log <- summary(aov(log(MDVP.Fo.Hz.) ~ status, data = patient.data))
aov.Fhi.Hz.log <- summary(aov(log(MDVP.Fhi.Hz.) ~ status, data = patient.data))
aov.Flo.Hz.log <- summary(aov(log(MDVP.Flo.Hz.) ~ status, data = patient.data))
cat("Fo.Hz = ", aov.Fo.Hz.log[[1]]$`Pr(>F)`[1], "\nFhi.Hz = ",
   aov.Fhi.Hz.log[[1]]$`Pr(>F)`[1], "\nFlo.Hz = ", aov.Flo.Hz.log[[1]]$`Pr(>F)`[1])
```

Attributes	P-values
Fo.Hz	1.405169e-06
Fhi.Hz	0.004524624
Flo.Hz	7.635047e-07

After normalizing the data the differences between the levels remain significant, though attributes Fo.Hz and Flo.Hz are less significant. This can be explained by the fact that their outliers held on more diverse values, that are now reduced.

Retrospective

From observing this data it can be concluded that healthy people are measured higher frequencies than sick people. Also, log-transforming successfully skewers outliers.

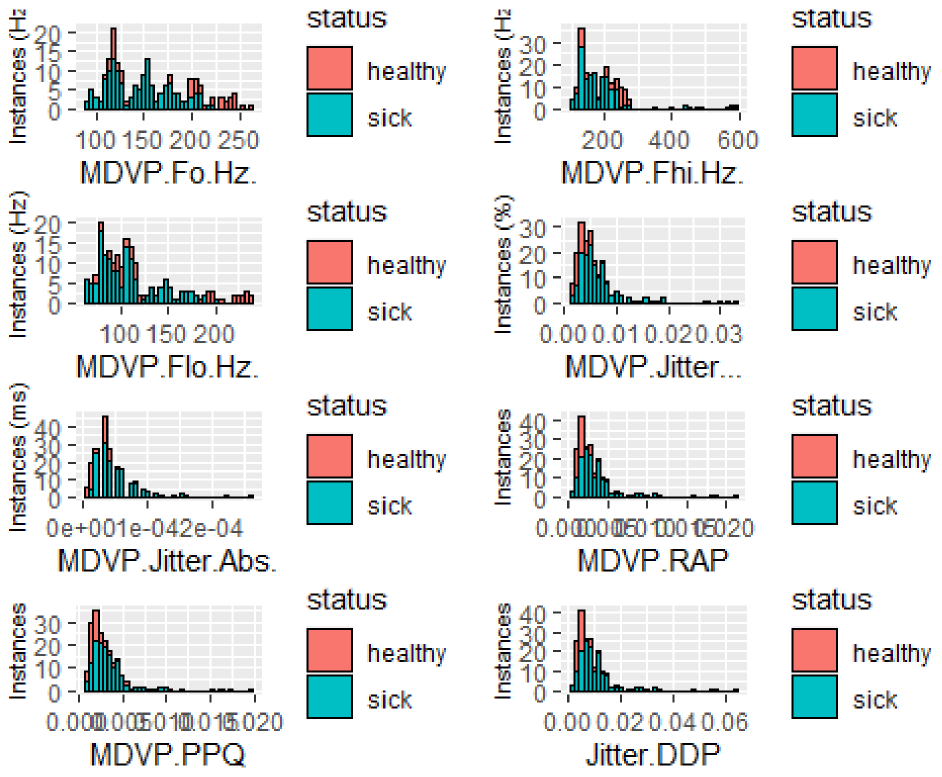
Identifying the data’s distribution

Histograms can used to visualize if attributes’ data are normally distributed. This will be done in a shared grid per (n) attributes. After observing if needed attributes with no normal distribution will be log-transformed, to see if that pushes the data more towards the center.

Histograms of first eight attributes in order to see if the attribute data is normally distributed.

```
F.1 <- ggplot(patient.data, aes(x=MDVP.Fo.Hz., fill = status)) + geom_histogram(color = "black", bins = 39)
+
  ylab("Instances (Hz)") + theme(axis.title.y = element_text(size = 8))
F.2 <- ggplot(patient.data, aes(x=MDVP.Fhi.Hz., fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances (Hz)") + theme(axis.title.y = element_text(size = 8))
F.3 <- ggplot(patient.data, aes(x=`MDVP.Flo.Hz.`, fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances (Hz)") + theme(axis.title.y = element_text(size = 8))
F.4 <- ggplot(patient.data, aes(x=`MDVP.Jitter...`, fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances (%)") + theme(axis.title.y = element_text(size = 8))
F.5 <- ggplot(patient.data, aes(x=MDVP.Jitter.Abs., fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances (ms)") + theme(axis.title.y = element_text(size = 8))
F.6 <- ggplot(patient.data, aes(x=MDVP.RAP, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.7 <- ggplot(patient.data, aes(x=MDVP.PPQ, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.8 <- ggplot(patient.data, aes(x=Jitter.DDP, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
grid.arrange(F.1, F.2, F.3, F.4, F.5, F.6, F.7, F.8, ncol = 2, nrow = 4)
```

Stacked histograms of first eight attributes

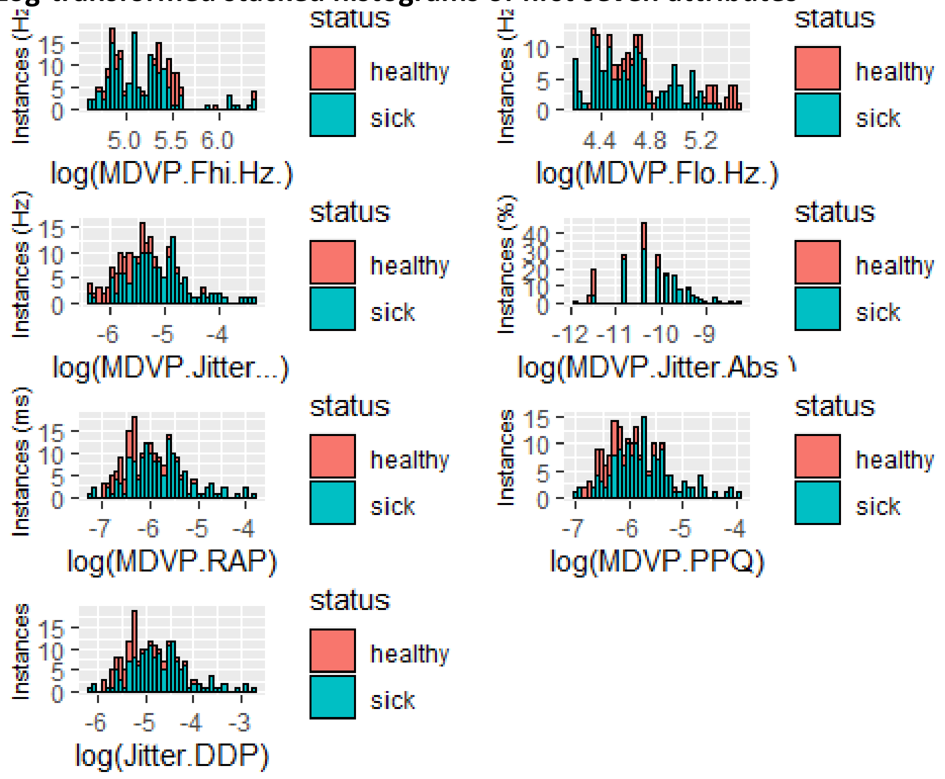


Shared histogram plot of first eight attributes

Apart from the first attribute MDVP.Fo.HZ, the rest of the histogram's data majorly orients towards the left side. Because of that they will be log-transformed with fold ten.

```
F.1.log <- ggplot(patient.data, aes(x=log(MDVP.Fhi.Hz.), fill = status)) + geom_histogram(color = "black",
bins = 39) +
  ylab("Instances (Hz)") + theme(axis.title.y = element_text(size = 8))
F.2.log <- ggplot(patient.data, aes(x=log(`MDVP.Flo.Hz.`), fill = status)) + geom_histogram(color = "black",
bins = 39) +
  ylab("Instances (Hz)") + theme(axis.title.y = element_text(size = 8))
F.3.log <- ggplot(patient.data, aes(x=log(`MDVP.Jitter...`), fill = status)) + geom_histogram(color = "black",
bins = 39) +
  ylab("Instances (Hz)") + theme(axis.title.y = element_text(size = 8))
F.4.log <- ggplot(patient.data, aes(x=log(MDVP.Jitter.Abs.), fill = status)) + geom_histogram(color =
"black", bins = 39) +
  ylab("Instances (%)") + theme(axis.title.y = element_text(size = 8))
F.5.log <- ggplot(patient.data, aes(x=log(MDVP.RAP), fill = status)) + geom_histogram(color = "black",
bins = 39) +
  ylab("Instances (ms) ") + theme(axis.title.y = element_text(size = 8))
F.6.log<- ggplot(patient.data, aes(x=log(MDVP.PPQ), fill = status)) + geom_histogram(color = "black", bins
= 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.7.log <- ggplot(patient.data, aes(x=log(Jitter.DDP), fill = status)) + geom_histogram(color = "black", bins
= 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
grid.arrange(F.1.log, F.2.log, F.3.log, F.4.log, F.5.log, F.6.log, F.7.log, ncol = 2, nrow = 4)
```

Log-transformed stacked histograms of first seven attributes



Shared histogram plot of first seven attributes that are log-transformed

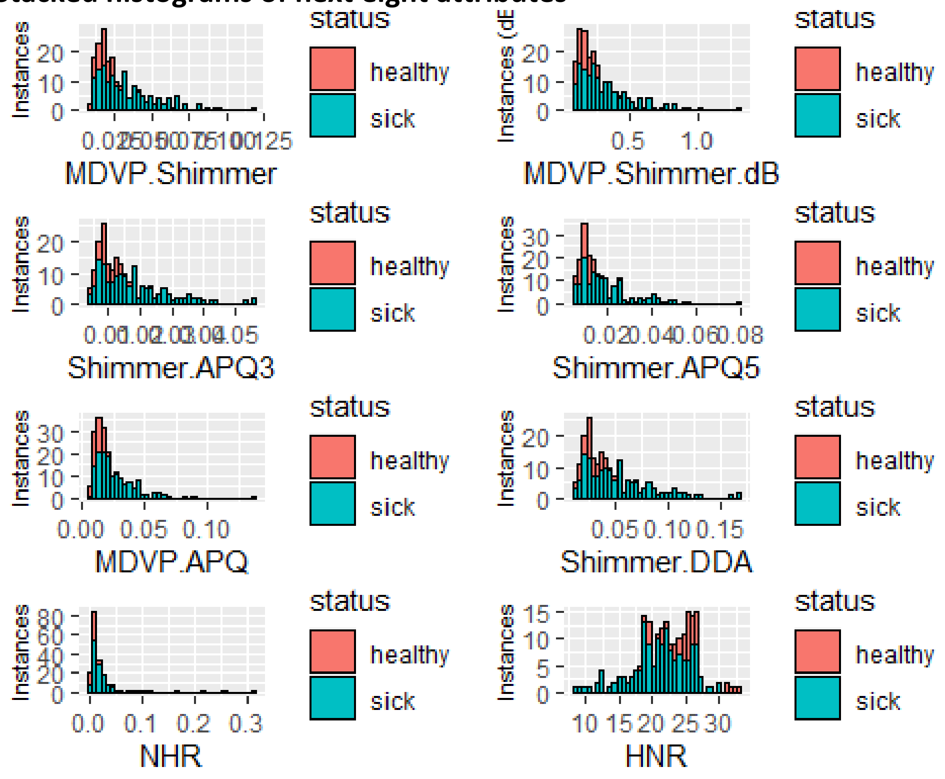
After log-transforming the figures above seem more normally distributed.

Histograms of next eight attributes in order to see if the attribute data is normally distributed.

```
F.9 <- ggplot(patient.data, aes(x=MDVP.Shimmer, fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.10 <- ggplot(patient.data, aes(x=MDVP.Shimmer.dB., fill = status)) + geom_histogram(color = "black",
bins = 39) +
  ylab("Instances (dB)") + theme(axis.title.y = element_text(size = 8))
F.11 <- ggplot(patient.data, aes(x=Shimmer.APQ3, fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
```

```
F.12 <- ggplot(patient.data, aes(x=Shimmer.APQ5, fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.13 <- ggplot(patient.data, aes(x=MDVP.APQ, fill = status)) + geom_histogram(color = "black", bins = 39)
+
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.14 <- ggplot(patient.data, aes(x=Shimmer.DDA, fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.15 <- ggplot(patient.data, aes(x=NHR, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.16 <- ggplot(patient.data, aes(x=HNR, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
grid.arrange(F.9, F.10, F.11, F.12, F.13, F.14, F.15, F.16, ncol = 2, nrow = 4)
```

Stacked histograms of next eight attributes



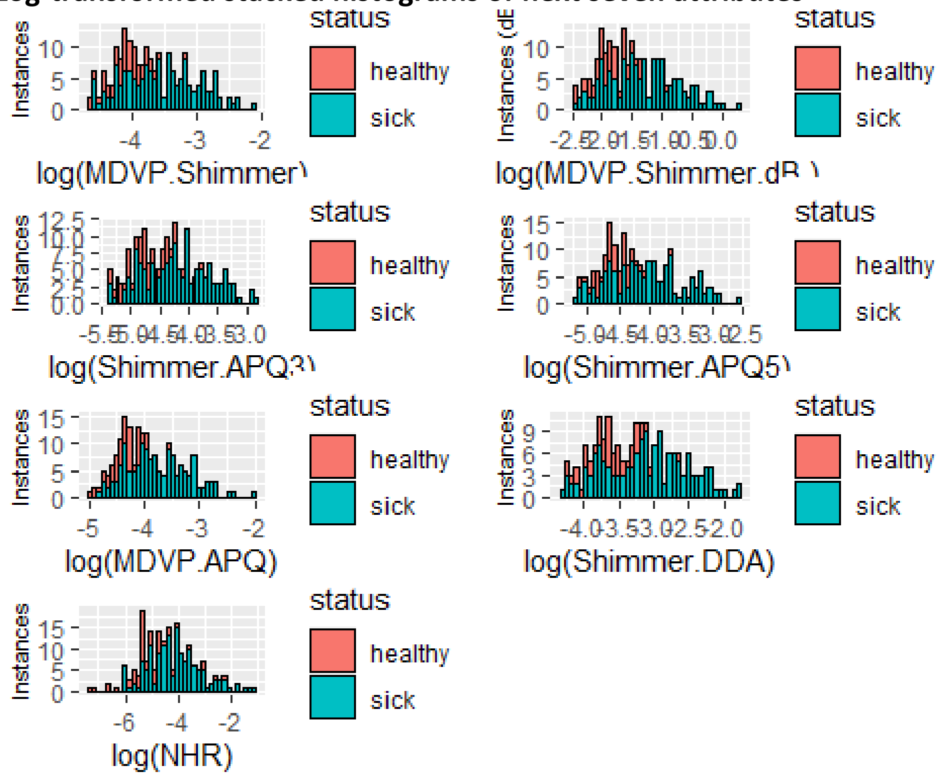
Shared histogram plot of next eight attributes

Seven of the eight attributes plotted above are oriented towards the left side. Because of that these attributes will be log-transformed with fold ten.

```
F.8.log <- ggplot(patient.data, aes(x=log(MDVP.Shimmer), fill = status)) + geom_histogram(color =
"black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
```

```
F.9.log <- ggplot(patient.data, aes(x=log(MDVP.Shimmer.dB.), fill = status)) + geom_histogram(color =
"black", bins = 39) +
  ylab("Instances (dB)") + theme(axis.title.y = element_text(size = 8))
F.10.log <- ggplot(patient.data, aes(x=log(Shimmer.APQ3), fill = status)) + geom_histogram(color =
"black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.11.log <- ggplot(patient.data, aes(x=log(Shimmer.APQ5), fill = status)) + geom_histogram(color =
"black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.12.log <- ggplot(patient.data, aes(x=log(MDVP.APQ), fill = status)) + geom_histogram(color = "black",
bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.13.log <- ggplot(patient.data, aes(x=log(Shimmer.DDA), fill = status)) + geom_histogram(color =
"black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.14.log <- ggplot(patient.data, aes(x=log(NHR), fill = status)) + geom_histogram(color = "black", bins =
39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
grid.arrange(F.8.log, F.9.log, F.10.log, F.11.log, F.12.log, F.13.log, F.14.log, ncol = 2, nrow = 4)
```

Log-transformed stacked histograms of next seven attributes



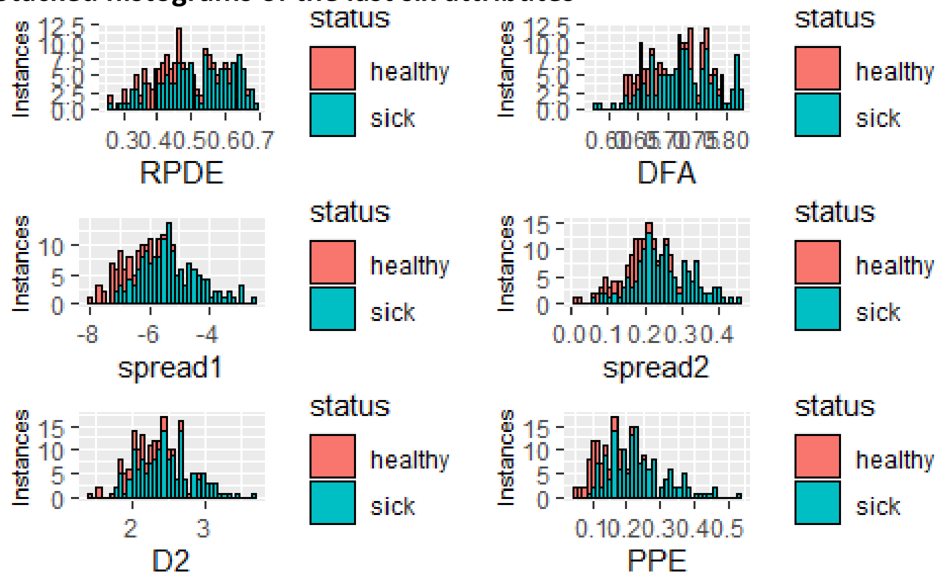
Shared histogram plot of remaining seven attributes that are log-transformed

After log-transforming the figures above the data seems more normally distributed.

Histogram of the last six attributes in order to see if the attribute data is normally distributed.

```r
F.17 <- ggplot(patient.data, aes(x=RPDE, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.18 <- ggplot(patient.data, aes(x=DFA, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.19 <- ggplot(patient.data, aes(x=spread1, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.20 <- ggplot(patient.data, aes(x=spread2, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.21 <- ggplot(patient.data, aes(x=D2, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
F.22 <- ggplot(patient.data, aes(x=PPE, fill = status)) + geom_histogram(color = "black", bins = 39) +
  ylab("Instances") + theme(axis.title.y = element_text(size = 8))
grid.arrange(F.17, F.18, F.19, F.20, F.21, F.22, ncol = 2, nrow = 4)
```

Stacked histograms of the last six attributes

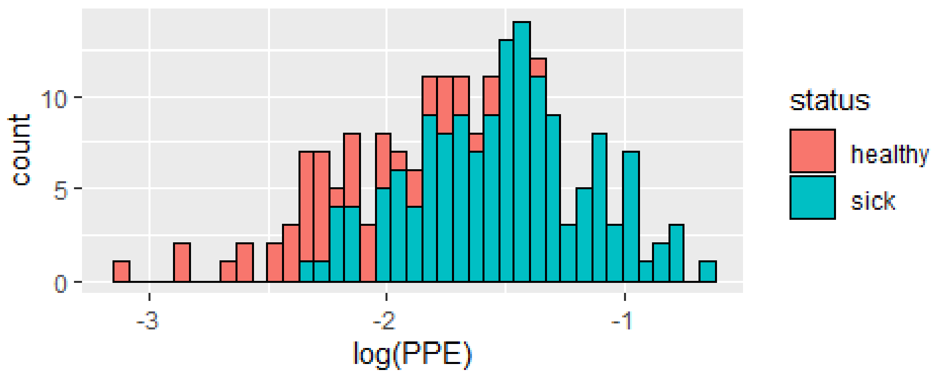


Shared histogram plot of last six attributes

The last six attributes seem to have well distributed data. Attribute PPE has left oriented data, because of that it will be log-transformed in the hope that the data shifts more towards the center.

```
F.22.log <- ggplot(patient.data, aes(x=log(PPE), fill = status)) + geom_histogram(color = "black", bins = 39)
grid.arrange(F.22.log, ncol = 1, nrow = 2)
```

Log-transformed stacked histogram of attribute PPE



Histogram of log-tranformed attribute “PPE“

After log-transforming attribute “PPE“, the data has shifted more towards the center with a better normal distribution.

Retrospective

Fourteen out of the 22 numeric attributes are not normally distributed. After log-transforming it with fold ten, it instead seems to spread more evenly. Realizing the log10 really changes the distributions, the entire data will be log-transformed for processing later. When comparing the original dataset with this new dataset the fourteen attributes found will be observed more carefully.

All numeric attributes are log10-transformed for later model processing. Attribute spread1 remains unchanged, because it has negative values that are not suitable for log-tranformating.

```r
# preparing dataset with scaled log data and an unchanged spread1 column.
data.for.scaling <- subset(patient.data, select = c(-name, -status, -spread1))

logTransformed.patient.data.left <- data.frame(log(data.for.scaling[1:18]), spread1 =
patient.data$spread1)
logTransformed.patient.data.right <- data.frame(log(data.for.scaling[19:21]))
combined.log.data <- data.frame(logTransformed.patient.data.left, logTransformed.patient.data.right,
status = patient.data$status)
write.csv(combined.log.data, file = "log_patient_data.csv", row.names = FALSE)
```

Are there clear differences in the class labels of each log-transformed attribute? The following function does a Welch t-test for each attribute. This function will not run automatically since the output isn’t very pretty. Therefore, it was run once and saved in csv format.

```r
# Function that tests each attribute based on the class labels
Welch.t.test <- function(data) {
  for (i in 1:ncol(data)) {
    if (is.numeric(data[, i])) {
      sick <- data[, i][data$status == "sick"]
      healthy <- data[, i][data$status == "healthy"]
      cat(colnames(data)[i], ",", t.test(sick, healthy)$p.value, "\n")

    }
  }
}
Welch.t.test(combined.log.data)

pander::pander(read.csv("differences_of_classLabels.csv", sep = ",", header = T))
```

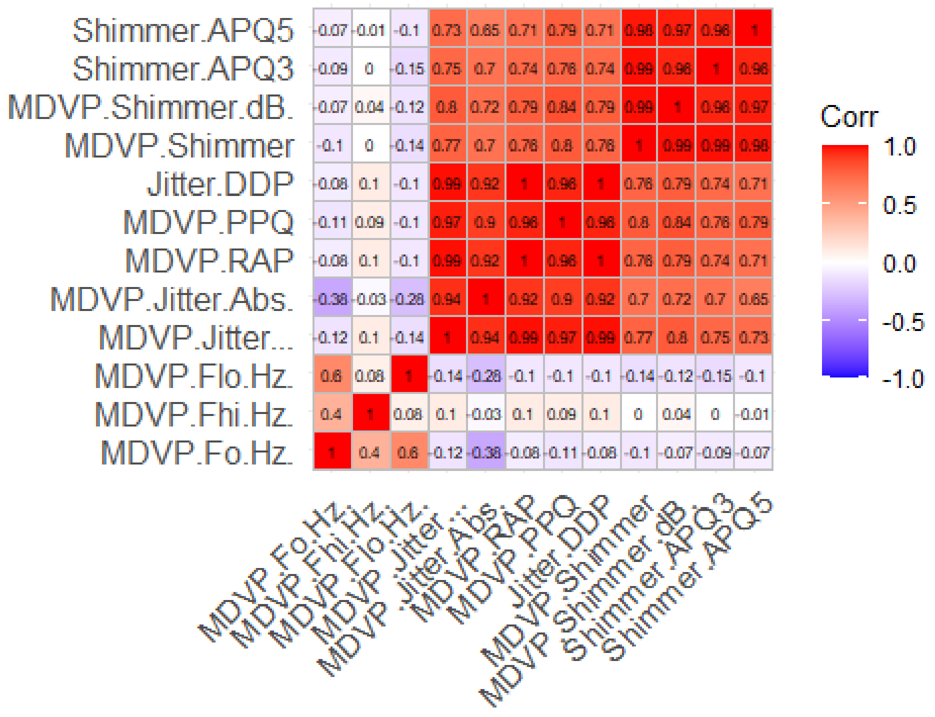
Attribute	p.value
MDVP.Fo.Hz.	8.1e-05
MDVP.Fhi.Hz.	0.007
MDVP.Flo.Hz.	5.7e-05
MDVP.Jitter...	1.5e-09
MDVP.Jitter.Abs.	2.7e-10
MDVP.RAP	6.6e-10
MDVP.PPQ	3.9e-11
Jitter.DDP	6.6e-10
MDVP.Shimmer	1.4e-14
MDVP.Shimmer.dB.	5.6e-14

Attribute	p.value
Shimmer.APQ3	2e-11
Shimmer.APQ5	3.3e-14
MDVP.APQ	3.1e-18
Shimmer.DDA	2.1e-11
NHR	1.7e-07
HNR	2.7e-09
RPDE	2.9e-05
DFA	0.0011
spread1	1.8e-21
spread2	3.3e-06
D2	1.2e-06
PPE	6.1e-17

From the Welch t-testing of each attribute it seems all p-values are below alpha = 0.5. This means that the data of the class labels differ significantly for each attribute. Attribute MDVP.APQ has the largest difference of 3.1e-18 and PPE the second largest of 6.1e-17.

A correlation heatmap using ggcorplot. It takes a matrix of the numeric data en a matrix of their p-values. Since the dataset is too large for a pretty heatmap, it's split into three.

```
# preparing dataset containing only of type numeric
data.numeric <- subset(patient.data, select = c(-name, -status))
# creating three matrices
corr.matrix1 <- cor(data.numeric[1:12])
corr.matrix2 <- cor(data.numeric[11:22])
corr.matrix3 <- cor(data.numeric[1:6], data.numeric[16:22])
# plotting heat map
ggcorrplot(corr.matrix1, lab = T, lab_size = 2, lab_col = "black") + labs(caption = "Correlation heatmap of first half of the dataset")
```

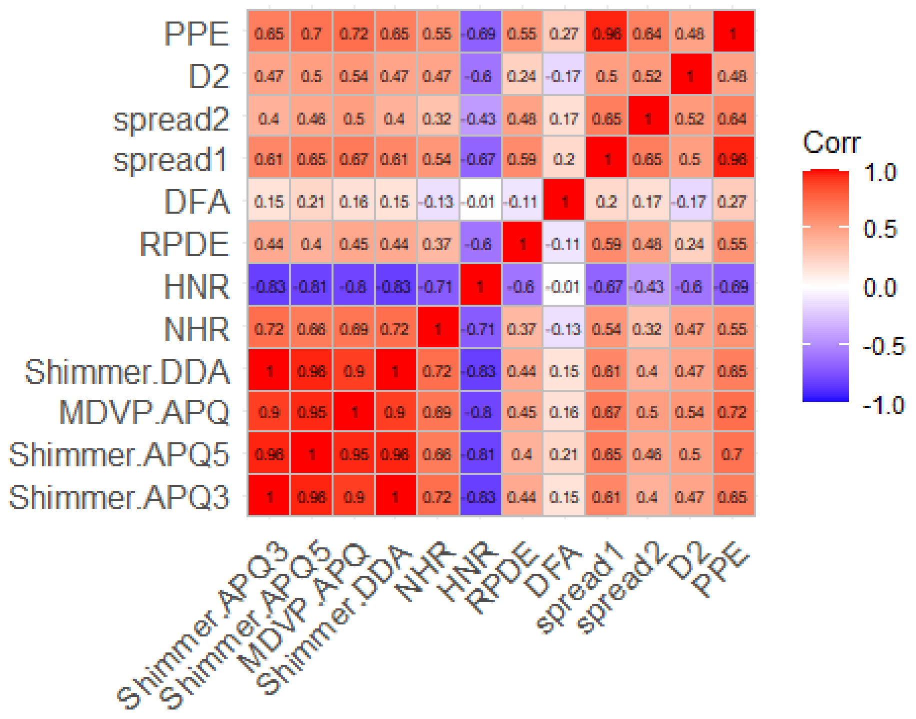


Correlation heatmap of first half of the dataset

The correlation matrix shows the similarity of all attributes. Assuming 1 means attributes producing very similar values and towards -1 more unique values.

Shimmer attributes seem to correlate a lot with each other, but also with a few MDVP and jitter attributes. Why are these attributes' instances so similar?

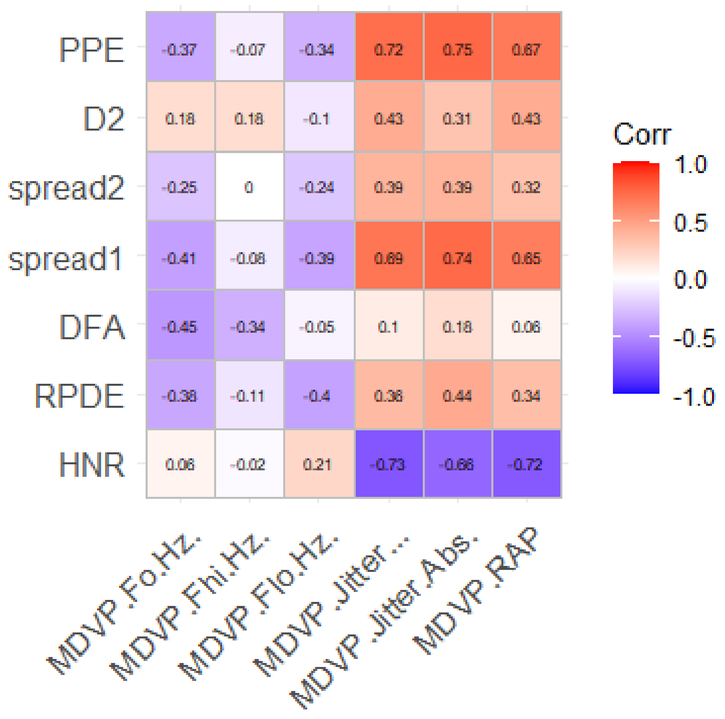
```
ggcorrplot(corr.matrix2, lab = T, lab_size = 2, lab_col = "black") + labs(caption = "Correlation heatmap of
other half of the dataset")
```



Correlation heatmap of other half of the dataset

NHR and HNR show very strong negative correlation like was established earlier after observing the data’s summary. DFA generally shows very weak correlation from all other attributes, indicating its data is very different. The shimmer attributes present moderate correlations with spread1, spread2, D2 and PPE. Apparently attributes PPE and spread1 have very similar data.

```
ggcorrplot(corr.matrix3, lab = T, lab_size = 2, lab_col = "black") + labs(caption = "Correlation heatmap of
first and last attributes")
```



Correlation heatmap of first and last attributes

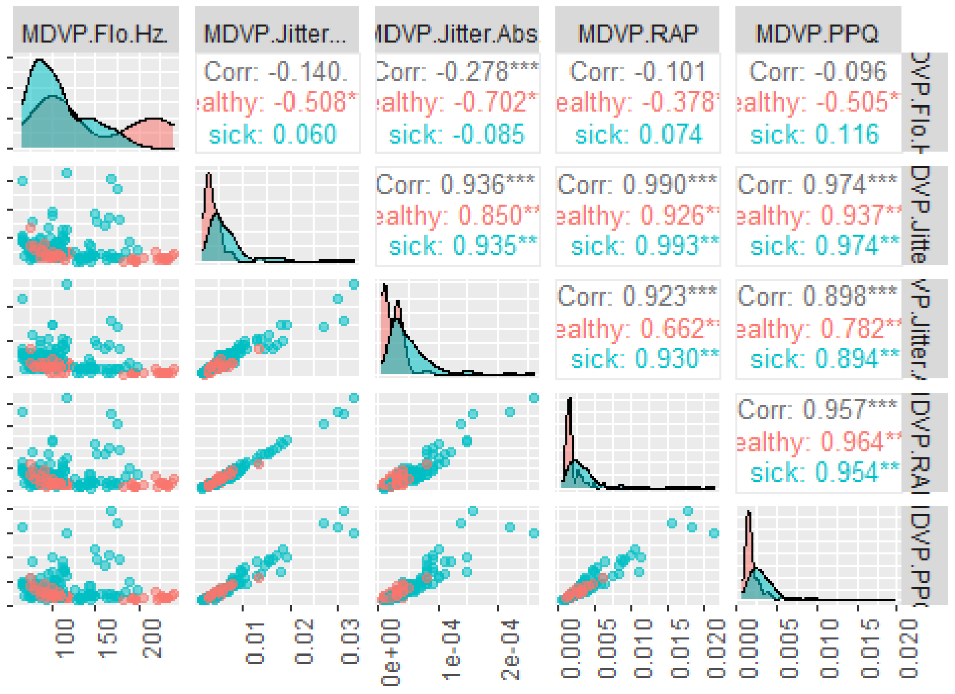
The first six and last six attributes correlate very diverse. HNR shows either very low similarity or strong negative similarity. From observing this matrix it becomes apparent that these attributes mostly are very different from one another, but nine squares correlate somewhat more.

Retrospective

Several attributes are very similar to each other. Some show their data to be face to face with strong negative values. From observing these differences and similarities in the dataset its attributes PPE (pitch period entropy), HNR (harmonics to noise ration), spread1 (nonlinear measurements) and DFA (exponent for invariance of mean and deviation against time) stand out the most.

Pair plot showing if attributes form pairs thus have correlating data. Only the groups that have correlations above 0.9 are plotted to see if their statuses are also similar. The first group contains the attributes [MDVP.Jitter... MDVP.Jitter.Abs. MDVP.RAP MDVP.PPQ Jitter.DDP].

```
ggpairs(data.numeric[MDVP.small-2], ggplot2:: aes(color=levels.of.status, alpha = 0.5), progress = FALSE,
upper = list(continuous = wrap(ggally_cor, size = 3.5))) + labs(caption = "Pair plot showing correlations
between six MDVP attributes\n and showing how similar the status data is between them") +
theme(axis.text.x = element_text(angle=90, hjust = 1),
      axis.text.y = element_blank())
```



Pair plot showing correlations between six MDVP attributes
and showing how similar the status data is between them

The columns MDVP.PPQ and MDVP.RAP correlate the best all-round, but MDVP.RAP and MDVP.Jitter perform the best correlation value of 0.990. More interesting MDVP.Jitter.Abs and MDVP.Flo.Hz show a strong negative correlation, between sick and healthy.

Though what does negative correlation really indicate? An article from Indeed states “one variable increases in value while the other decreases“. This statement is what attributes NHR and HNR showed earlier and clarifies that the attributes MDVP.Jitter.Abs and MDVP.Flo.Hz have values that stand face to face. Since the healthy and sick values are both negative with a large difference it really tells how divided the levels are with this attribute combination.

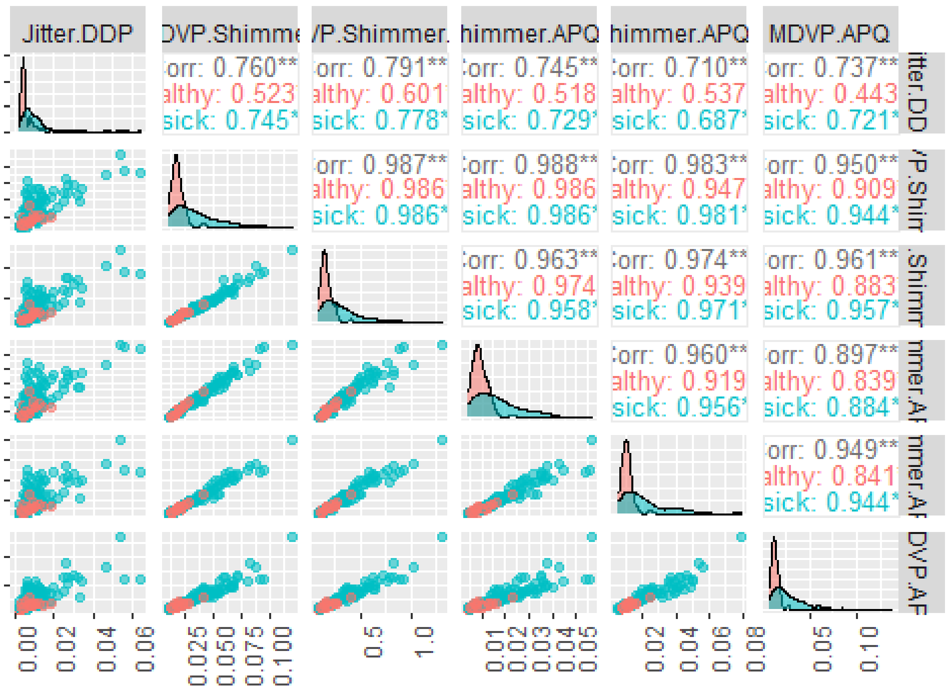
With a welch t-test attributes MDVP.Jitter.Abs and MDVP.Flo.Hz will be compared to see how different their data really is.

```
t.test(patient.data$MDVP.Jitter.Abs.,  patient.data$MDVP.Flo.Hz.)

##
##  Welch Two Sample t-test
##
## data:  patient.data$MDVP.Jitter.Abs. and patient.data$MDVP.Flo.Hz.
## t = -37.324, df = 194, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -122.4714 -110.1778
## sample estimates:
##    mean of x    mean of y
## 4.395897e-05 1.163246e+02
```

There is a very strong significant difference between the attributes of 2.2e-16 < 0.05.

The shimmer attributes correlated very strongly, therefore it's assumed the statuses also have very similar values. The attributes [MDVP.Shimmer MDVP.Shimmer.dB. Shimmer.APQ3 Shimmer.APQ5 MDVP.APQ Shimmer.DDA] are going to be compared.

```
ggpairs(data.numeric[MDVP.shimmer-2], ggplot2:: aes(color=levels.of.status, alpha = 0.5), progress =
FALSE, upper = list(continuous = wrap(ggally_cor, size = 3.5))) + labs(caption = "Pair plot showing
correlations between six MDVP.shimmer attributes\n and showing how similar the status data is
between them", legend = 1) + theme(axis.text.x = element_text(angle=90, hjust = 1), axis.text.y =
element_blank(), legend.position = "bottom")
```



Pair plot showing correlations between six MDVP.shimmer attributes
and showing how similar the status data is between them

Attributes Shimmer.APQ3 and MDVP.Shimmer show the best correlation in the plot above. Though many shimmer and MDVP attributes perform highly. The higher the correlation means the more positive the attributes react onto each other to increase their values. This can be seen in the graphs that move from the zero corner to the top right corner.

Also, what's quite remarkable is that the majority of the scatterplots show that healthy values group against the more spread lines of the sick values. This mainly occurs in graphs that correlate positively. A way of interpreting the grouping is that the values of healthy are different with a clear border from the sick values. Another idea is that the fact that most healthy values cluster, it means that all those values react according to the correlation coefficients. For example all values of healthy between Shimmer.APQ5 and MDVP.APQ impact the sick values to increase in value.

Does behaviour of how instances move in a chart represent the similarity in data between attributes? This will be tested with a welch t-test with attributes Shimmer.APQ3 and MDVP.Shimmer.

```
t.test(patient.data$Shimmer.APQ3,  patient.data$MDVP.Shimmer)

##
##  Welch Two Sample t-test
##
## data:  patient.data$Shimmer.APQ3 and patient.data$MDVP.Shimmer
## t = -9.1577, df = 297.76, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.01706319 -0.01102676
## sample estimates:
##  mean of x  mean of y
## 0.01566415 0.02970913
```

Likewise, to the attributes with a negative correlation, these attributes with a high positive correlation have very dissimilar data with a p-value of $2.2e-16 < 0.05$. This answers the question that correlation rather shows data's behaviour than similarity of values.

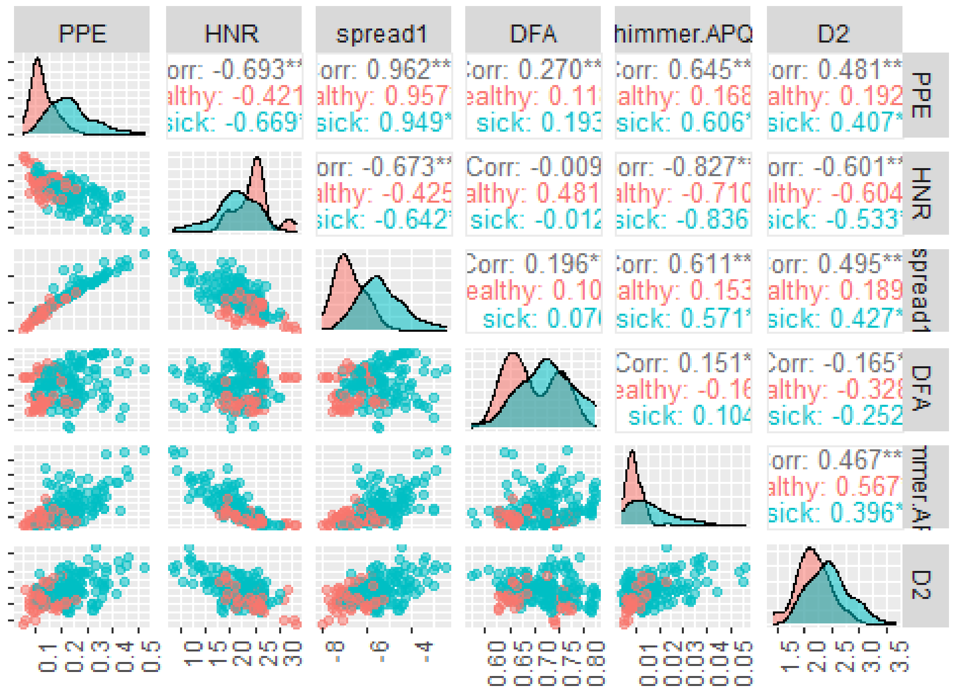
Retrospective

When a high correlation value is met between a pair it means, the data behaves in a similar way. An assumption is that high correlation shows there's less to none impact of attributes and their statuses onto each other, which means that combination of attributes may not be used to diagnose a PD patient.

Remarkably the combination [MDVP.Jitter.Abs and MDVP.Flo.Hz] differentiated sick values from healthy values. This combination will later be tested in model training, to affirm is that's really the case.

Instead of looking at how similar data performs on the statuses a final pair plot will be made of the four attributes that stood out in the correlation matrix as more unique data. The attributes PPE, HNR, spread1 and DFA are plotted along with two other attributes for a broader image of their differences. The attributes Shimmer.APQ3 and D2 are added, because they have high and moderate correlations.

```
striking.attributes <- data.numeric[,c("PPE", "HNR", "spread1", "DFA", "Shimmer.APQ3", "D2")]
ggpairs(striking.attributes, ggplot2:: aes(color=levels.of.status, alpha = 0.05), progress = FALSE, upper =
list(continuous = wrap(ggally_cor, size = 3.5))) + labs(caption = "Pair plot showing correlations between
six attributes with generally high (dis)similarity\n and showing how similar the status data is between
them", legend = 1) + theme(axis.text.x = element_text(angle=90, hjust = 1), axis.text.y = element_blank(),
legend.position = "bottom")
```



r plot showing correlations between six attributes with generally high (dis)similarity
and showing how similar the status data is between them

Generally the attributes statuses differ with 0-0.200. There are several combinations with negative corr values. Testing those later with a machine learning model might give perspective of how mild negative correlating attributes behave differently from strong negative correlations.

Attributes DFA and HNR, Shimmer.APQ3 and PPE show differences of 0.400 and strike as best differentiating combinations for this plot. PPE and spread1 show the cleanest positive correlation in the plot with a clear difference in statuses visually. It’s interesting to find out if observing the correlation coefficients is better or not than observing the scatterplots for choosing a pair that might be good for diagnosing.

Retrospective

The six attributes show low differences in the statuses or negative correlations. Four attributes have been found with larger differences, which might indicate that these combinations can be used for distinguishing.

Exploring Weka algorithms on dataset

Explorer

First let’s see in the Explorer mode how several algorithms perform on the original data.

ZeroR

ZeroR predicts class value: sick Correctly Classified Instances 147 75.3846 % Incorrectly Classified Instances 48 24.6154 %

ZeroR does a fair guess as expected for what rows are labeled sick.

OneR

PPE: < 0.1044335 -> healthy < 0.1165645 -> sick < 0.129713 -> healthy >= 0.129713 -> sick (175/195 instances correct)

Correctly Classified Instances 168 86.1538 % Incorrectly Classified Instances 27 13.8462 %

OneR does better at guessing, though its thresholds aren’t very logical. For attribute PPE it says two times healthy and sick constantly higher than the previous. There isn’t a real boundary.

The next Explored mode will be J48 pruned tree

Correctly Classified Instances 157 80.5128 % Incorrectly Classified Instances 38 19.4872 %

A better view is constructed of dividing the levels since more attributes are shown.

For example

MDVP.Flo.Hz. <= 100.209: healthy (2.0) | | | | | | | MDVP.Flo.Hz. > 100.209: sick (6.0)

MDVP.Flo.Hz. shows a clear threshold

Another example

spread2 <= 0.213353: healthy (12.0) | | | spread2 > 0.213353: sick (3.0)

spread2 also gives a clear threshold

Even though the J48 tree is less accurate it does give more predictions. Let’s move on to the Experimenter environment to compare more algorithms.

Experimenter

This testing phase is executed in the Experimenter environment of Weka. By pressing "New" in the top right corner the first dataset can be added. Next press"Add new" under "Datasets" towards the left and choose the file and its type you wish to use. For example iris dataset with .arff extension. After adding the desired files under Algorithms "Add new" and then "Choose"; several algorithms can be added for comparison. Under "Experiment Type" the kind of training can be chosen i.e. training dataset or cross validation.

Next go to "Run" and press start, for this run the error "Class attribute is not nominal!" appeared. This might be because it's expected that the class attribute is the last column of the dataframe. This is tested by moving the status column to the end.

```
# Excluding the rownames
edit.patientData <- data.frame(patient.data[,
c(2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,24,19,20,21,22,23,18)])
# Writing dataframe to csv file
write.csv(edit.patientData, file = "patient_data_labels.csv", row.names = FALSE)
```

No errors occur when running the experimenter with the new csv ordered file. Thus, when adding a csv file the nominal attribute that is splitted on must be the last column.

Default parameters

For the first test the algorithms IBk, J48, ZeroR, OneR, RandomForest and NaiveBayes are tested with the default parameters and Weka settings.

Dataset | (1) rules.Zero | (2) rules.One |(3) bayes |(4) trees (J48) |(5) trees (RandomForest) |(6) lazy. IBk log_patient_data | (100) 75.39 | 87.81 v | 72.75 | 84.63 v | 91.70 v | 93.03 v patient_data_labels | (100) 75.39 | 87.70 v | 70.14 | 84.79 v | 91.41 v | 95.91 v | (v/ /*) | (2/0/0) | (0/2/0) | (2/0/0) | (2/0/0) | (2/0/0)

NaiveBayes is the least accurate and IBk is the most, though the model IBk is overfitted with the default option of KNN = 1. RandomForest tries voting for better results, which is reflected with its accuracy if ~ 91 percent.

Retrospective IBk -1 and RandomForest are most accurate.

Edited parameters

Next let’s compare the default models with edited models.

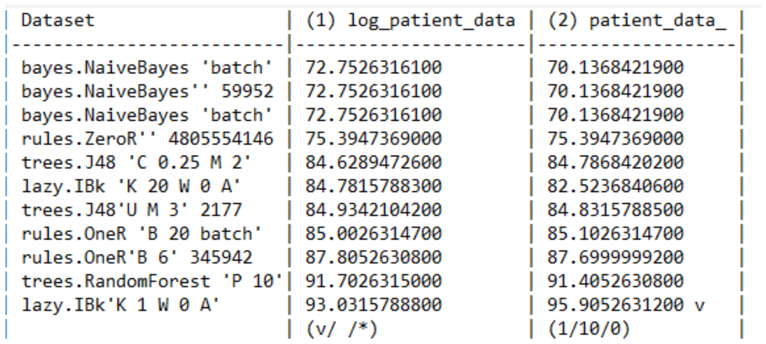
What would happen if for J48 the unpruning is set to true with a minNumObj to three and a confidenceFactor of 0.20? What happens if the batchSize is increased to 120, i.e. the amount in instances to process is increased.

The bucketsize of OneR is set to twenty and the batchsize to 120. Also, for NaiveBayes one model is added with a batchsize of 120 and one is added of 80. The amount of KNN is increased to twenty to decrease the overfitting. RandomForest remains the same, because regardless it searches for the best/better outcome. The significant differences between datasets is shown with the (v) and less significant than that is the star (*) symbol.

All models will be compared with different comparison fields. Let’s also sort the results by pressing Sorting (asc.) by e.g. percent_correct. For each next run the “sort by“ will be applied with its corresponding comparison field.

The first comparison field will be percent_correct that determines in percentages how well the predictions are made by each model.

Percent_correct



```
| Dataset                  | (1) log_patient_data | (2) patient_data_ |
|--------------------------|----------------------|-------------------|
| bayes.NaiveBayes 'batch' | 72.7526316100        | 70.1368421900     |
| bayes.NaiveBayes'' 59952 | 72.7526316100        | 70.1368421900     |
| bayes.NaiveBayes 'batch' | 72.7526316100        | 70.1368421900     |
| rules.ZeroR'' 4805554146 | 75.3947369000        | 75.3947369000     |
| trees.J48 'C 0.25 M 2'   | 84.6289472600        | 84.7868420200     |
| lazy.IBk 'K 20 W 0 A'    | 84.7815788300        | 82.5236840600     |
| trees.J48'U M 3' 2177    | 84.9342104200        | 84.8315788500     |
| rules.OneR 'B 20 batch'  | 85.0026314700        | 85.1026314700     |
| rules.OneR'B 6' 345942   | 87.8052630800        | 87.6999999200     |
| trees.RandomForest 'P 10'| 91.7026315000        | 91.4052630800     |
| lazy.IBk'K 1 W 0 A'      | 93.0315788800        | 95.9052631200 v   |
|                          | (v/ /*)              | (1/10/0)          |
```

First of all generally the log_patient data does better than the normal patient data. Though there are no real significant differences in accuracy, except for the overfitted IBk.

J48 seems to increase in accuracy with a small amount with no pruning, which means that pruning removes some branches that are actually good. OneR’s accuracy decreases by about two after increasing the bucket size of six to twenty. Since the dataset has approximately 200 rows, ten buckets are made where the best attribute is chosen from. The fewer buckets are made in a dataset, the chance then increases for errors to be made.

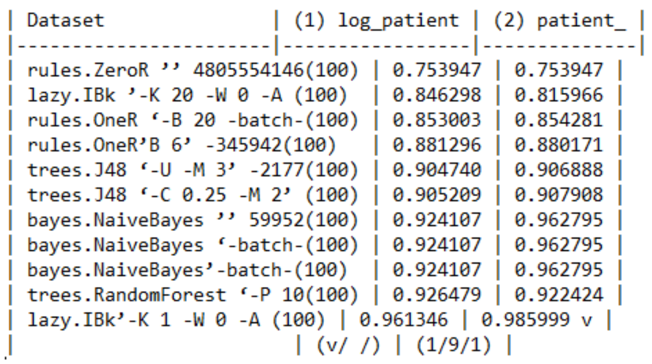
NaiveBayes does exactly the same, therefore increasing or decreasing the instances to process doesn't do anything significantly. IBk decreases in accuracy signifcantly which clearly shows it first was overfitted. This also means the most accurate model is RandomForest and the second best model is OneR. Since the lower the bucket size of OneR means more overfitting and a bucket size of twenty is only a bit less accurate than a bucket size of six, the edited OneR model will be used for feature selection later on.

Retrospective IBk is overfitted and therefore default RandomForest and OneR -B 6 are most accurate when edited models are included.

Precision

Which values are labeled True Positive and are actually True positive? I.e. how many of the instances that are actually sick are labeled sick? This is known as the Positive Predictive Value or precision. Therefore, choose IR_precision.

Dataset (1) log_patient | (2) patient_



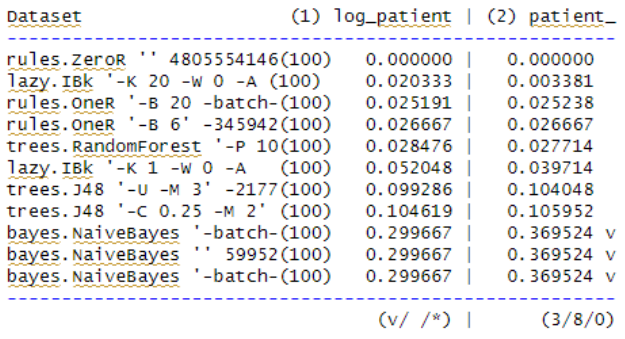
```
| Dataset                 | (1) log_patient | (2) patient_ |
|-------------------------|-----------------|--------------|
| rules.ZeroR '' 4805554146(100) | 0.753947 | 0.753947 |
| lazy.IBk '-K 20 -W 0 -A (100)  | 0.846298 | 0.815966 |
| rules.OneR '-B 20 -batch-(100) | 0.853003 | 0.854281 |
| rules.OneR'B 6' -345942(100)   | 0.881296 | 0.880171 |
| trees.J48 '-U -M 3' -2177(100) | 0.904740 | 0.906888 |
| trees.J48 '-C 0.25 -M 2' (100) | 0.905209 | 0.907908 |
| bayes.NaiveBayes '' 59952(100) | 0.924107 | 0.962795 |
| bayes.NaiveBayes '-batch-(100) | 0.924107 | 0.962795 |
| bayes.NaiveBayes'-batch-(100)  | 0.924107 | 0.962795 |
| trees.RandomForest '-P 10(100) | 0.926479 | 0.922424 |
| lazy.IBk'-K 1 -W 0 -A (100) | 0.961346 | 0.985999 v |
|                             | (v/ /) | (1/9/1) |
```

Assuming the higher the value the better the precision, the overfitted IBk stands number one in precision and the RandomForest tree model is second. Also, generally the precision is divided more evenly over the two datasets, but a bit better for the original data. Furthermore, it´s interesting to see that IBk -20 is next to ZeroR with weakest the precision. NaiveBayes does observably better on the original data.

Retrospective RandomForest and NaiveBayes have the best precision. The overfitted IBk will be ignored every retrospective.

False negative Rate

More importantly how many values have been assigned negative, but are actually positive i.e. is sick and is assigned healthy? Choose the False_negative_Rate.



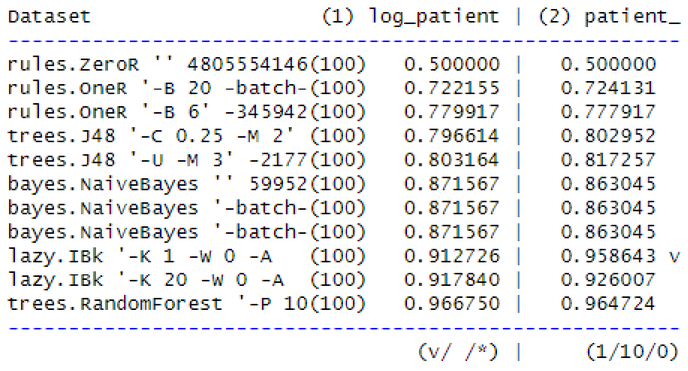
```
Dataset                   (1) log_patient | (2) patient_
-----------------------------------------------------------
rules.ZeroR '' 4805554146(100)   0.000000 |   0.000000
lazy.IBk '-K 20 -W 0 -A (100)    0.020333 |   0.003381
rules.OneR '-B 20 -batch-(100)   0.025191 |   0.025238
rules.OneR '-B 6' -345942(100)   0.026667 |   0.026667
trees.RandomForest '-P 10(100)   0.028476 |   0.027714
lazy.IBk '-K 1 -W 0 -A    (100)  0.052048 |   0.039714
trees.J48 '-U -M 3' -2177(100)   0.099286 |   0.104048
trees.J48 '-C 0.25 -M 2' (100)   0.104619 |   0.105952
bayes.NaiveBayes '-batch-(100)   0.299667 |   0.369524 v
bayes.NaiveBayes '' 59952(100)   0.299667 |   0.369524 v
bayes.NaiveBayes '-batch-(100)   0.299667 |   0.369524 v
-----------------------------------------------------------
                                  (v/ /*) |     (3/8/0)
```

Assuming the higher the value is the more mistakes are made, predicting is the worst for NaiveBayes. Obviously ZeroR makes the least mistakes, because it only chooses TP or TN. More interestingly IBk -20 makes the least mistakes and RandomForest has dipped down a bit.

Retrospective NaiveBayes predicts false negative the most, but IBK -20 and OneR -B 20 predict the least (ignoring ZeroR).

Area Under the (ROC) Curve (AUC)

Next let's set the comparison field to the Area Under the (ROC) Curve. This setting shows the extent of which a model can distinguish. Also, the higher the value is, the higher quality measure the model has.



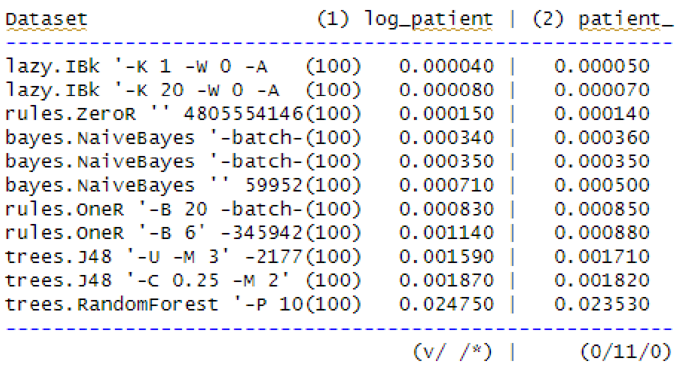
```
Dataset                   (1) log_patient | (2) patient_
-----------------------------------------------------------
rules.ZeroR '' 4805554146(100)   0.500000 |   0.500000
rules.OneR '-B 20 -batch-(100)   0.722155 |   0.724131
rules.OneR '-B 6' -345942(100)   0.779917 |   0.777917
trees.J48 '-C 0.25 -M 2' (100)   0.796614 |   0.802952
trees.J48 '-U -M 3' -2177(100)   0.803164 |   0.817257
bayes.NaiveBayes '' 59952(100)   0.871567 |   0.863045
bayes.NaiveBayes '-batch-(100)   0.871567 |   0.863045
bayes.NaiveBayes '-batch-(100)   0.871567 |   0.863045
lazy.IBk '-K 1 -W 0 -A    (100)  0.912726 |   0.958643 v
lazy.IBk '-K 20 -W 0 -A   (100)  0.917840 |   0.926007
trees.RandomForest '-P 10(100)   0.966750 |   0.964724
-----------------------------------------------------------
                                  (v/ /*) |    (1/10/0)
```

Assuming the higher the values the better the quality wherein a model can distinguish, RandomForest takes the price. Also, IBk -20 is the second best in distinguishing. From observing the previous comparison fields with the current one J48 is usually around the middle, which means it neither bad nor good for these datasets.

Retrospective ZeroR and OneR -B 20 are the worst at distinguishing. RandomForest and IBk -20 do the best.

Runtime (Elapsed_time_Training)

Also, how fast is the runtime of training for each model? Choose Elapsed_time_Training under comparison field. Since the models are generally really fast the runtime's will be close to zero. To solve results of only zero press Select next to Output Format and increase the mean and stDev precision rates until the values are comparable to the eye. For this data the rates are increased to six.



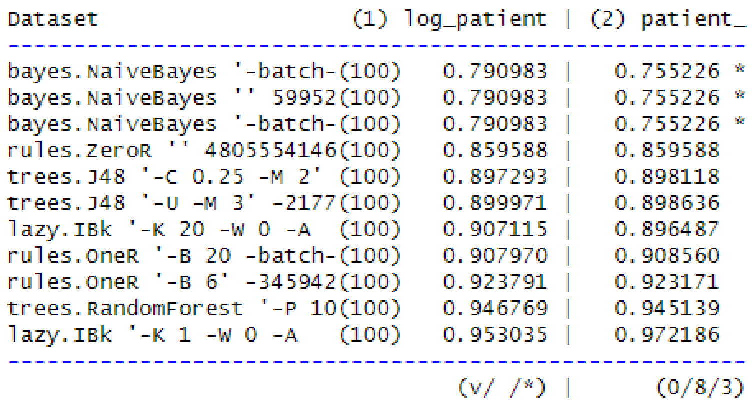
```
Dataset                      (1) log_patient | (2) patient_
-----------------------------------------------------------
lazy.IBk '-K 1 -W 0 -A    (100)   0.000040 |   0.000050
lazy.IBk '-K 20 -W 0 -A   (100)   0.000080 |   0.000070
rules.ZeroR '' 4805554146(100)   0.000150 |   0.000140
bayes.NaiveBayes '-batch-(100)   0.000340 |   0.000360
bayes.NaiveBayes '-batch-(100)   0.000350 |   0.000350
bayes.NaiveBayes '' 59952(100)   0.000710 |   0.000500
rules.OneR '-B 20 -batch-(100)   0.000830 |   0.000850
rules.OneR '-B 6' -345942(100)   0.001140 |   0.000880
trees.J48 '-U -M 3' -2177(100)   0.001590 |   0.001710
trees.J48 '-C 0.25 -M 2' (100)   0.001870 |   0.001820
trees.RandomForest '-P 10(100)   0.024750 |   0.023530
-----------------------------------------------------------
                                   (v/ /*) |     (0/11/0)
```

Assuming that the runtime is in seconds and zero is the fastest both IBk's take the cherry on the cake. As expected the trees perform worse. For J48 making branches takes more time and RandomForest is additionally to making branches also run several times with different coincidence values, which takes even more time.

Retrospective Both IBk have the fastest runtimes and both trees have the slowest runtimes. Though no significant differences.

F_measure

Since the values are distributed unevenly 3-1 ratio, a different way of evaluating the accuracy of the model can be chosen. By choosing F_measure, the levels are weighted evenly between both precision (PPV) and recall (TPR). The values lie between zero and one, where zero has no relevant information gain and one if all information gained is relevant. Look at the bibliography under F-measure for the complete definition by SpringerLink.



```
Dataset                    (1) log_patient | (2) patient_
------------------------------------------------------------
bayes.NaiveBayes '-batch-(100)   0.790983 |   0.755226 *
bayes.NaiveBayes '' 59952(100)   0.790983 |   0.755226 *
bayes.NaiveBayes '-batch-(100)   0.790983 |   0.755226 *
rules.ZeroR '' 4805554146(100)   0.859588 |   0.859588
trees.J48 '-C 0.25 -M 2' (100)   0.897293 |   0.898118
trees.J48 '-U -M 3' -2177(100)   0.899971 |   0.898636
lazy.IBk '-K 20 -W 0 -A  (100)   0.907115 |   0.896487
rules.OneR '-B 20 -batch-(100)   0.907970 |   0.908560
rules.OneR '-B 6' -345942(100)   0.923791 |   0.923171
trees.RandomForest '-P 10(100)   0.946769 |   0.945139
lazy.IBk '-K 1 -W 0 -A   (100)   0.953035 |   0.972186
------------------------------------------------------------
                                  (v/ /*) |      (0/8/3)
```

Next to the overfitted IBk, RandomForest has the second best relevant information gain and OneR places third. NaiveBayes and ZeroR have the least relevant information gain.

Retrospective RandomForest and default OneR have the best relevant information gain.

Retrospective of all comparisons

On average RandomForest does the best. It does worse for runtime, but it's still really fast and therefore redundant. OneR -B 20 usually also does quite good. Since OneR -B 6 is known for overfitting, OneR -B 20 will be placed second best model for these datasets.

Observing accuracy file

The run time data of can also be saved to csv by choosing CSV file under "Results destination", giving the desired location and then run the setup again.

Let's see what the run data looks like.

```
experimenter.two.files <- read.csv("Experimenter/default_two_files.csv", header = T, sep = ",")
head(experimenter.two.files)

#preview

##   Number_of_training_instances Number_of_testing_instances Number_correct

## 1                    175                 20           15

## 2                    175                 20           15

## 3                    175                 20           15

## 4                    175                 20           15

## 5                    175                 20           15

## 6                    176                 19           15
```

There are quite a lot of interesting columns in this file, basically they're all statistics in how the model has performed. For example it shows that 175 of the instances are used for training and twenty for training. Furthermore, it shows how many number of times the levels are guessed correctly and incorrectly. The means of the data are also shown.

Select features in explorer

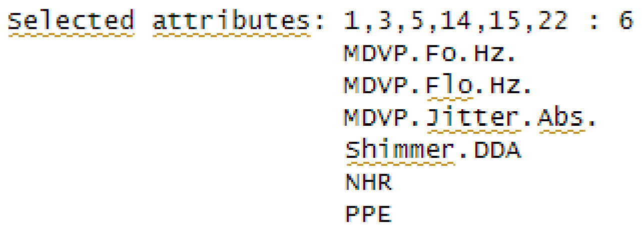
This chapter is for searching the best attributes (features). Earlier several attributes were found with interesting behaviours. By applying attribute selection to the dataset the best "attributes" can be found.

Single-attribute evaluation

This evaluation removes features that are less informative than the top informative features.

WrapperSubsetEval

Since the most accurate model was RandomForest lets choose in "Select Attributes" under "Attribute evaluator": WrapperSubsetEval and for "Search method": BestFirst by pressing WrapperSubsetEval a window pops out. RandomForest is chosen as classifier and "accuracy" as evaluationMeasure for discrete class only. Since the log data was better for RandomForest it will be chosen for this run.



```
Selected attributes: 1,3,5,14,15,22 : 6
                     MDVP.Fo.Hz.
                     MDVP.Flo.Hz.
                     MDVP.Jitter.Abs.
                     Shimmer.DDA
                     NHR
                     PPE
```

Some of these attributes that are found remarkably coincide with what attributes where labeled interesting in the EDA chapter. Primarily the best features are MDVP.Fo.Hz. and MDVP.Flo.Hz.

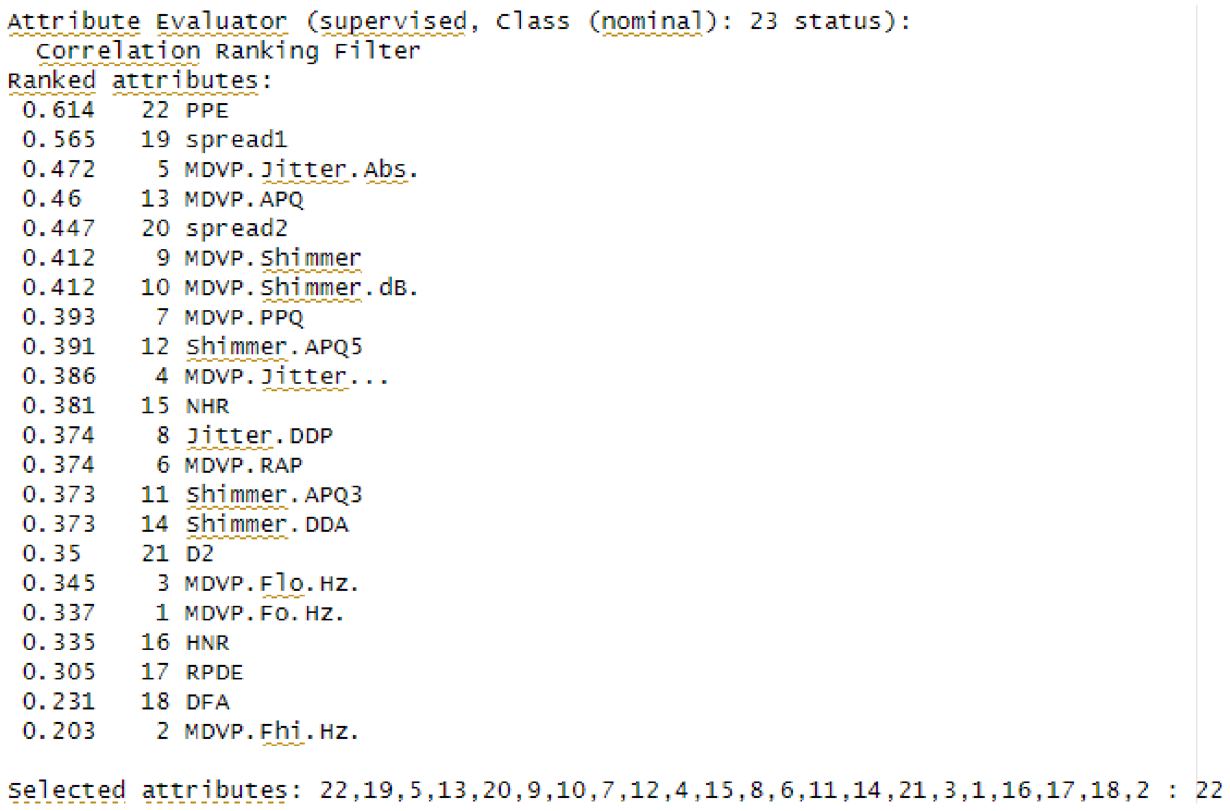
Let's now look at how OneR -B 20 performs with the same settings as above.

Selected attributes: 22 : 1 PPE

OneR finds PPE as the best attribute which is also included by RandomForest.

CorrelationAttributeeval

With the Correlation attribute evaluator and ranker search method a ranking can be made of the extent how relevant the attributes are according to distinguishing the class status.

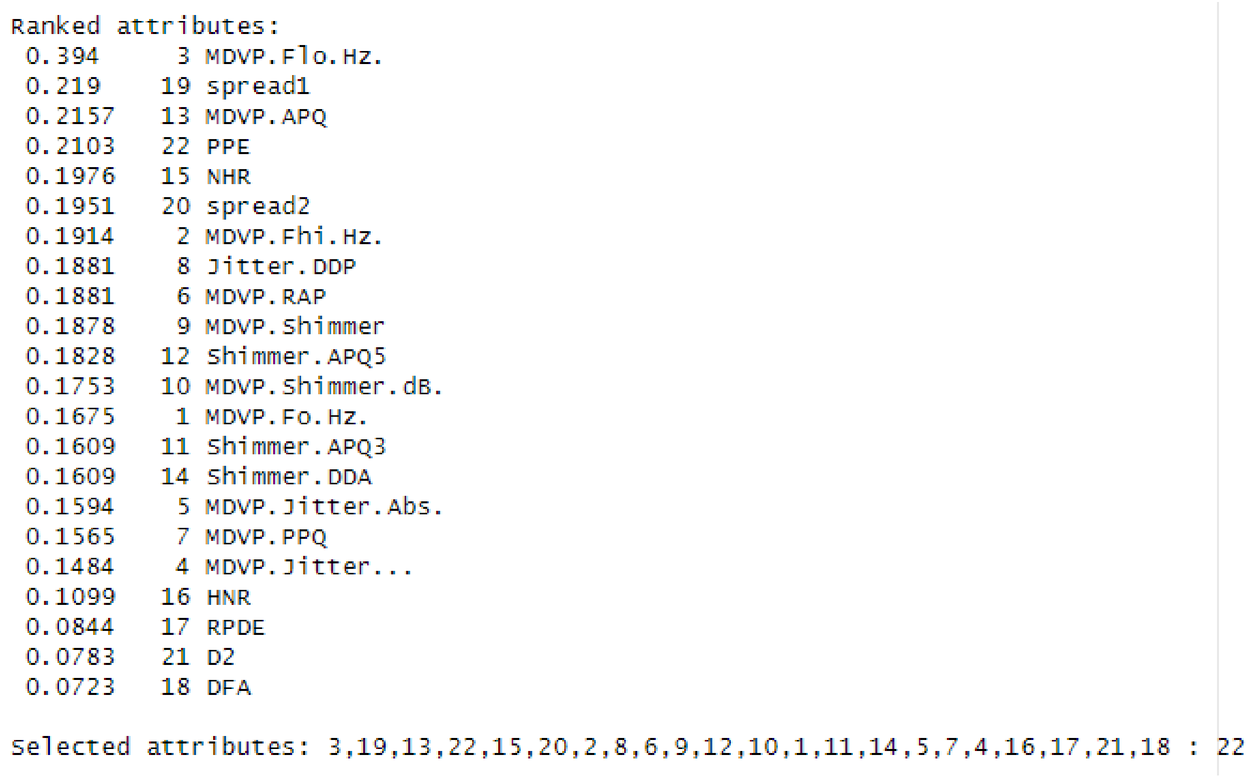


```
Attribute Evaluator (supervised, Class (nominal): 23 status):
  Correlation Ranking Filter
Ranked attributes:
 0.614   22 PPE
 0.565   19 spread1
 0.472    5 MDVP.Jitter.Abs.
 0.46    13 MDVP.APQ
 0.447   20 spread2
 0.412    9 MDVP.Shimmer
 0.412   10 MDVP.Shimmer.dB.
 0.393    7 MDVP.PPQ
 0.391   12 Shimmer.APQ5
 0.386    4 MDVP.Jitter...
 0.381   15 NHR
 0.374    8 Jitter.DDP
 0.374    6 MDVP.RAP
 0.373   11 Shimmer.APQ3
 0.373   14 Shimmer.DDA
 0.35    21 D2
 0.345    3 MDVP.Flo.Hz.
 0.337    1 MDVP.Fo.Hz.
 0.335   16 HNR
 0.305   17 RPDE
 0.231   18 DFA
 0.203    2 MDVP.Fhi.Hz.

Selected attributes: 22,19,5,13,20,9,10,7,12,4,15,8,6,11,14,21,3,1,16,17,18,2 : 22
```

Assuming towards zero is worse than towards one PPE and MDVP.Jitter.Abs return in the top three. Also, in the EDA spread1 was a notable attribute and now it is ranked as second best attribute.

GainRatioAttributeEval

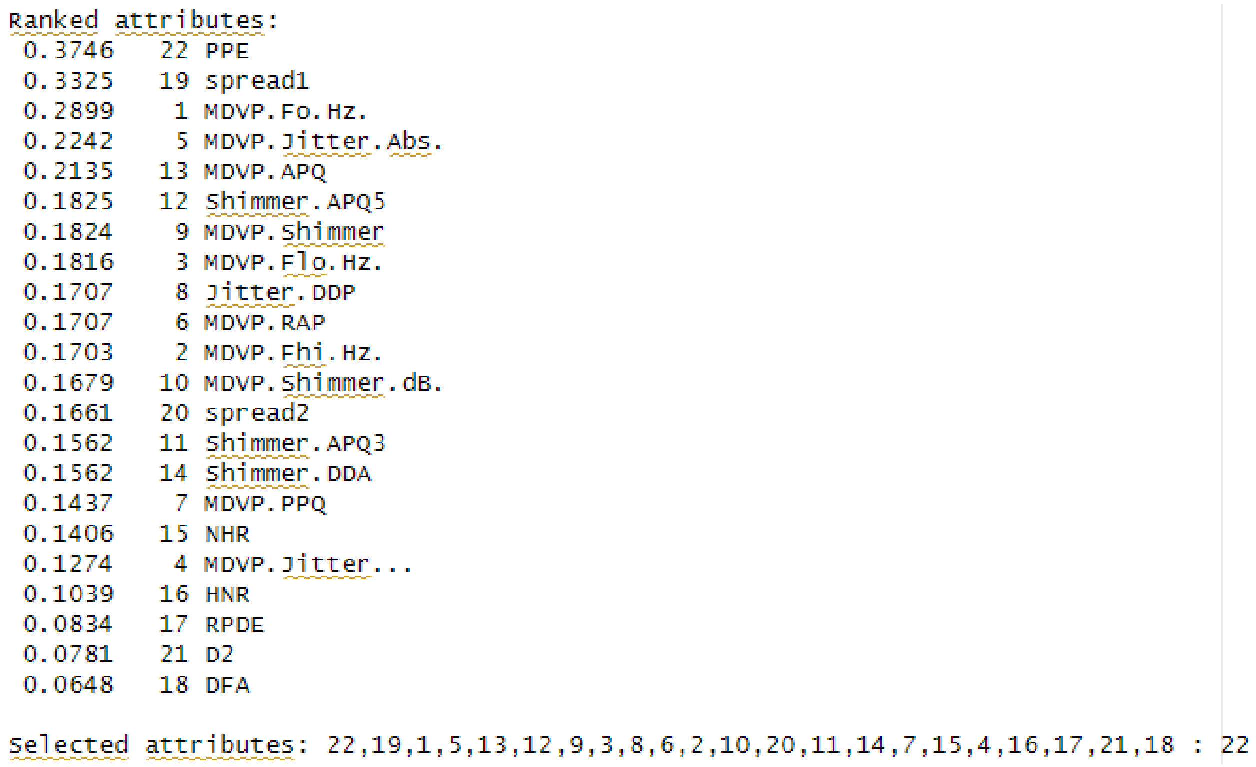
For GainRatio let’s also choose Ranker as a search method.



```
Ranked attributes:
 0.394      3 MDVP.Flo.Hz.
 0.219     19 spread1
 0.2157    13 MDVP.APQ
 0.2103    22 PPE
 0.1976    15 NHR
 0.1951    20 spread2
 0.1914     2 MDVP.Fhi.Hz.
 0.1881     8 Jitter.DDP
 0.1881     6 MDVP.RAP
 0.1878     9 MDVP.Shimmer
 0.1828    12 Shimmer.APQ5
 0.1753    10 MDVP.Shimmer.dB.
 0.1675     1 MDVP.Fo.Hz.
 0.1609    11 Shimmer.APQ3
 0.1609    14 Shimmer.DDA
 0.1594     5 MDVP.Jitter.Abs.
 0.1565     7 MDVP.PPQ
 0.1484     4 MDVP.Jitter...
 0.1099    16 HNR
 0.0844    17 RPDE
 0.0783    21 D2
 0.0723    18 DFA

Selected attributes: 3,19,13,22,15,20,2,8,6,9,12,10,1,11,14,5,7,4,16,17,21,18 : 22
```

GainRatio ranks MDVP.Flo.Hz, spread1 and MDVP.APQ in the top three. MDVP.APQ was also high in CorrelationAttributeeval. And PPE and NHR follow the top three.

InfoGainAttributeEval



```
Ranked attributes:
 0.3746   22 PPE
 0.3325   19 spread1
 0.2899    1 MDVP.Fo.Hz.
 0.2242    5 MDVP.Jitter.Abs.
 0.2135   13 MDVP.APQ
 0.1825   12 Shimmer.APQ5
 0.1824    9 MDVP.Shimmer
 0.1816    3 MDVP.Flo.Hz.
 0.1707    8 Jitter.DDP
 0.1707    6 MDVP.RAP
 0.1703    2 MDVP.Fhi.Hz.
 0.1679   10 MDVP.Shimmer.dB.
 0.1661   20 spread2
 0.1562   11 Shimmer.APQ3
 0.1562   14 Shimmer.DDA
 0.1437    7 MDVP.PPQ
 0.1406   15 NHR
 0.1274    4 MDVP.Jitter...
 0.1039   16 HNR
 0.0834   17 RPDE
 0.0781   21 D2
 0.0648   18 DFA

Selected attributes: 22,19,1,5,13,12,9,3,8,6,2,10,20,11,14,7,15,4,16,17,21,18 : 22
```

InformationGain evaluates PPE, spread1 and MDVP.Fo.Hz. in the top three. Again MDVP.Jitter.Abs. and MDVP.APQ remain high in the rankings.

OneRAttributeEval

This evaluator evaluates each attribute using OneR. Again the bucketsize is set to twenty, though the batchsize here is immutable. Also, let's choose full training set. As search method choose Ranker.

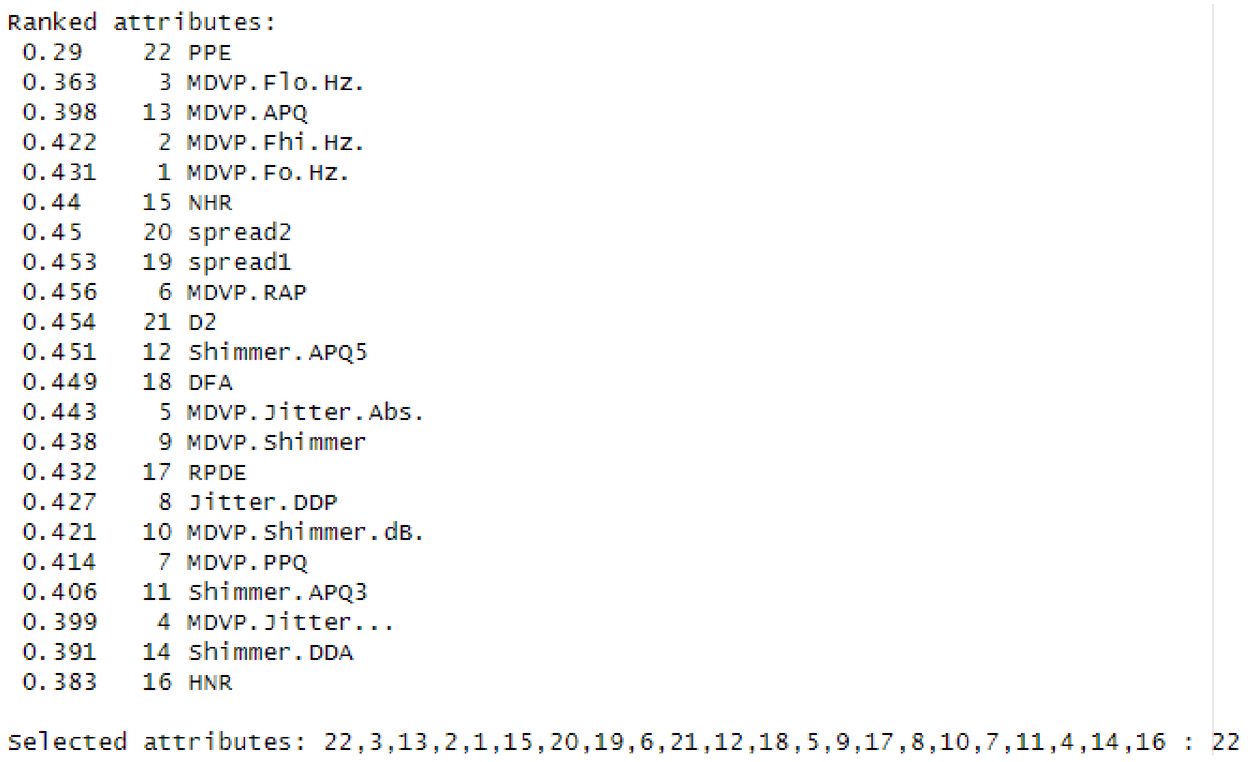
```
Ranked attributes:
88.2051   22 PPE
84.6154    1 MDVP.Fo.Hz.
83.5897    5 MDVP.Jitter.Abs.
83.0769   19 spread1
81.0256    3 MDVP.Flo.Hz.
79.4872    7 MDVP.PPQ
77.9487   15 NHR
76.9231    2 MDVP.Fhi.Hz.
76.9231    4 MDVP.Jitter...
76.4103    8 Jitter.DDP
74.8718    6 MDVP.RAP
73.8462   13 MDVP.APQ
73.8462   20 spread2
73.8462   16 HNR
73.3333   17 RPDE
72.3077   11 Shimmer.APQ3
72.3077   14 Shimmer.DDA
70.7692   21 D2
69.7436   18 DFA
69.7436   10 MDVP.Shimmer.dB.
68.2051   12 Shimmer.APQ5
66.6667    9 MDVP.Shimmer

Selected attributes: 22,1,5,19,3,7,15,2,4,8,6,13,20,16,17,11,14,21,18,10,12,9 : 22
```

Again attribute PPE is the best and generally the same attributes remain top. Though MDVP.Shimmer has dropped down all the way to the bottom.

Attribute-subset evaluation

This evaluation evaluates groups of attributes together. Let's choose GreedyStepwise for search method and turn generateRanking to True.

CfsSubsetEval

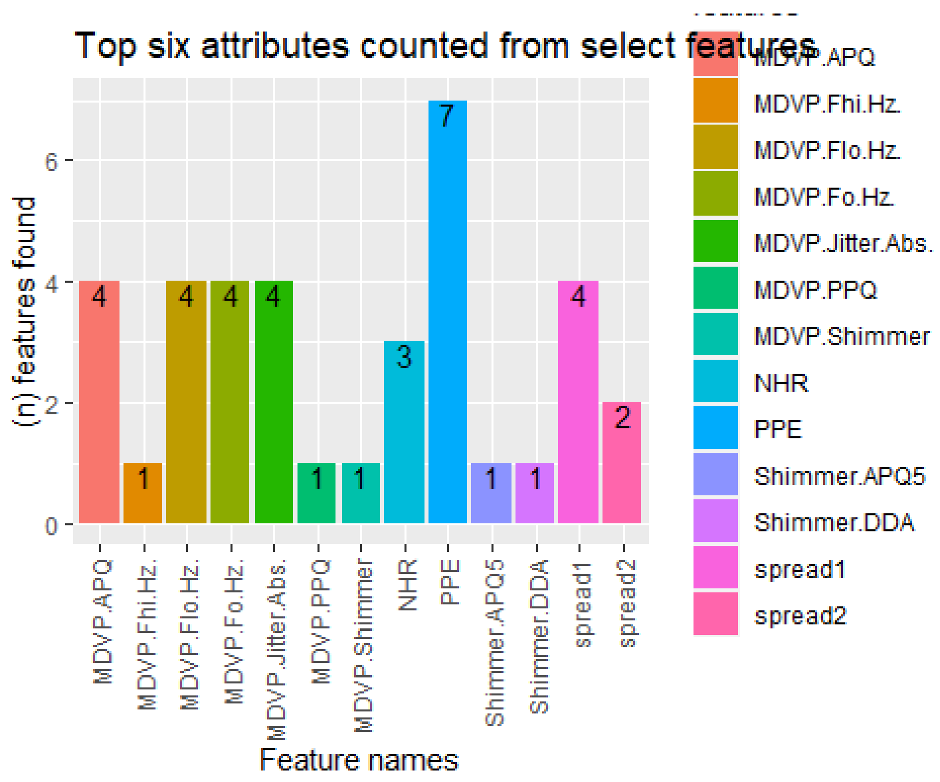


```
Ranked attributes:
 0.29     22 PPE
 0.363     3 MDVP.Flo.Hz.
 0.398    13 MDVP.APQ
 0.422     2 MDVP.Fhi.Hz.
 0.431     1 MDVP.Fo.Hz.
 0.44     15 NHR
 0.45     20 spread2
 0.453    19 spread1
 0.456     6 MDVP.RAP
 0.454    21 D2
 0.451    12 Shimmer.APQ5
 0.449    18 DFA
 0.443     5 MDVP.Jitter.Abs.
 0.438     9 MDVP.Shimmer
 0.432    17 RPDE
 0.427     8 Jitter.DDP
 0.421    10 MDVP.Shimmer.dB.
 0.414     7 MDVP.PPQ
 0.406    11 Shimmer.APQ3
 0.399     4 MDVP.Jitter...
 0.391    14 Shimmer.DDA
 0.383    16 HNR

Selected attributes: 22,3,13,2,1,15,20,19,6,21,12,18,5,9,17,8,10,7,11,4,14,16 : 22
```

Again PPE is at the top and MDVP.APQ has risen to the third place. All MDVP.Flo/Fhi/- and Fo.Hz attributes are at the top as well. Also, NHR and spread1 are fairly high as well.

Let's count the occurrences by eye of all top six attributes and make a bar plot to visualize the best attributes all round. The counts are added in a file called "best_features_counts.txt".

```
best.features.counts <- read.csv("best_features_counts.csv", header = T)

ggplot(best.features.counts, aes(x = features, fill = features)) + geom_bar() + xlab("Feature names") +
  ylab("(n) features found") + labs(title = "Top six attributes counted from select features") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) + geom_text(stat='count',
aes(label=after_stat(count)), vjust=1)
```



In the bar plot it becomes clear that five attributes: MDVP.APQ, MDVP.Flo.Hz, MDVP.Fo.Hz, MDVP.Jitter.Abs and spread1 occurred an equal amount of times in the top six. Attribute PPE occurred in the top six of all evaluations in select features, which is seven evaluations that have been performed.

Retrospective The observation from Cfs actually summarizes the general best placement for all attributes per evaluation method. After comparing all methods PPE seems the best attribute to split on and spread1 as second attribute. Attributes MDVP.Flo.Hz and MDVP.Jitter.Abs are also mainly quite good.

Baselearners & increasing performance

There are several methods for combining algorithms and increasing performance.

Vote

With the vote meta classifier algorithms results can be combined for a new outcome via voting. Choose classifiers in the vote menu and add the preferred classifiers. So for this experiment RandomForest and OneR with -B 20 and batchsize 120. Also, let's choose Majority voting for combinationRule.

Correctly Classified Instances 165 84.6154 % Incorrectly Classified Instances 30 15.3846 %

The accuracy is less than RandomForest on its own.

a b <– classified as 142 5 | a = sick 25 23 | b = healthy

Though very few false negatives remain.

PPE: < -2.25923967465801 -> healthy '>= -2.25923967465801 -> sick (168/195 instances correct)

PPE was found as the best attribute to split on and a clear boundary was found to differentiate healthy from sick.

Bagging

A classifier for reducing variance. Choose the preferred classifier in the Bagging menu.

RandomForest
Correctly Classified Instances 180 92.3077 % Incorrectly Classified Instances 15 7.6923 %

From the experiment it became clear that default RandomForest = 91.7026315000 | 91.4052630800. The bagging method increases the accuracy of RandomForest by about one percent.

a b <– classified as 144 3 | a = sick 12 36 | b = healthy

The amount of FP and FN's have decreased in comparison to the combination of RandomForest and OneR.

OneR

Correctly Classified Instances 162 83.0769 % Incorrectly Classified Instances 33 16.9231 %

From the experiment it became clear that the edited OneR = 85.0026314700 | 85.1026314700. Therefore, bagging affects the results negatively by about two percent.

a b <– classified as 141 6 | a = sick 27 21 | b = healthy

The amount of FP and FN's have increased in comparison to the combination of RandomForest and OneR.

Stacking

A classifier for assessing the outcomes of algorithms using a meta-learner, which then tries to provide new predictions. The classifiers can be added under classifiers in the Stacking menu and the meta-learner can be chosen under metaClassifier. Let's choose J48 with unpruning to True, the confidenceFactor to 0.20 and the minNumObj to three as meta-learner.

Correctly Classified Instances 173 88.7179 % Incorrectly Classified Instances 22 11.2821 %

J48 was 84.9342104200 | 84.8315788500 percent and therefore has increase by about four percent. Eventhough it didn't surpass RandomForest, it did surpass OneR.

If J48 and OneR are swapped from meta-learner to classifier i.e. if OneR learns from J48 and RandomForest, how will that affect OneR's results?

Correctly Classified Instances 174 89.2308 % Incorrectly Classified Instances 21 10.7692 %

OneR increases by four percent and surpasses the previous J48 meta-learner with approximately 0.5 percent.

Retrospective Using base learning techniques RandomForest and OneR's performance can be increased. The stacking method didn't work positively though stacking and bagging did. Using bagging RandomForest was increased and using stacking OneR increased by meta-learning. The quality of these new models will be tested in the next part.

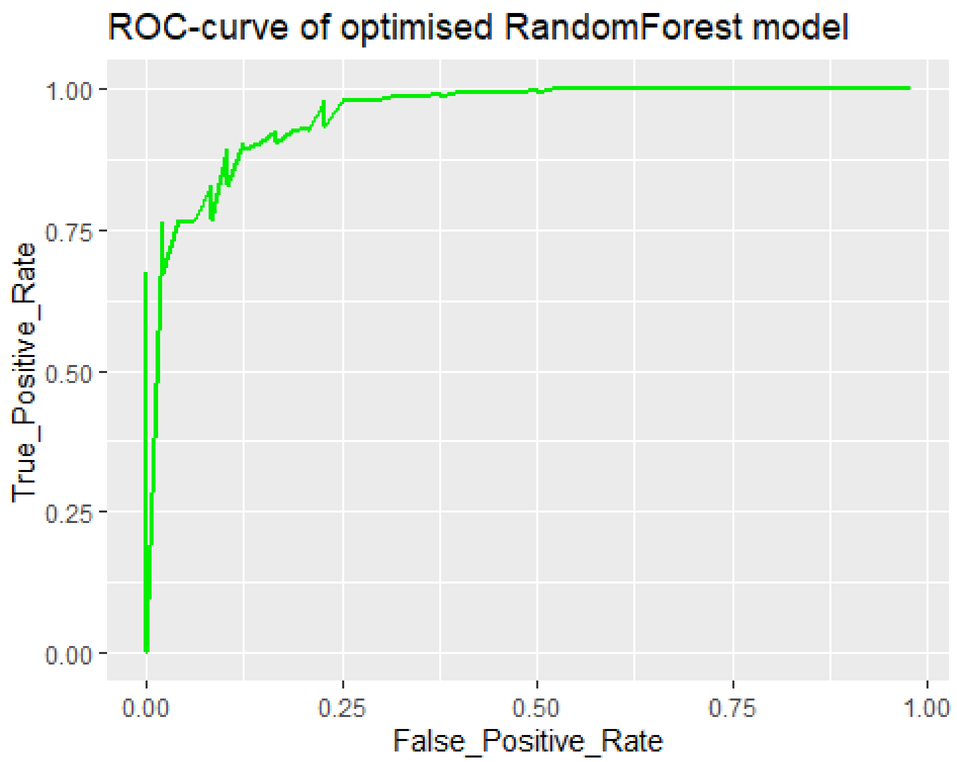
The model runs will be saved for plotting through the experimenter where each model is saved separately. This is done the same way as in part "Observing accuracy file".

ROC-curve

Of each model-run the quality can be determined by making a ROC-curve. This is done by plotting FPR (specificity) against TPR (sensitivity).

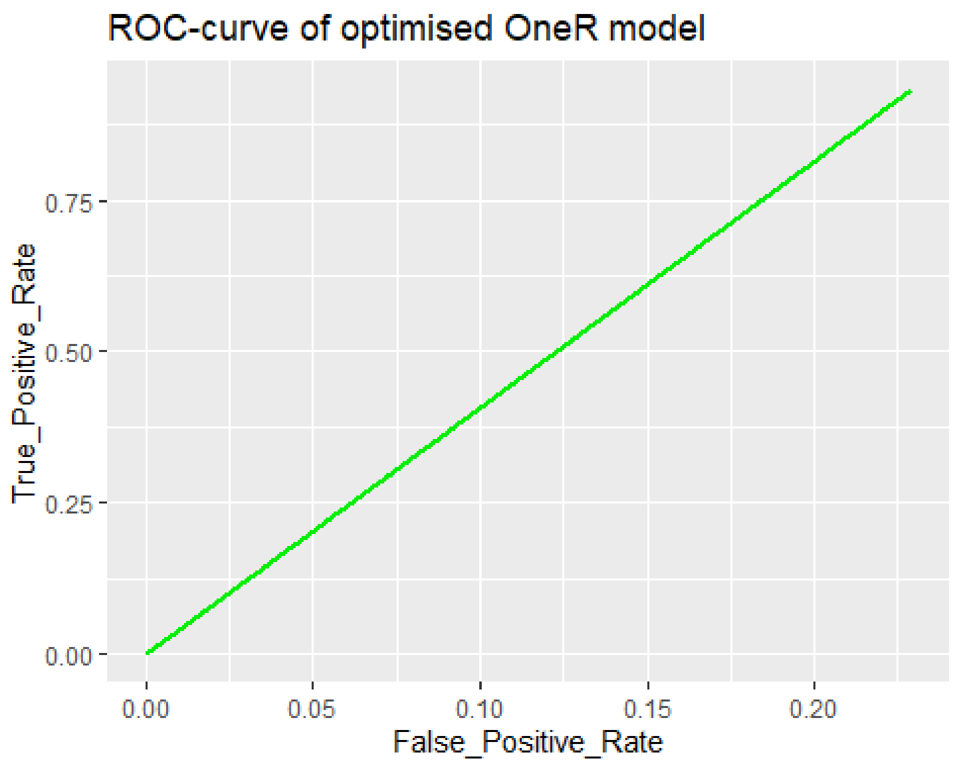
RandomForest + Bagging

```
randomForest.data <- data.frame(read.csv("Final_models/RF_bagging.arff", skip = 18))
colnames(randomForest.data) <- c("Instance_number", "True_Positives",
"False_Negatives","False_Positives",
                    "True_Negatives", "False_Positive_Rate", "True_Positive_Rate",
                    "Precision", "Recall", "Fallout", "FMeasure", "Sample_Size",
                    "Lift", "Threshold")
ggplot(randomForest.data, aes(x = False_Positive_Rate, y = True_Positive_Rate)) + geom_line(color =
"green2", size = 1) +
  labs(title = "ROC-curve of optimised RandomForest model")
```



OneR + Stacking

```
oneR.data <- data.frame(read.csv("Final_models/OneR_stacking.arff", skip = 18))
colnames(oneR.data) <- c("Instance_number", "True_Positives", "False_Negatives","False_Positives",
                    "True_Negatives", "False_Positive_Rate", "True_Positive_Rate",
                    "Precision", "Recall", "Fallout", "FMeasure", "Sample_Size",
                    "Lift", "Threshold")
ggplot(oneR.data, aes(x = False_Positive_Rate, y = True_Positive_Rate)) + geom_line(color = "green2",
size = 1) +
  labs(title = "ROC-curve of optimised OneR model")
```



Optimize Random-forest bagging

From selecting features and testing WrapperSubsetEval on Random forest the following 6 attributes ranked best : MDVP.Fo.Hz., MDVP.Flo.Hz., MDVP.Jitter.Abs., Shimmer.DDA, NHR, PPE.

What if we only use these attributes and tell bagging - random forest to choose a certain amount of random attributes? Go to the settings of random forest inside the bagging menu and set the option numFeatures to seven (including status attribute).

Correctly Classified Instances 182 93.3333 % Incorrectly Classified Instances 13 6.6667

a b <– classified as 142 5 | a = sick 8 40 | b = healthy

The accuracy has increased by one percent. After testing other options it seems this will be the optimized model to use for predicting on the data.

Retrospective A final model has been found with a 93.3333 percent accuracy; a by bagging optimized random forest model that chooses seven random attributes.

References

- Indeed editorial team (04-02-2023). *Negative Correlation: Definition and Examples (With Types)*

Link : https://www.indeed.com/career-advice/career-development/negative-correlation-definition-and-examples

- Kaggle - Yunfeng, W. et al. (03-05-2017). *Parkinson's Disease Data Set*

Link : https://www.kaggle.com/datasets/vikasukani/parkinsons-disease-data-set?resource=download

- Yunfeng, W. et al. (03-05-2017). *Research from diagnosing people with PD using vocal recordings*

Link : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5434464/

- Zhang, E., Zhang, Y. (2009). *F-Measure*

Link : https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_483