

Logbook of the heartdisease dataset

Sander J. Bouwman

27/09/2021

Contents

1	Logbook EDA	2
1.1	Dataset information	2
1.1.1	Research question	2
1.2	Dataparsing	2
1.3	Summary	3
1.4	Visualisations	4
1.4.1	Datapoints per sex	4
1.4.2	Histogram	5
1.4.3	Heartdisease linked to age	6
1.5	Plotting pain types with different variables	7
1.5.1	Pain types table	7
1.5.2	Chestpain and their outcome	8
1.5.3	Resting blood pressure split by pain type	9
1.5.4	The effect of chestpain on cholesterol or vice versa	10
1.5.5	Is the max heart rate affected by certain chestpain?	11
1.5.6	Density of ages split by sex	13
1.5.7	Finding correlation between MaxHR and HD	14
1.6	PCA plotting	15
1.7	Algorithms	15
1.8	Analysis of multiple ML algorithms	18

1 Logbook EDA

1.1 Dataset information

Kaggle direct link to dataset: <https://www.kaggle.com/fedesoriano/heart-failure-prediction>

1.1.1 Research question

Is it possible to create an accurate (F/P. $\leq 20\%$ & F/N $\leq 10\%$) machine learning model that predicts the chance of developing heartdisease in a broad spectrum of patients?

Due to the potential damage false negatives (F/N) will cause, it is important to reduce the F/N. Although there is a balance between F/N and false positive (F/P) A high F/P will cause unnecessary testing and will increase costs, while a high F/N will be potential dangerous as it misses true positive (T/P) patients.

1.2 Dataparsing

```
heartdata <- read.table("data/heart.csv", sep=";", header=TRUE)
heartdata$HeartDisease <- as.logical(heartdata$HeartDisease)
heartdata$FastingBS <- as.logical(heartdata$FastingBS)

flags = data.frame(Reduce(cbind,
  lapply(levels(heartdata$ChestPainType), function(x){(heartdata$ChestPainType == x)*1})
))
colnames(flags) <- c("ASY", "ATA", "NAP", "TA")
heartdata$ASY = as.logical(flags$ASY)
heartdata$ATA = as.logical(flags$ATA)
heartdata$NAP = as.logical(flags$NAP)
heartdata$TA = as.logical(flags$TA)

flags = data.frame(Reduce(cbind,
  lapply(levels(heartdata$ST_Slope), function(x){(heartdata$ST_Slope == x)*1})
))
colnames(flags) <- c("Flat", "Down", "Up")
heartdata$ST_Slope_Flat = as.logical(flags$Flat)
heartdata$ST_Slope_Down = as.logical(flags$Down)
heartdata$ST_Slope_Up = as.logical(flags$Up)

# Changes Y/N to Boolean
heartdata$ExerciseAngina = heartdata$ExerciseAngina == "Y"
heartdata$FastingBS = as.logical(heartdata$FastingBS)

s <- sapply(heartdata, typeof)
sr <- data.frame("attribute" = colnames(heartdata),
  "type" = sapply(heartdata, typeof),
  "class" = sapply(heartdata, class),
  row.names = NULL)
codebook <- as_tibble(sr)
kable(codebook, caption = "Codebook") %>%
  kable_styling(latex_options = c("hold_position"))
```

Table 1: (`#tab:get data types`)Codebook

attribute	type	class
Age	integer	integer
Sex	integer	factor
ChestPainType	integer	factor
RestingBP	integer	integer
Cholesterol	integer	integer
FastingBS	logical	logical
RestingECG	integer	factor
MaxHR	integer	integer
ExerciseAngina	logical	logical
Oldpeak	double	numeric
ST_Slope	integer	factor
HeartDisease	logical	logical
ASY	logical	logical
ATA	logical	logical
NAP	logical	logical
TA	logical	logical
ST_Slope_Flat	logical	logical
ST_Slope_Down	logical	logical
ST_Slope_Up	logical	logical

1.3 Summary

```
summ <- summary(heartdata[c("Age", "RestingBP", "Cholesterol", "MaxHR", "Oldpeak")])
tab <- sub('.*:', '', summ)
rownames(tab) <- c("Min", "1st. Qu", "Median", "Mean", "3rd. Qu", "Max")
kable(tab, caption = "5 num of numerical valies") %>%
  kable_styling(latex_options="scale_down")
```

Table 2: 5 num of numerical valies

	Age	RestingBP	Cholesterol	MaxHR	Oldpeak
Min	28.00	0.0	0.0	60.0	-2.6000
1st. Qu	47.00	120.0	173.2	120.0	0.0000
Median	54.00	130.0	223.0	138.0	0.6000
Mean	53.51	132.4	198.8	136.8	0.8874
3rd. Qu	60.00	140.0	267.0	156.0	1.5000
Max	77.00	200.0	603.0	202.0	6.2000

1.4 Visualisations

1.4.1 Datapoints per sex

```
# counts (or sums of weights)
ggplot(heartdata) +
  geom_bar(aes(Sex, fill=HeartDisease)) +
  labs(title = "Datapoints per sex") +
  ylab("Datapoints")
```

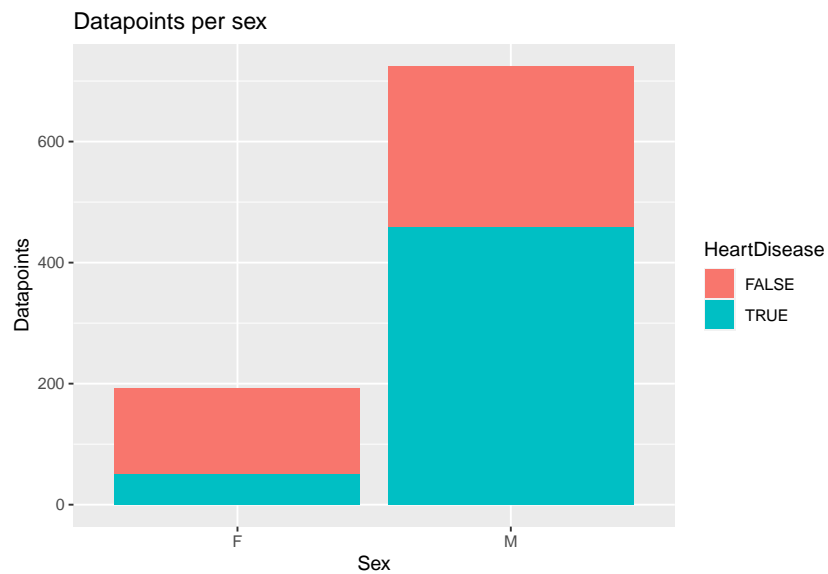


Figure 1: Ratio male to female

(#fig:Ratio male to female)

```
p <- prop.table(table(heartdata$Sex))
```

As visible in the barplot, the ratio of female(21.02%) to male(78.98%) is very skewed towards males. Females have a much lower proportion of positive heartdiseases, whereas males are more likely to have heartdisease in this dataset.

1.4.2 Histogram

```
ggplot(heartdata, aes(Age, fill=HeartDisease)) +  
  geom_histogram(bins=12) +  
  geom_vline(aes(xintercept=mean(Age[HeartDisease==F])), colour="Mean age without HD", size=1, linetype="dashed") +  
  geom_vline(aes(xintercept=mean(Age[HeartDisease==T])), colour="Mean age with HD", size=1, linetype="dashed") +  
  scale_color_manual(name="Means", values = c("dodgerblue1", "red")) +  
  guides(col = guide_legend(override.aes = list(shape = 15, size=5))) +  
  ggtitle("Histogram of age while also visualising HD")
```

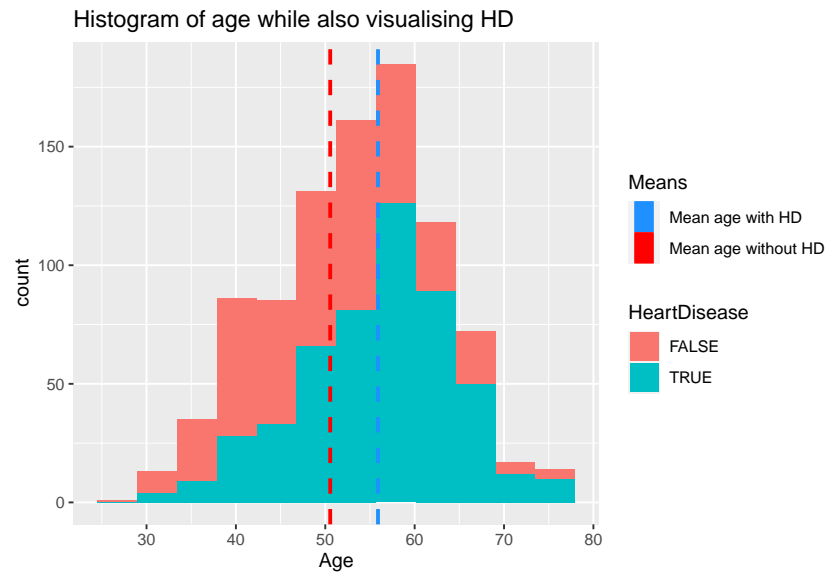


Figure 2: Histogram of age grouped by HD diagnoses

As seen in the above graph, we can discover that there seems to be a relation between age increase and an increase in heartdisease. We will explore this further in the underlying graph. When watching the age mean lines it is further seen that the average age of patients with HD is higher then the average patient without HD.

1.4.3 Heartdisease linked to age

```
ggplot(heartdata) +  
  geom_smooth(aes(Age, HeartDisease * 100), method="loess", formula = "y ~ x", color=hue_pal()(1), fill="white", size=1) +  
  ylab("Heart Disease (%)") +  
  ggtitle("Percentage of patients affected by HD by age") +  
  ylim(c(0, 100))
```

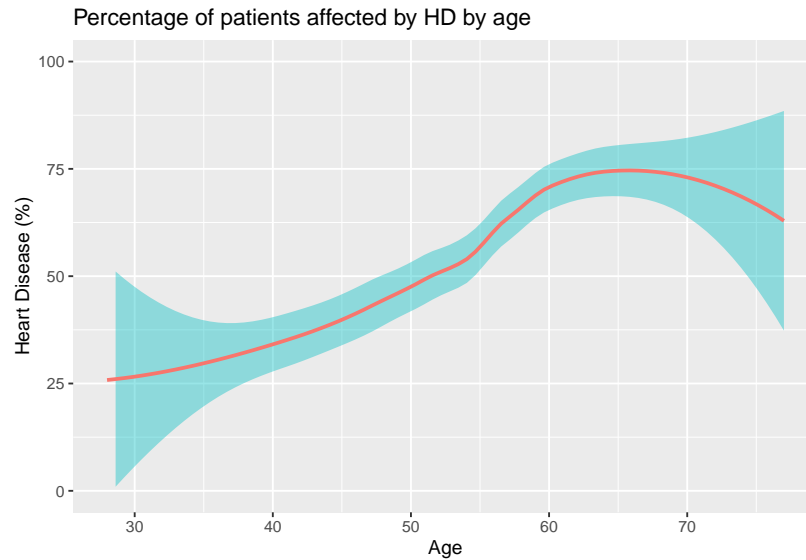


Figure 3: Percentage of patients affected by HD by age

Using a Smooth line we can see that the percentage of patients with HD increases with age and declines after 65. The reason for decline might be that patients who have HD above the age of 65 will die earlier than healthy individuals. A short internet search will also confirm that HD is the leading cause of death for people above the age of 65. Underlying image provides further insight. Source: ec.europa.eu

```
# Define variable containing url  
url <- "https://ec.europa.eu/eurostat/statistics-explained/images/2/25/Major\_causes\_of\_death%2C\_EU-27%2C\_2014.png"
```

1.5 Plotting pain types with different variables

1.5.1 Pain types table

The graphs below use abbreviations. The meaning of these can be found in this table:

Pain type:

Code	Type	Explanation
TA	Typical Angina	Meets all three of the following characteristics: 1. Substernal chest discomfort of characteristic quality and duration 2. Provoked by exertion or emotional stress 3. Relieved by rest and/or nitrates within minutes.
ATA	Atypical Angina	Meets two of the above characteristics
NAP	Non-Anginal Pain	Lacks or meets only one or none of the characteristics
ASY	Asymptomatic Pain	

1.5.2 Chestpain and their outcome

```
ggplot(heartdata) +  
  geom_bar(aes(ChestPainType, fill=HeartDisease)) + ggtitle("Chestpaintype outcome")
```

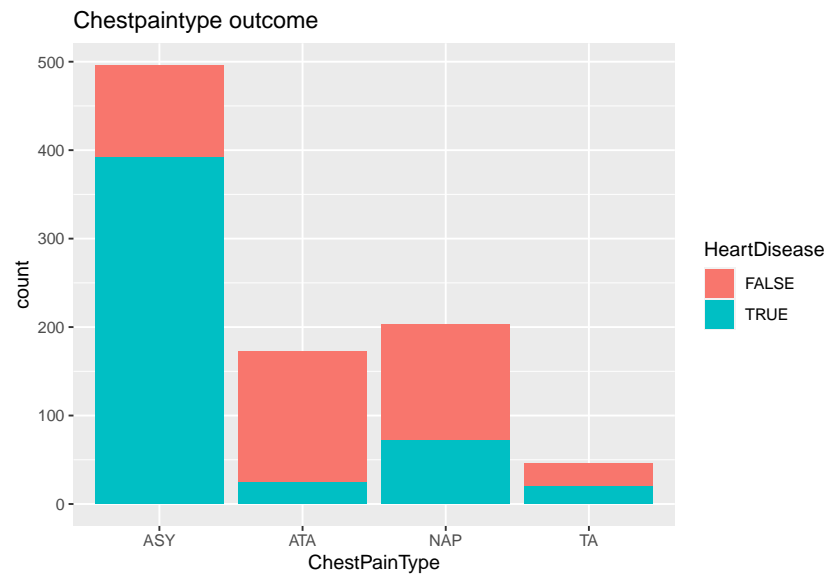


Figure 4: Ratio of diagnosis grouped by chestpain type

As visible in the above barplot, asymptomatic pain (ASY) is the most frequently reported and has the highest probability of HD.

1.5.3 Resting blood pressure split by pain type

```
p <- ggplot(heartdata, aes(x=RestingBP, group=ChestPainType, colour=ChestPainType)) +  
  geom_density() +  
  xlab("Resting BP (mm Hg)") +  
  ggtitle("Density plot of RestingBP split by ChestPainType")
```

p

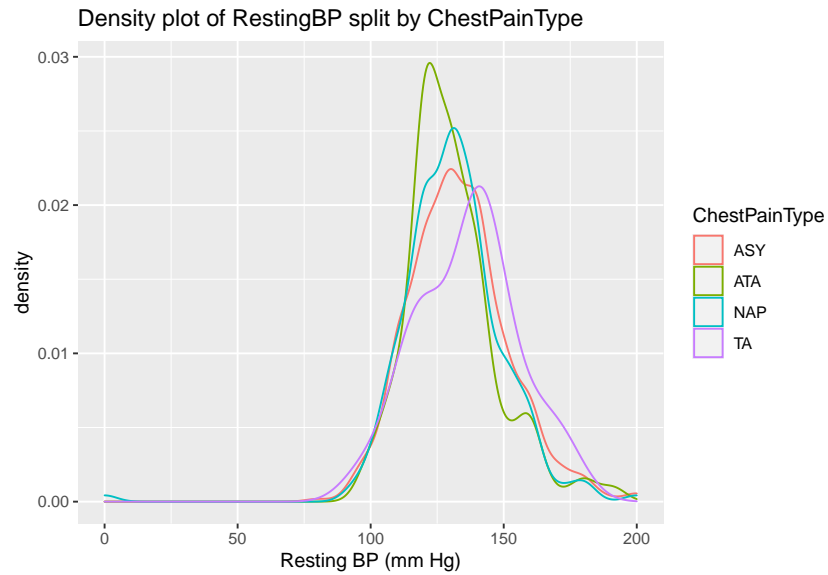


Figure 5: Density plot of Resting BP by pain type

The type of pain an individual experiences doesn't seem to change the blood pressure of the individual.

1.5.4 The effect of chestpain on cholesterol or vice versa

```
p <- ggplot(heartdata, aes(x=Cholesterol, group=ChestPainType, colour=ChestPainType)) +  
  geom_density() +  
  xlab("Cholesterol in serum (mm/dl)") +  
  ggtitle("Density plot of cholesterol in serum split by ChestPainType")
```

p

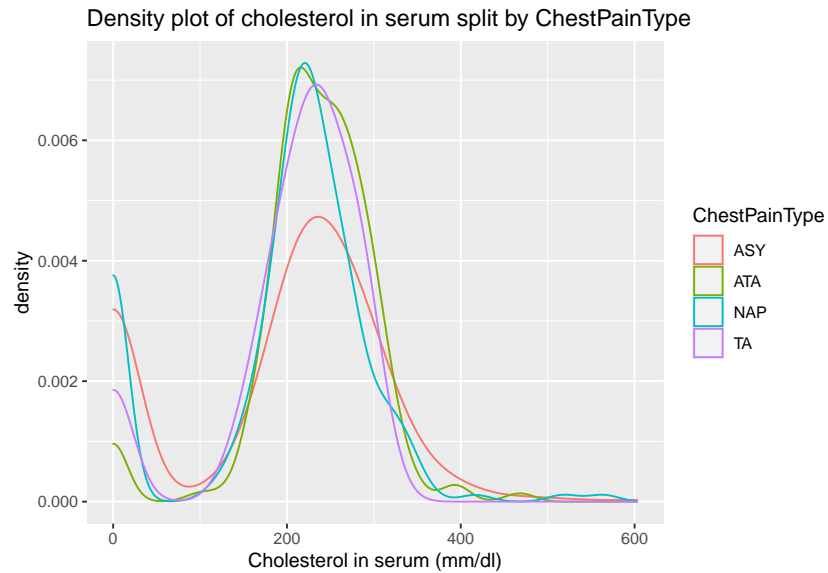


Figure 6: Density plot of cholesterol by pain type

As visible in the above plot. The type of chestpain doesn't seem to differ based on the concentration of cholesterol in blood.

1.5.5 Is the max heart rate affected by certain chestpain?

```
p <- ggplot(heartdata, aes(x=MaxHR, group=ChestPainType, colour=ChestPainType)) +  
  geom_density() +  
  xlab("Max heartrate") +  
  ggtitle("Density plot of MaxHR split by ChestPainType")
```

p

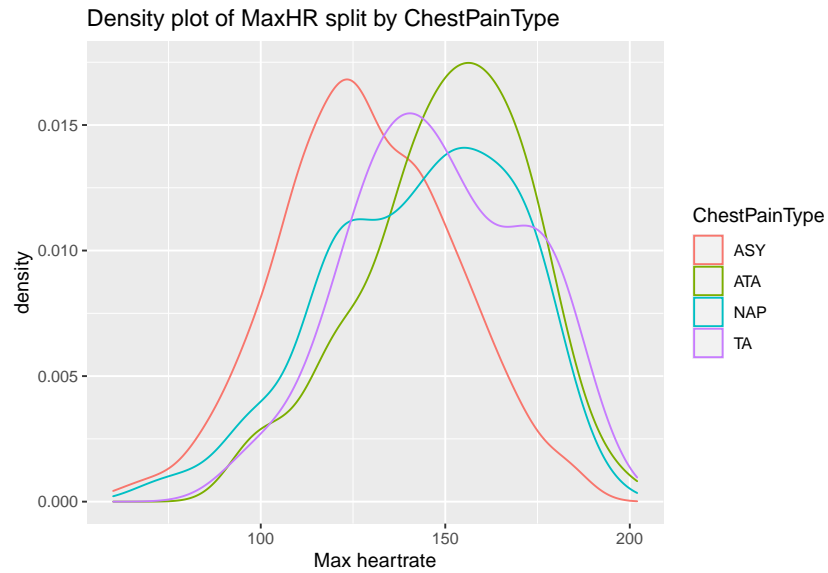


Figure 7: Density plot of maxHR by pain type

There seems to be a small gap in max heartrate with patients ASY (Asymptomatic) vs ATA (Atypical Angina).

```
### Age vs MaxHR grouped by Sex
```

```
ggplot(heartdata, aes(Age, MaxHR, group=Sex, colour = Sex)) +  
  geom_point(alpha=0.15) + geom_smooth(se=F) + ggtitle("Comparing MaxHR with age split by sex")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

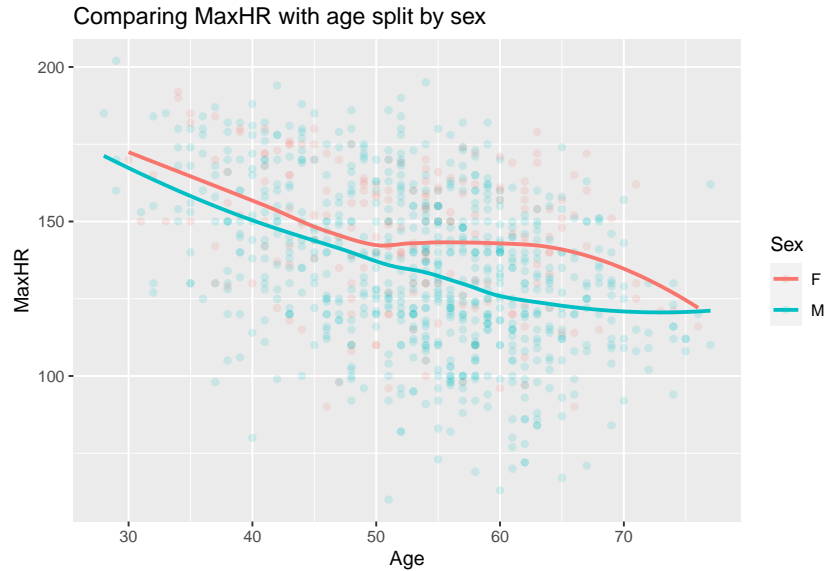


Figure 8: Max heartrate by sex and age

(#fig:Age-Sex vs MaxHR)

As expected MaxHR goes down with age. The female lines doesnt go down as expected, this might be due to the fact there is a smaller amount of data for older females(Age > 55 = 71) in comparison with males (Age > 55 = 335). To further analyse the difference in datapoints per sex, a density plot is made.

1.5.6 Density of ages split by sex

```
ggplot(heartdata) +  
  geom_density(aes(Age, colour=Sex, fill=Sex), alpha=0.05) +  
  ggtitle("Density plot comparing sexes")
```

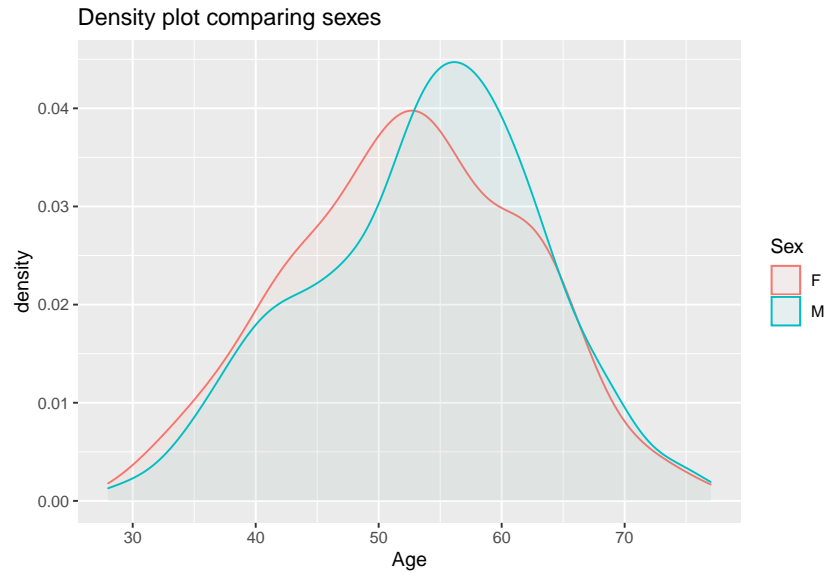


Figure 9: Amount of datapoints per age grouped by sex

The above graph shows that there is not much difference in age comparing male and female.

1.5.7 Finding correlation between MaxHR and HD

```
ggplot(heartdata[heartdata$Cholesterol > 0,], aes(x=Age, y=Cholesterol, colour=HeartDisease, shape=ChestPainType)) +  
  geom_point() +  
  ggtitle("Age vs Cholesterol") +  
  geom_abline(color = "dodgerblue1")
```

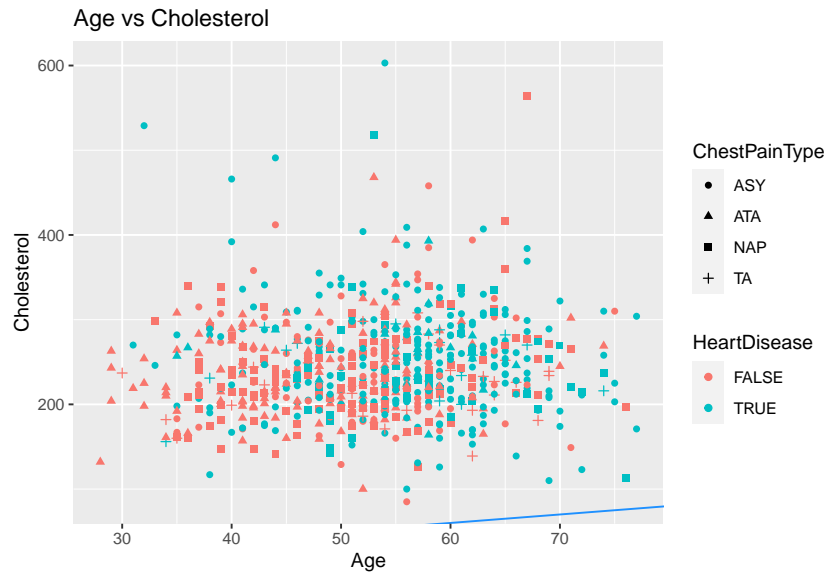


Figure 10: Correlation between age and cholesterol

The amount of cholesterol increases slightly with age.

1.6 PCA plotting

```
heartdata.active = heartdata[, c(1,4,5,6,8,13:16)]
res.pca <- prcomp(heartdata.active, scale = TRUE)

fviz_pca_var(res.pca,
  col.var = "contrib",
  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
  repel = TRUE
)
```

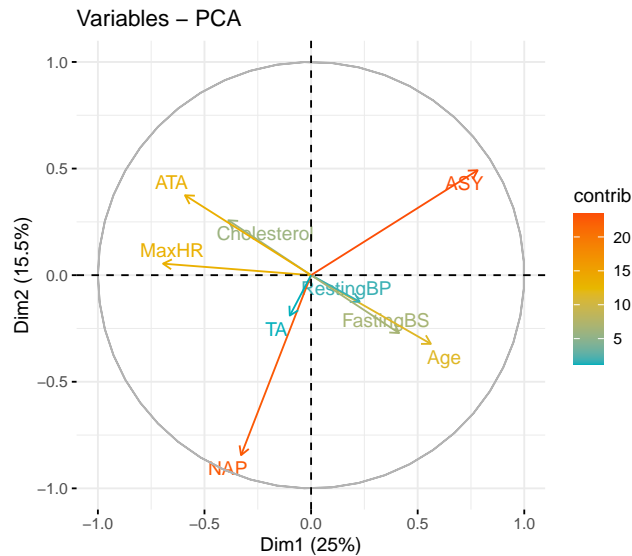


Figure 11: PCA Biplot of all non-categorical variables

```
fviz_pca_biplot(res.pca,
  col.var = "#2E9FDF", # Variables color
  col.ind = heartdata$HeartDisease,
  label = T,
  addEllipses = TRUE,
  geom.var = c("arrow", "text"),
  ellipse.type = "t",
  axes=c(1,2),
  legend.title = "Heartdisease")
```

Using the above lying PCA graphs it seems that asymptomatic pain is the biggest contributor in the diagnosis of heartdisease in this dataset. This makes it hard to predict HD as asymptomatic patients have no (visible) symptoms. The other pain categories do not seem to have a direct relation with the presence of HD. So far we can expect that age is the best variable in predicting HD as the percentage goes from around 25% at age ~30 to 75% at age ~65.

1.7 Algorithms

```
write.csv(heartdata, "data/heart_cleaned_report.csv", sep=";", row.names = FALSE)
heartdata$ChestPainType <- NULL
heartdata$ST_Slope <- NULL

heartdata$Cholesterol[heartdata$Cholesterol == 0] <- "?"
```

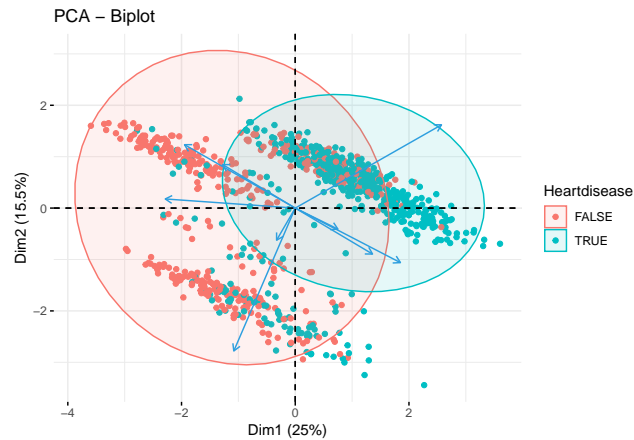


Figure 12: PCA Variable plot of all non-categorical variables

```
heartdata$RestingBP[heartdata$RestingBP == 0] <- "?"

heartdata$Cholesterol <- as.integer(heartdata$Cholesterol)
heartdata$RestingBP <- as.integer(heartdata$RestingBP)
## Set Heartdata as last column before exporting
disease <- heartdata$HeartDisease
heartdata$HeartDisease <- NULL
heartdata$HeartDisease <- disease
write.arff(heartdata, "data/heart_cleaned.arff")
write.csv(heartdata, "data/heart_cleaned.csv", sep=",", row.names = FALSE)
```

Autoparsing data from weka text files to csv so that it can be used in R.

```
import os
import pandas as pd

path = "data/weka_data"
output_path = "data/weka_parse.csv"

pdcons = []
for file in os.listdir(path):
    if "txt" not in file:
        continue
    fileinfo = {}
    with open(os.path.join(path, file), "r") as stream:
        fileinfo["name"] = file.split(".")[0]
        for line in stream:
            if line.startswith("Correctly Classified Instances "):
                fileinfo["accuracy"] = line.split(" ")[-2]
            elif line.startswith("Incorrectly Classified Instances "):
                fileinfo["incorrect"] = line.split(" ")[-2]
            elif line.startswith("Scheme: "):
                fileinfo["parameters"] = line.split("Scheme: ")[1].replace("\n", "")
        pdcons.append(fileinfo)
dataframe = pd.DataFrame.from_dict(pdcons)
dataframe.to_csv(output_path, index=False)
```



```
mlresults <- read.table("data/weka_parse.csv", sep=";", header=T)

ggplot(mlresults, aes(x=reorder(name, -accuracy), y=accuracy)) +
  geom_bar(stat = "identity", fill="dodgerblue1") +
  coord_flip() + ylim(c(0,100)) + ggtitle("Accuracy of models") + xlab("Algorithm") + ylab("Accuracy (%)")

## Warning: Removed 1 rows containing missing values (position_stack).
```

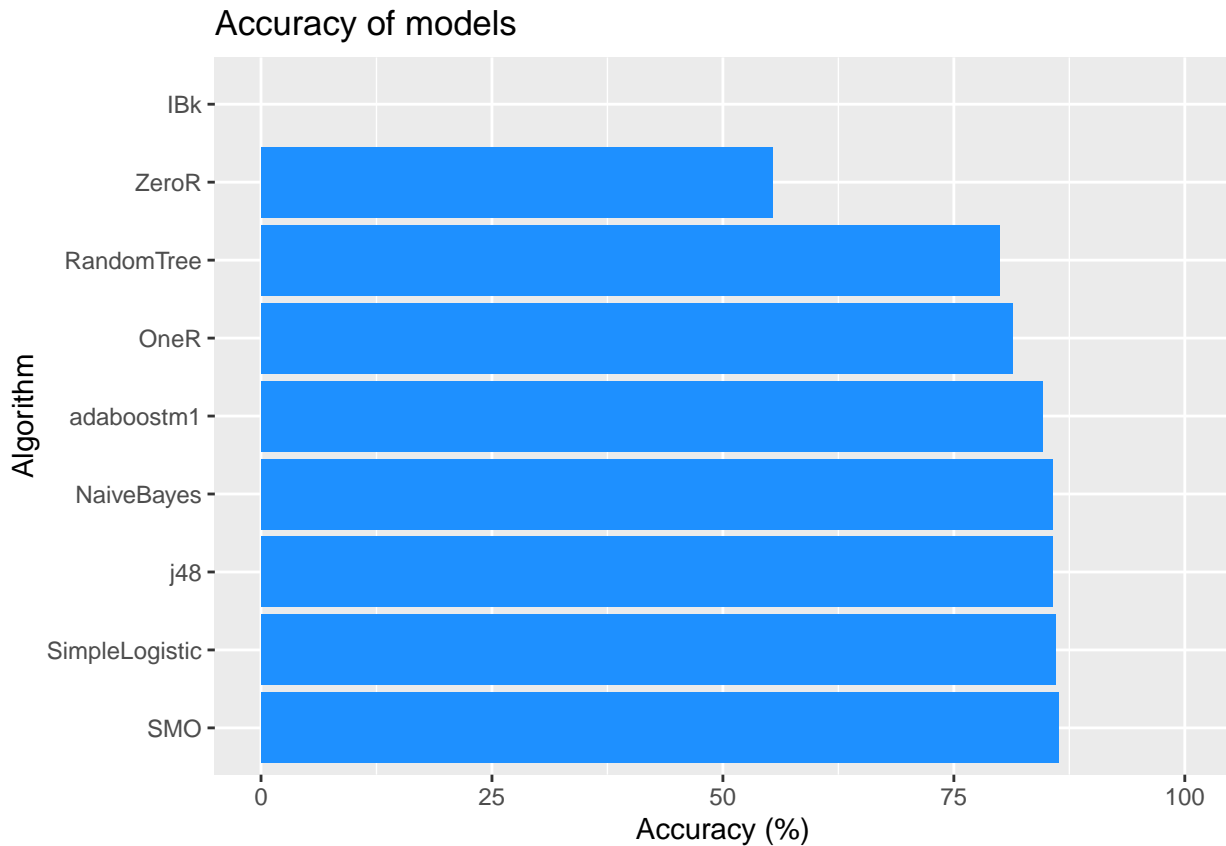


Figure 13: Accuracy by algorithm

```
kable(mlresults[order(mlresults$accuracy, decreasing = T), ], col.names = c("Algorithm", "Run Parameters"))
```

	Algorithm	Run Parameters
9	SMO	weka.classifiers.functions.SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.ZeroR"
7	SimpleLogistic	weka.classifiers.functions.SimpleLogistic -I 0 -M 500 -H 50 -W 0.0
1	NaiveBayes	weka.classifiers.bayes.NaiveBayes
6	j48	weka.classifiers.trees.J48 -C 0.25 -M 15
5	adaboostm1	weka.classifiers.meta.AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump
2	OneR	weka.classifiers.rules.OneR -B 6
8	RandomTree	weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1
3	ZeroR	weka.classifiers.rules.ZeroR
4	IBk	weka.classifiers.lazy.IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance\""

1.8 Analysis of multiple ML algorithms

A fair amount of algorithms are significantly better than ZeroR, with all of them having an accuracy $\geq 80\%$.

```
algorithms <- c("ZeroR", "OneR", "J48", "RandomTree", "AdaBoostM1", "SMO", "SimpleLogistic", "NaiveBayes")
correct_p <- c(55.36, 81.37, 83.35, 79.89, 86.03, 86.00, 85.76, 85.33, 79.24)
sig_vs_zeror <- c(F, T, T, T, T, T, T, T, T)
sig_vs_ada <- c(T, T, T, T, F, F, F, F, T)
experimenter <- data.frame(algorithm = algorithms, accuracy = correct_p, significance_zeror = sig_vs_zeror, significance_ada = sig_vs_ada)
experimenter <- tibble::rowid_to_column(experimenter, "Key")
```

```
ggplot(experimenter, aes(x=reorder(algorithm, accuracy), y=accuracy, fill=significance_ada)) +
  geom_bar(stat = "identity") +
  coord_flip() + ylim(c(0,100)) + ggtitle("Accuracy of models") + xlab("Algorithm") + ylab("Accuracy (%)")
```

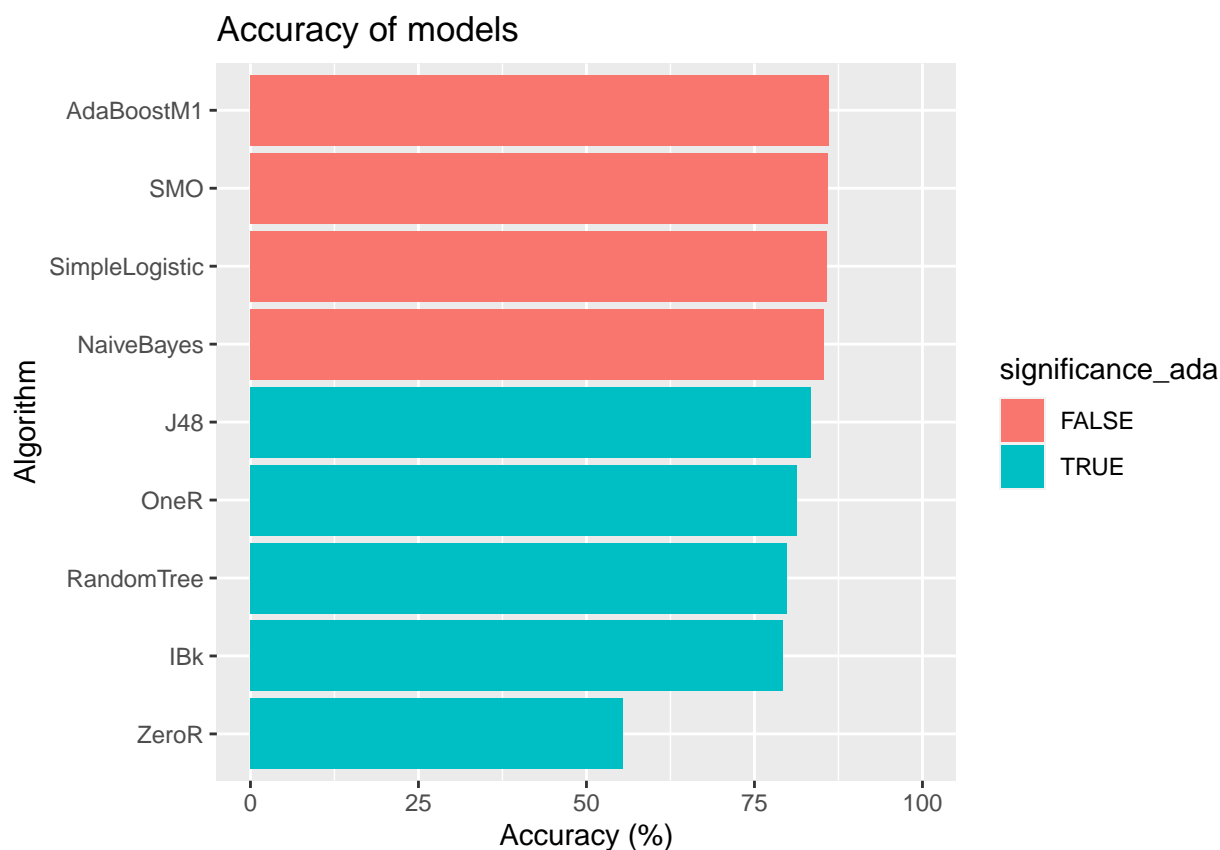


Figure 14: Accuracy of various models. Coloured by significance

```
kable(experimenter[order(-correct_p),], col.names = c("Key", "Algorithm", "Accuracy (%)", "Significant"),
      kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Key	Algorithm	Accuracy (%)	Significant different then ZeroR	Significant different then AdaBoostM1
5	AdaBoostM1	86.03	TRUE	FALSE
6	SMO	86.00	TRUE	FALSE
7	SimpleLogistic	85.76	TRUE	FALSE
8	NaiveBayes	85.33	TRUE	FALSE
3	J48	83.35	TRUE	TRUE
2	OneR	81.37	TRUE	TRUE
4	RandomTree	79.89	TRUE	TRUE
9	IBk	79.24	TRUE	TRUE
1	ZeroR	55.36	FALSE	TRUE

As we can see in the above table (figure ??) AdaBoostM1 scores, although there is no significant difference with non-meta algorithms like SMO and SimpleLogistic. Other ensemble learners will be tested now.

```

algorithms <- c("ZeroR", "Bagging", "Stacking", "Vote", "AdaBoostM1", "SMO", "SimpleLogistic", "NaiveBayes")
accuracy <- c(55.34, 84.54, 85.53, 86.20, 85.82, 85.73, 85.68, 85.40)
stdeviation <- c(0.2, 3.61, 3.53, 3.37, 3.49, 3.46, 3.51, 3.32)
stat_dif_ada <- c(T, F, F, F, F, F, F, F)

ensemble <- data.frame(algorithm = algorithms, accuracy = accuracy, stdeviation = stdeviation, significance_ada = stat_dif_ada)
ensemble <- tibble::rowid_to_column(ensemble, "Key")

ggplot(ensemble, aes(x=reorder(algorithms, accuracy), y=accuracy, fill=significance_ada)) +
  geom_bar(stat = "identity") +
  coord_flip() + ylim(c(0,100)) + ggtitle("Accuracy of models") + xlab("Algorithm") + ylab("Accuracy (%)")

```



Params: (1) rules.ZeroR " 48055541465867954

```
(2) meta.Bagging '-P 100 -S 1 -num-slots 1 -I 10 -W trees.REPTree -M 2 -V 0.001 -N 3 -S 1 -L -1 -I 0.0'
-115879962237199703
(3) meta.Stacking '-X 10 -M "trees.J48 -C 0.25 -M 2" -S 1 -num-slots 1 -B "functions.SMO -C 1.0 -L
0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -E 1.0 -C 250007\" -calibrator
\"functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4\" -B "functions.SimpleLogistic -I 0 -M 500 -H 50
-W 0.0" -B "bayes.NaiveBayes " 5134738557155845452
(4) meta.Vote '-S 1 -B "meta.AdaBoostM1 -P 100 -S 1 -I 10 -W trees.DecisionStump" -B "functions.SMO
-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -E 1.0 -C 250007\"
-calibrator \"functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4\" -B "functions.SimpleLogistic -I 0 -M
500 -H 50 -W 0.0" -B "bayes.NaiveBayes " -R AVG' -637891196294399624
(5) meta.AdaBoostM1 '-P 100 -S 1 -I 10 -W trees.DecisionStump' -1178107808933117974
(6) functions.SMO '-C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "functions.supportVector.PolyKernel -E
1.0 -C 250007" -calibrator "functions.SimpleLogistic -I 0 -M 500 -H 50 -W 0.0" -6585883636378691736
(7) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0' 7397710626304705059
(8) bayes.NaiveBayes " 5995231201785697655
```

```
kable(ensemble[order(-accuracy),] , col.names = c("Key", "Algorithm", "Accuracy (%)", "Standard deviation", "Significant different then AdaBoostM1"),
      kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Key	Algorithm	Accuracy (%)	Standard deviation	Significant different then AdaBoostM1
4	Vote	86.20	3.37	FALSE
5	AdaBoostM1	85.82	3.49	FALSE
6	SMO	85.73	3.46	FALSE
7	SimpleLogistic	85.68	3.51	FALSE
3	Stacking	85.53	3.53	FALSE
8	NaiveBayes	85.40	3.32	FALSE
2	Bagging	84.54	3.61	FALSE
1	ZeroR	55.34	0.20	TRUE

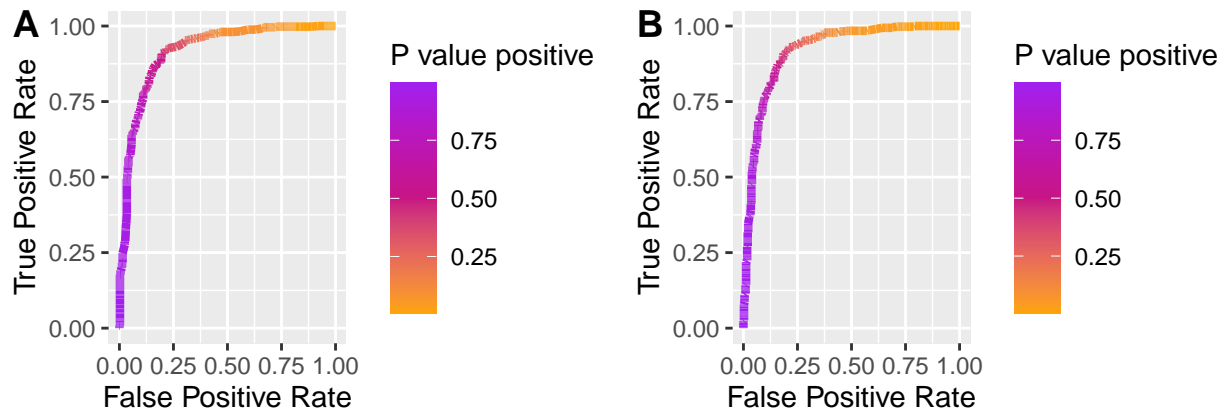
Adaboost has a lower processing

```
roc.smplog.stand <- read.arff("data/roc/roc_simplelog_stand.arff")
roc.smplog.cost <- read.arff("data/roc/roc_simplelog_0120.arff")

colourss <- hue_pal()(3)
stand <- ggplot(roc.smplog.stand) +
  geom_path(aes(x = `False Positive Rate`, y=`True Positive Rate`, color = Threshold), size=1.5) + scale_color_discrete()

cost <- ggplot(roc.smplog.cost) +
  geom_path(aes(x = `False Positive Rate`, y=`True Positive Rate`, color = Threshold), size=1.5) + scale_color_discrete()

ggarrange(stand, cost,
           ncol = 2, nrow = 2,
           labels = c("A", "B"))
```

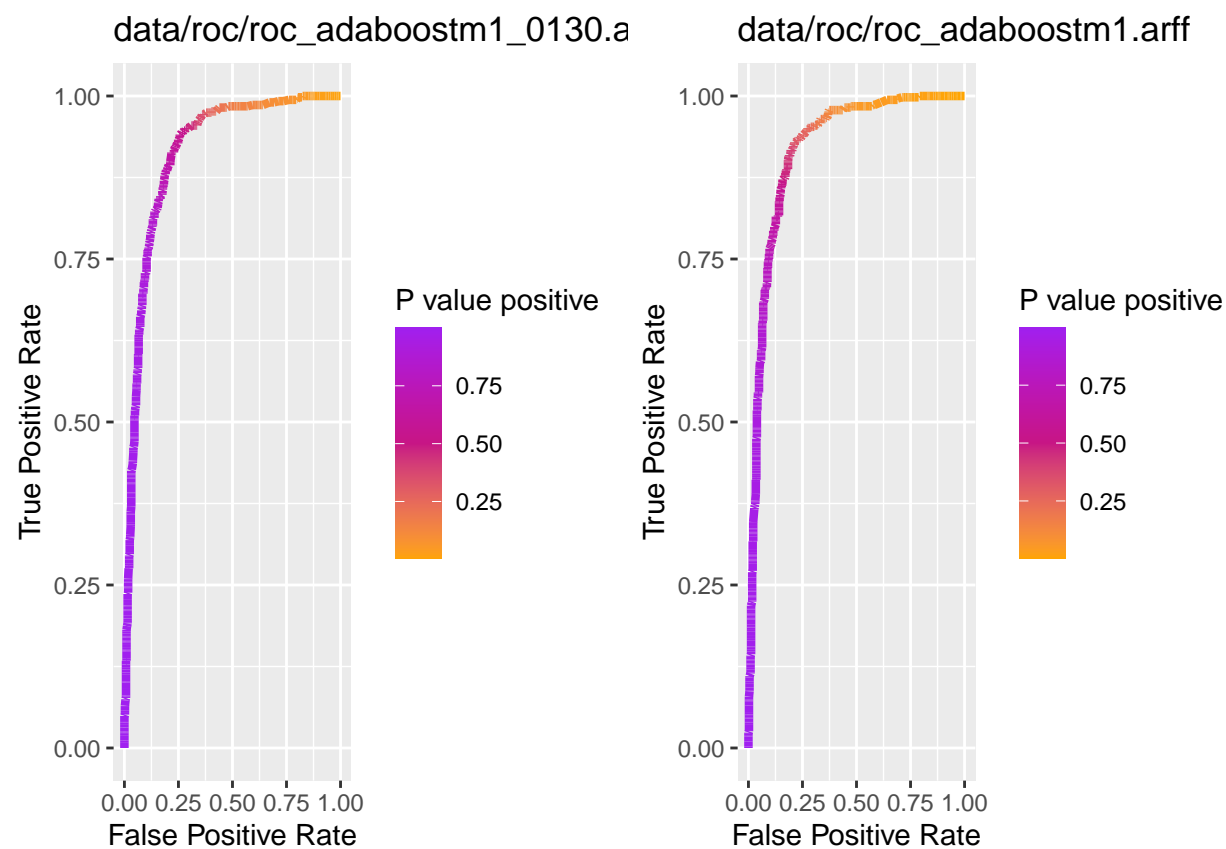


```
drawplots <- function(filepath) {
  files <- paste(filepath, list.files(filepath), sep = "/")
  plots = lapply(files, genplot)
  return(plots)
}

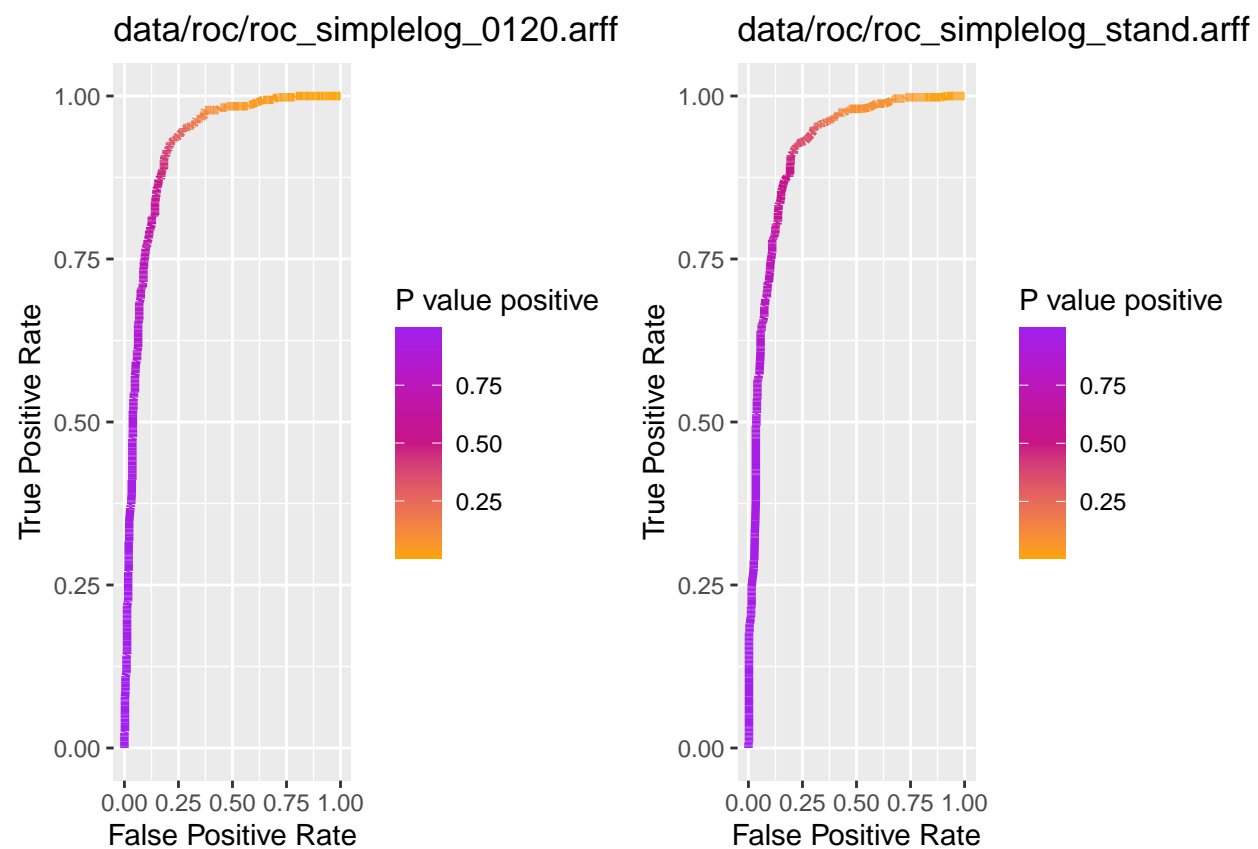
genplot <- function(x){
  roc.load = read.arff(x)
  pl <- ggplot(roc.load) +
    geom_path(aes(x = `False Positive Rate`, y=`True Positive Rate`, color = Threshold), size=1.5) + scale_x_continuous(limits=c(0,1))
  return(pl)
}

xs = drawplots("data/roc")
do.call(ggarrange, c(xs, ncol = 2))

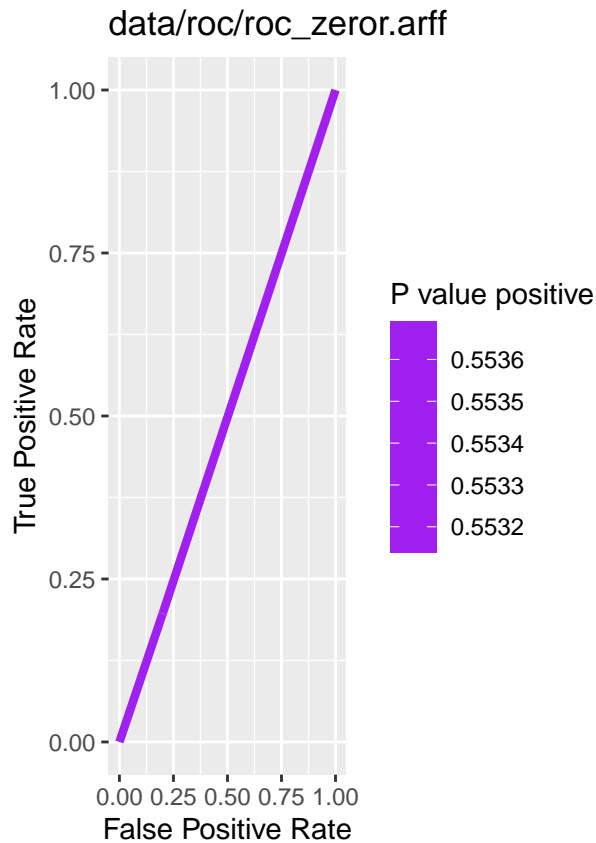
## $^1`
```



\$`2`



\$`3`



```
##
## attr("class")
## [1] "list"      "ggarrange"
```

Area under curve is calculated but will not be used as it is not a good metric as the ratio between FP and FN intentionally not the same. False negatives weigh more as false negatives are patients who will not get any further diagnoses as they are deemed not having heart disease

Further experimenting teaches that using a cost matrix of $[0 \ 1 \ 3 \ 0]$ decreases the amount of false negatives effectively. Using a voting algorithm in combination with a cost sensitive classifier performs worse than a cost sensitive classifier in with adaboost m1, using the previously mentioned cost matrix. The resulting algorithm has an accuracy of 84.5% with the following confusion matrix: a b <- classified as

```
295 115 | a = FALSE
27 481 | b = TRUE
```

Using the same algorithm using the cost matrix of $[0 \ 1 \ 1 \ 0]$ will result in an accuracy of 85.7 and the following confusion matrix: a b <- classified as

```
345 65 | a = FALSE
66 442 | b = TRUE
```

Increasing the FN costs to 3 will result in approximately halving the amount of false negative results doubling the amount of false positives while reducing the overall accuracy by roughly 1.2%. Increasing the costs of false negatives any further will create a lot more false positives which will increase monetary costs while false negatives will not be reduced in the same rate as the increase of false positives.

The precise setting is no exact science so the resulting algorithm used will be a cost sensitive adaboostm1 model with cost matrix set to $[0 \ 1 \ 3 \ 0]$. Weka Scheme: **weka.classifiers.meta.CostSensitiveClassifier -cost-matrix "[0.0 1.0; 3.0 0.0]" -S 1 -W weka.classifiers.meta.AdaBoostM1 -P 100 -S 1 -I 10 -W weka.classifiers.trees.DecisionStump** This will result in an overall accuracy of 84.5%

Since Adaboost m1 is used the data needs to be checked for large outliers as Adaboost tries to correct outliers and tries to prevent missclassification. Reducing the amount of attributes does not increase model accuracy.