Membrane composition prediction using machine learning, Logbook

Kasper Notebomer

9/14/2021

Contents

Setup of the libraries.

```
# Libraries
library("ggplot2")
```

```
## Registered S3 methods overwritten by 'tibble':
##   method     from
##   format.tbl pillar
##   print.tbl  pillar

## Warning: replacing previous import 'vctrs::data_frame' by 'tibble::data_frame'
## when loading 'dplyr'
```

```
library("kableExtra")
library("factoextra")
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library("cowplot")
library("gridExtra")
library("dplyr")
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:gridExtra':
##
##     combine

## The following object is masked from 'package:kableExtra':
##
##     group_rows

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library("scales")
library(ggpubr)
```

```
##
## Attaching package: 'ggpubr'

## The following object is masked from 'package:cowplot':
##
##     get_legend
```

```
library(ggfortify)
library(tidyr)
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:scales':
##
##     discard
```

```
library(ggcorrplot)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(e1071)
library(Hmisc)
```

```
## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

##
## Attaching package: 'Hmisc'

## The following object is masked from 'package:e1071':
##
##     impute

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following objects are masked from 'package:base':
##
##     format.pval, units
```

```
library(forcats)
```

1 Introduction

The data was provided by Tsjerk Wassenaar of the RuG. The data set contains data on membrane composition and characteristics. The data set is not publicly available. The data set also does not have a publication linked to it.

Seeing as there is no paper linked to the data set I can only guess using the information that I did get that the data was gathered by making a membrane using specific variables and the measuring the resulting membranes to get variables like the thickness and compressibility. The data set contains 14 variables and 2843 different measurement. The goal is to use measurement to predict the composition of the membrane. Typically you would use independent variables to predict one dependent variable. In this data set this doesn't seem to be the case, seeing as the variables that are given as parameters are seen as class variables, this means that these would be considered the labels. The parameters are: Temperature, Sterol type, Sterol concentration, Other (phospho)lipids in membrane, Aliphatic tails, Saturation index, Phosphatidyl choline concentration and Ethanol concentration. So the biggest question that the EDA should answer is which of these variables is most interesting to use as the label, or could we even predict multiple of them using the given variables.

2 Data exploration

2.1 Data reading & codebook

To start off the data is read in using read.csv(). The data frame that is created by this step is turned into a tibble and then used to create a codebook. The codebook is a csv file that contains the abbreviation, type and the description of every variable in the data set.

```
# Read the data and transform it to a tibble
data <- read.csv("data/dataFrame_all_sims.csv",
                 sep = ",", header=TRUE,
                 stringsAsFactors=FALSE)
data <- as_tibble(data)
```

Here, the codebook is made of all of the abbreviations in the data frame, their meaning and their type.

```
# Create the codebook
codebook <- data.frame("Abbreviation" = colnames(data), "Type" = sapply(data, typeof),
                       "Class" = sapply(data, class),
                       "Desciption" = c("Temperature (Kelvin)",
                                        "Sterol type",
                                        "Sterol concentration (%)",
                                        "Other (phospho)lipids in membrane (headgroup)",
                                        "Aliphatic tails",
                                        "Saturation index (double bonds per tail)",
                                        paste("Phosphatidyl choline concentration",
                                              "(% of non-sterol lipids)"),
                                        "Ethanol concentration (% of solvent)",
                                        "Area per lipid (nm^2)",
                                        "Thickness (nm)", "Bending rigidity (kB T)",
                                        "Tilt angle (degrees)",
                                        "Z-order", "Compressibility (cN / m)"),
                       "Units" = c("Kelvin",
                                   "",
```

```
                                    "%",
                                    "",
                                    "",
                                    "",
                                    "",
                                    "",
                                    "nm^2",
                                    "nm",
                                    "kB T",
                                    "degrees",
                                    "",
                                    "cN / m"))
# Turn dataframe into a tibble
codebook <- as_tibble(codebook)
kable(codebook, caption = "Codebook") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 1: Codebook

Abbreviation	Type	Class	Desciption	Units
temperature	integer	integer	Temperature (Kelvin)	Kelvin
sterol.type	character	character	Sterol type	
sterol.conc	integer	integer	Sterol concentration (%)	%
other.phosph	character	character	Other (phospho)lipids in membrane (headgroup)	
tails	character	character	Aliphatic tails	
satur.index	double	numeric	Saturation index (double bonds per tail)	
PC.conc	integer	integer	Phosphatidyl choline concentration (% of non-sterol lipids)	
ethanol.conc	integer	integer	Ethanol concentration (% of solvent)	
APL	double	numeric	Area per lipid (nm^2)	nm^2
thickness	double	numeric	Thickness (nm)	nm
bending	double	numeric	Bending rigidity (kB T)	kB T
tilt	double	numeric	Tilt angle (degrees)	degrees
zorder	double	numeric	Z-order	
compress	double	numeric	Compressibility (cN / m)	cN / m

```
# Write codebook to .csv file
write.csv(codebook, "data/codebook.csv", row.names = FALSE)
```

The codebook, shown in table 1, seems to have all of the correct data and datatypes. To easily access the description based on the abbreviation the following function is used.

```
# Get the description by the abbreviation from the codebook
get_des_by_ab <- function(df, abbreviation){
  description <- df[df$Abbreviation == abbreviation,]$Desciption
  return(description)
}
```

2.2 Structure & summary

Here I'll take a look at the structure and the five number summary of the data to look for any problems or discrepancies.

```r
# Print the structure of the data
data.frame(variable = names(data),
           classe = sapply(data, class),
           first_values = sapply(data, function(x) paste0(head(x),  collapse = ", ")),
           row.names = NULL) %>%
  kable(caption="Data structure")  %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 2: Data structure

variable	classe	first_values
temperature	integer	298, 298, 298, 298, 298, 298
sterol.type	character	chole, chole, chole, chole, chole, chole
sterol.conc	integer	20, 20, 20, 20, 20, 20
other.phosph	character	PE, PE, PE, PE, PE, PE
tails	character	PI, PI, PI, PI, PI, PI
satur.index	numeric	1, 1, 1, 1, 1, 1
PC.conc	integer	33, 50, 25, 100, 0, 75
ethanol.conc	integer	20, 20, 20, 20, 20, 20
APL	numeric	0.736911, 0.746226, 0.731945, 0.76963, 0.719509, 0.760091
thickness	numeric	3.62897268631964, 3.59699140829278, 3.6526367497855, 3.49605162681907, 3.71001820243157, 3.53890823219068
bending	numeric	10.0802, 10.1403, 10.2547, 10.0232, 10.4654, 9.86163
tilt	numeric	14.7578, 14.3075, 14.9777, 13.2029, 15.753, 13.7148
zorder	numeric	0.175499, 0.172084, 0.178494, 0.163349, 0.182191, 0.166807
compress	numeric	24.1821842679417, 24.3249489613324, 23.766069963793, 22.9607685634682, 23.1050322963544, 23.8604998172132

Looking at the structure of the data, shown in table 2, it seems like all of the columns have been read and have the right datatype.

```r
# Get the five number summary of the selected columns
sum <- summary(data[c("temperature","sterol.conc","satur.index",
                      "PC.conc","ethanol.conc","APL", "thickness", "bending",
                      "tilt", "zorder","compress")])

sum <- sub(".*:", "", sum)
sum[is.na(sum)] <- 0
rownames(sum) <- c("Minimum", "Q1", "Median", "Mean", "Q3", "Maximum", "Number of NA's")

# Print five number summary using kable
kable(sum, caption="Five number summary") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 3: (#tab:Five number summary)Five number summary

	temperature	sterol.conc	satur.index	PC.conc	ethanol.conc	APL	thickness	bending	tilt	zorder	compress
Minimum	298.0	0.00	0.0000	0.00	0	0.4514	3.003	0.2025	6.774	0.02736	1.286
Q1	298.0	10.00	0.0000	25.00	5	0.6350	3.627	9.8579	14.490	0.18604	23.097
Median	298.0	20.00	0.5000	50.00	15	0.6885	3.812	12.1000	18.157	0.25860	31.346
Mean	305.2	16.55	0.6208	50.02	15	0.6807	3.934	22.0256	35.732	0.33269	48.714
Q3	298.0	30.00	1.0000	75.00	25	0.7394	4.238	25.5712	35.933	0.45559	43.348
Maximum	328.0	30.00	2.0000	100.00	30	0.8924	5.101	125.3820	227.813	0.93013	538.446
Number of NA's	0	0	0	0	0	104	103	104	104	153	103

Looking at the structure there doesn't seem to be immediate problem with the data, like the data being read as the wrong datatype. When looking at the five number summary, depicted in table 3, there do seem to be some rows that are missing values like the APL, thickness, bending and tilt values. These rows will be omitted for plotting and other research seeing as they can't be used. The maximum of the bending rigidity

and tilt angle seem to be too high, this probably means that these are outliers seeing as the rest of the values in the summary seem to be believable. To check this we'll make a histogram and a boxplot of these variables.

```
# Remove NA's
row.has.na <- apply(data, 1, function(x){any(is.na(x))})
nona_data <- na.omit(data)
row.has.na.2 <- apply(nona_data, 1, function(x){any(is.na(x))})
paste("Number of NA's before: ", sum(row.has.na),
", Number of NA's after:", sum(row.has.na.2), sep="")
```

```
## [1] "Number of NA's before: 153, Number of NA's after:0"
```

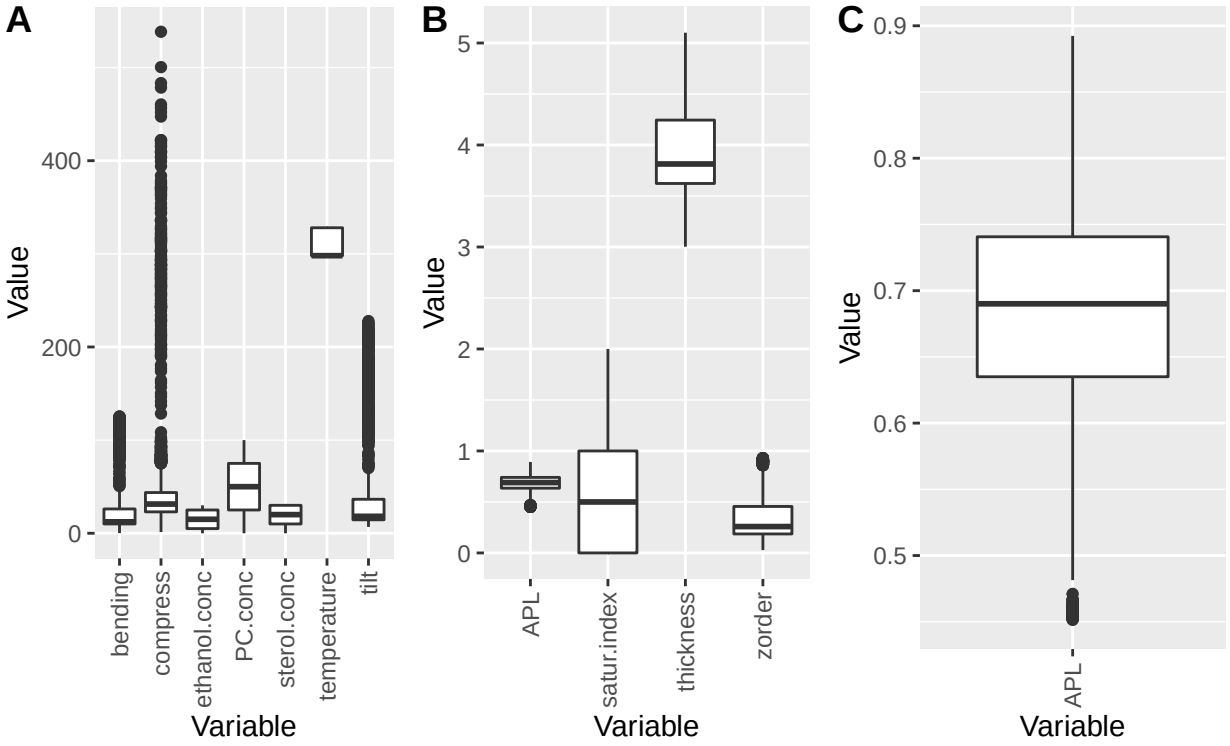
153 lines with NA's where omitted from the data set.

Here is a boxplor of all of the variables to check for any outliers.

```
# Make a boxplot of the numeric columns
plot1 <- nona_data %>%
  select("temperature","sterol.conc",
                       "PC.conc","ethanol.conc","bending",
                       "tilt","compress") %>%
  pivot_longer(., cols = c("temperature","sterol.conc",
                       "PC.conc","ethanol.conc","bending",
                       "tilt","compress"), names_to = "Variable", values_to = "Value") %>%
  ggplot(aes(x = Variable, y = Value)) +
  geom_boxplot() + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

plot2 <- nona_data %>%
  select("satur.index","APL","thickness","zorder") %>%
  pivot_longer(., cols = c("satur.index","APL","thickness","zorder"), names_to = "Variable", values_to =
  ggplot(aes(x = Variable, y = Value)) +
  geom_boxplot() + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

plot3 <- nona_data %>%
  select("APL") %>%
  pivot_longer(., cols = c("APL"), names_to = "Variable", values_to = "Value") %>%
  ggplot(aes(x = Variable, y = Value)) +
  geom_boxplot() + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

plots <- ggarrange(plot1, plot2, plot3,
                   labels = c("A", "B", "C"),
                   ncol = 3, nrow = 1)

title <- expression(atop(bold("Figure 0:"),
                         scriptstyle("Distribution of data per variable.")))
annotate_figure(plots,
                top=text_grob(title))
```

Figure 0:

Distribution of data per variable.

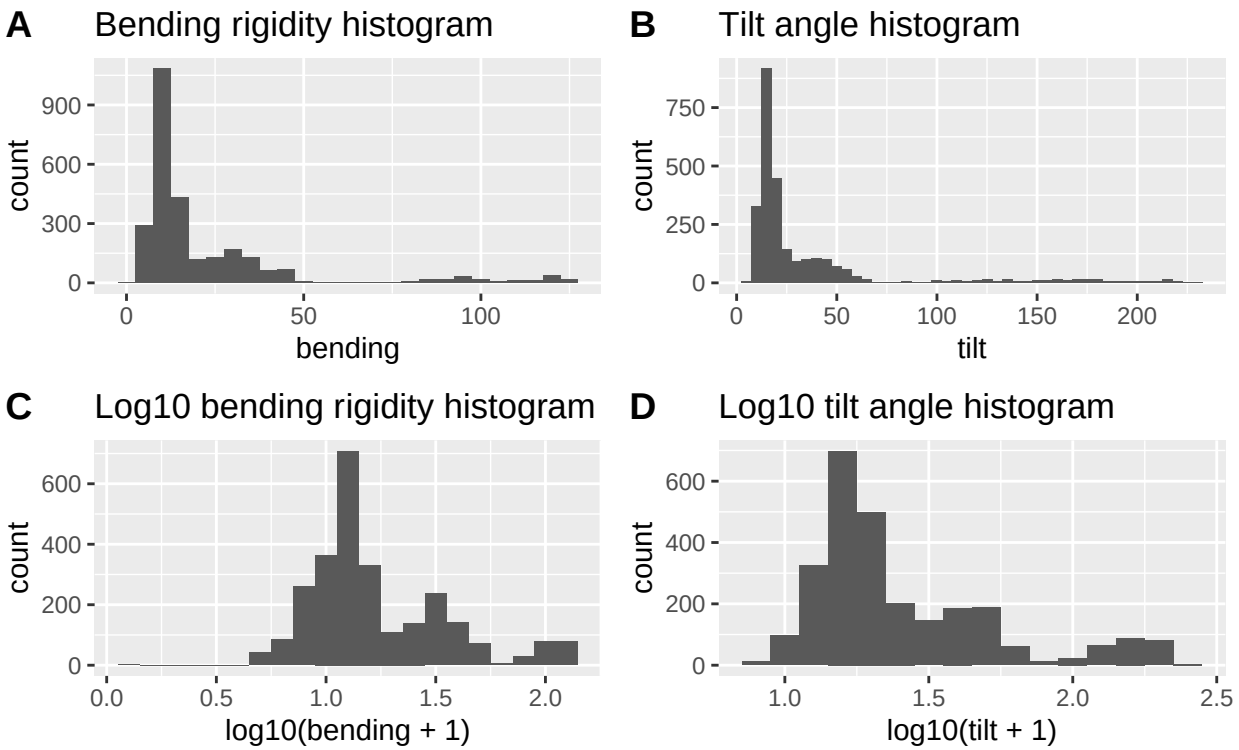


The boxplot clearly shows that there are indeed a lot of outliers to be found in the dataset, especially in the bending rigidity, compressibility and tilt ange columns. Outliers are shown as black dots in the plot. Some variables also have very clean distributions, this is mostly because some variables are numerical but only contain a few different levels because of the experiment that they where gathered from.

```
# Create 4 histograms
p <- ggplot(nona_data, aes(x= bending)) +
  geom_histogram(binwidth=5) +
  ggtitle("Bending rigidity histogram")
p2 <- ggplot(nona_data, aes(x=tilt)) +
  geom_histogram(binwidth=5) +
  ggtitle("Tilt angle histogram")
p3 <- ggplot(nona_data, aes(x= log10(bending + 1))) +
  geom_histogram(binwidth=0.1) +
  ggtitle("Log10 bending rigidity histogram")
p4 <- ggplot(nona_data, aes(x=log10(tilt + 1))) +
  geom_histogram(binwidth=0.1) +
  ggtitle("Log10 tilt angle histogram")
plots <- ggarrange(p, p2, p3, p4,
                   labels = c("A", "B", "C", "D"),
                   ncol = 2, nrow = 2)
title <- expression(atop(bold("Figure 1:"),
                         scriptstyle("Histograms of bending rifidity and tilt angle")))
annotate_figure(plots,
                top=text_grob(title))
```

Figure 1:

Histograms of bending rifidity and tilt angle



Both the bending and the tilt variable seem to have a lot of outliers when looking at figure 1. Even when transformed using log10 they still seem to be quite skewed. This is something we need to keep in mind in future research.

3 Exploring relations between variables

3.1 Density plot

I will now make density plots of some of the variables to see whether or not they look promising in finding class distinctions. These density plots will also make any skewing in the data apparent.

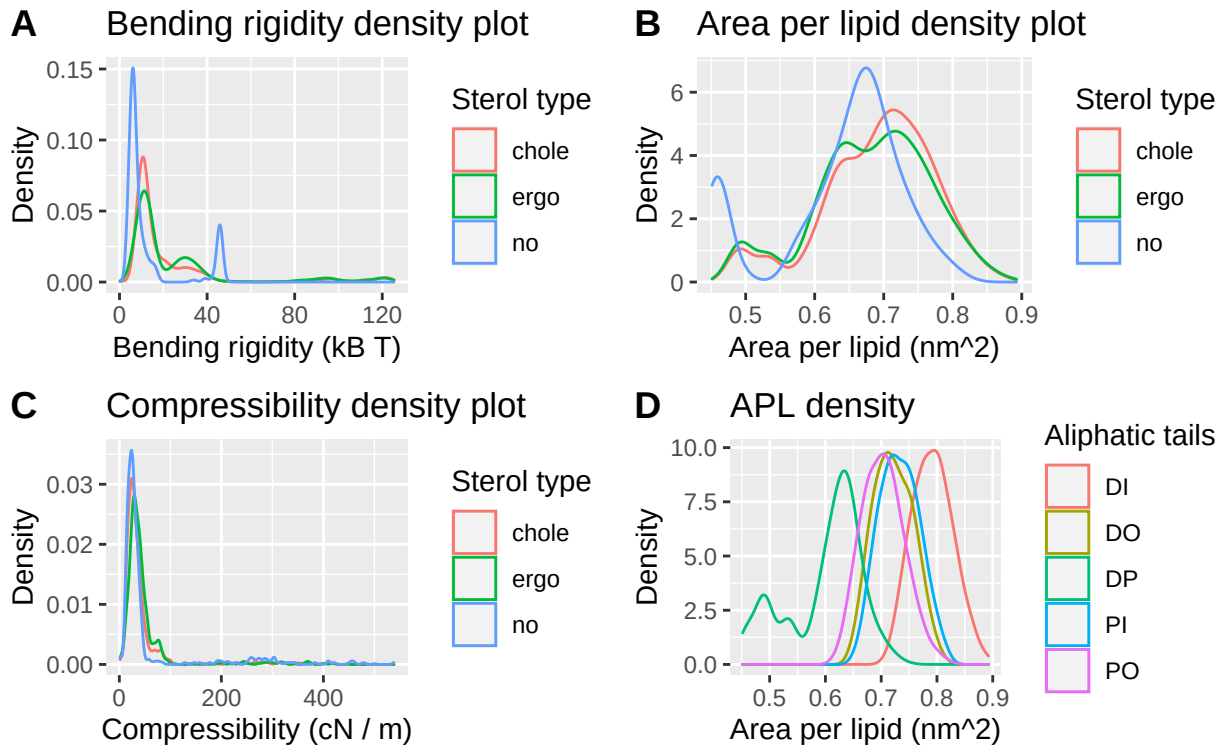
```
# Create density plots
p <- ggplot(nona_data, aes(x=bending, colour = factor(sterol.type))) +
  geom_density() +
  xlab(get_des_by_ab(codebook, "bending")) +
  ylab("Density") +
  ggtitle("Bending rigidity density plot") +
  labs(colour = get_des_by_ab(codebook, "sterol.type"))

p2 <- ggplot(nona_data, aes(x=APL, colour = factor(sterol.type))) +
  geom_density() +
  xlab(get_des_by_ab(codebook, "APL")) +
  ylab("Density") +
  ggtitle("Area per lipid density plot") +
  labs(colour = get_des_by_ab(codebook, "sterol.type"))
```

```
p3 <- ggplot(nona_data, aes(x=compress, colour = factor(sterol.type))) +
  geom_density() +
  xlab(get_des_by_ab(codebook, "compress")) +
  ylab("Density") +
  ggtitle("Compressibility density plot") +
  labs(colour = get_des_by_ab(codebook, "sterol.type"))

p4 <- ggplot(nona_data, aes(x=APL, colour = factor(tails))) +
  geom_density() +
  xlab(get_des_by_ab(codebook, "APL")) +
  ylab("Density") +
  ggtitle("APL density") +
  labs(colour = get_des_by_ab(codebook, "tails"))
```

```
# Plot density plots
plots <- ggarrange(p, p2, p3, p4,
                   labels = c("A", "B", "C", "D"),
                   ncol = 2, nrow = 2)
title <- expression(atop(bold("Figure 2:"),
                         scriptstyle(paste("Density plots of multiple",
                                           " variables coloured on sterol type"))))
annotate_figure(plots,
                top=text_grob(title))
```

Figure 2:

Density plots of multiple variables coloured on sterol type



Plot A in figure 2 show that there is basically no distinction between cholesterol and ergosterol based on the bending rigidity. However, distinction between no sterol or a sterol does seem to be possible based on

the bending rigidity seeing as the peaks of there classes only overlaps a small bit. There does seem to be an odd peak in the no sterol class at around 50 kB/t bending rigidity.

In plot B it looks like there is absolutely no way of distinguishing different classes based on the area per lipid, seeing as the peaks are basically in the same place. But just to be certain we'll still plot in against some other variables to make sure, seeing as it might be a very useful variable when paired with something like bending rigidity. The results depicted in plot C also don't look very promising seeing as every peak is around the same place again. Plot D on the other hand seems a lot more interesting seeing as all of the peaks are in slightly different locations, the DO, PO and the PI seem to overlap a lot but the DP and DI tails seem to have little overlap with the rest. It seems like APL could possibly be used as a variable to distinguish between aliphatic tails when paired with another variable.

When looking at all of the plots it looks like most of the data is skewed in one way or another, even though when we looked at the structure there didn't seem to be much. This can probably be explained by the extra dimension that was added in these plots, the sterol type and the aliphatic tails.

3.2 Scatter plot

```
# Code not run
# Results are too big and too clumpy for the analysis
pairs(nona_data,
      panel = "panel.smooth",
      pch = 20,
      cex = 0.35,
      col = rgb(0.1, 0.4, 1, alpha = 0.4))
```

This code is for a paired scatter plot. The eval is currently set to false seeing as there are a lot of variables that are compared to each other in this data set. Running this code will result in a big plot that isn't very well interpretable in a pdf file.

First we'll plot the bending area per lipid against the rigidity of the membrane, colouring the points based on the sterol type, to look for any kind of clustering or correlation. To make it even cleared we'll also make a second plot with a loess regression. The reason we use loess regression instead of linear regression is because loess will show us a better trend line, seeing as it isn't trying to fit a straight line like linear regression.

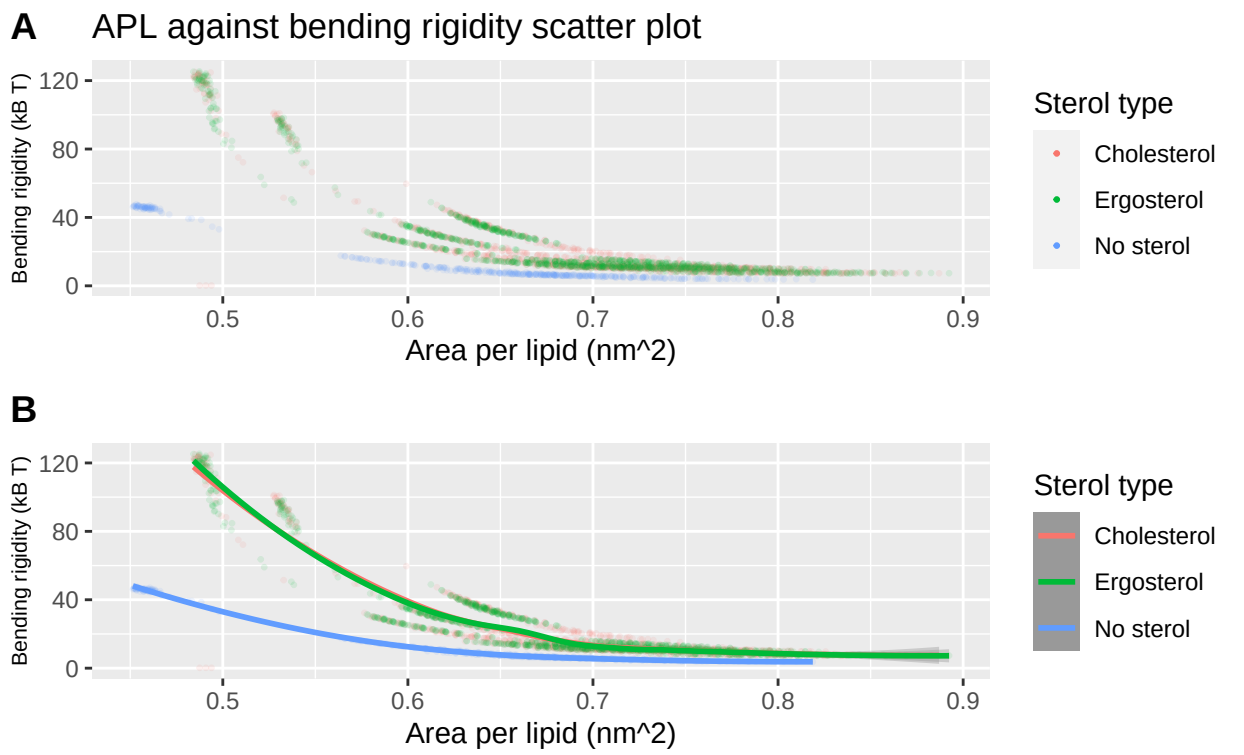
```
#Plot APL against bending rigidity
chart <- ggplot(nona_data, aes(APL, bending, colour = factor(sterol.type) )) +
  geom_point(alpha=1/10, size=0.5) +
  xlab(get_des_by_ab(codebook, "APL")) +
  ylab(get_des_by_ab(codebook, "bending")) +
  theme(axis.title.y = element_text(size=8)) +
  ggtitle("APL against bending rigidity scatter plot") +
  scale_color_manual(name=get_des_by_ab(codebook, "sterol.type"),
                     labels = c("Cholesterol",
                                "Ergosterol",
                                "No sterol"),
                     values = hue_pal()(3)) +
  guides(colour = guide_legend(override.aes = list(alpha = 1)))

plots <- ggarrange(chart,
                   chart +
                     geom_smooth(method = "loess") + ggtitle(""),
                   labels = c("A", "B"),
                   ncol = 1, nrow = 2)
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
title <- expression(atop(bold("Figure 3:"),
                         scriptstyle(paste("APL against bending rigidity,",
                                           " with and without loess regression"))))
# Give the plot a number
annotate_figure(plots,
                top=text_grob(title))
```

Figure 3:

APL against bending rigidity, with and without loess regression



Looking at the plots in figure 3 we can see that there is a clear separation in the area per lipid to bending rigidity ratio when comparing no sterol present to both cholesterol and ergosterol. When comparing cholesterol and ergosterol we can't see such a clear separation, there seems to be a lot of overlap. There are also a few outliers that are barely visible at area per lipid 0.4 and bending rigidity 0. The correlation seem to inverse logarithmic, so to check this I'll make a plot where the bending rigidity is log transformed.

```
# Create plot
chart <- ggplot(nona_data, aes(APL, log10(bending + 1), colour = factor(sterol.type) )) +
  geom_point(alpha=1/10, size=0.5) +
  xlab(get_des_by_ab(codebook, "APL")) +
  ylab(paste( "Log10 ", get_des_by_ab(codebook, "bending"))) +
  geom_smooth(method = "loess") +
  ggtitle("APL against log10 bending rigidity scatter plot") +
  scale_color_manual(name=get_des_by_ab(codebook, "sterol.type"),
                     labels = c("Cholesterol",
                                "Ergosterol",
                                "No sterol"),
```

```
                    values = hue_pal()(3))

# Plot the Plot and add a tag
chart + labs(tag = "Figure 4: APL against log transformed bending regidity") +
      theme(plot.title = element_text(hjust = 0.5),
          plot.margin = margin(t = 10, r = 10, b = 40, l = 10),
          plot.tag.position = c(0.4, -0.1)
          )
```

```
## `geom_smooth()` using formula 'y ~ x'
```

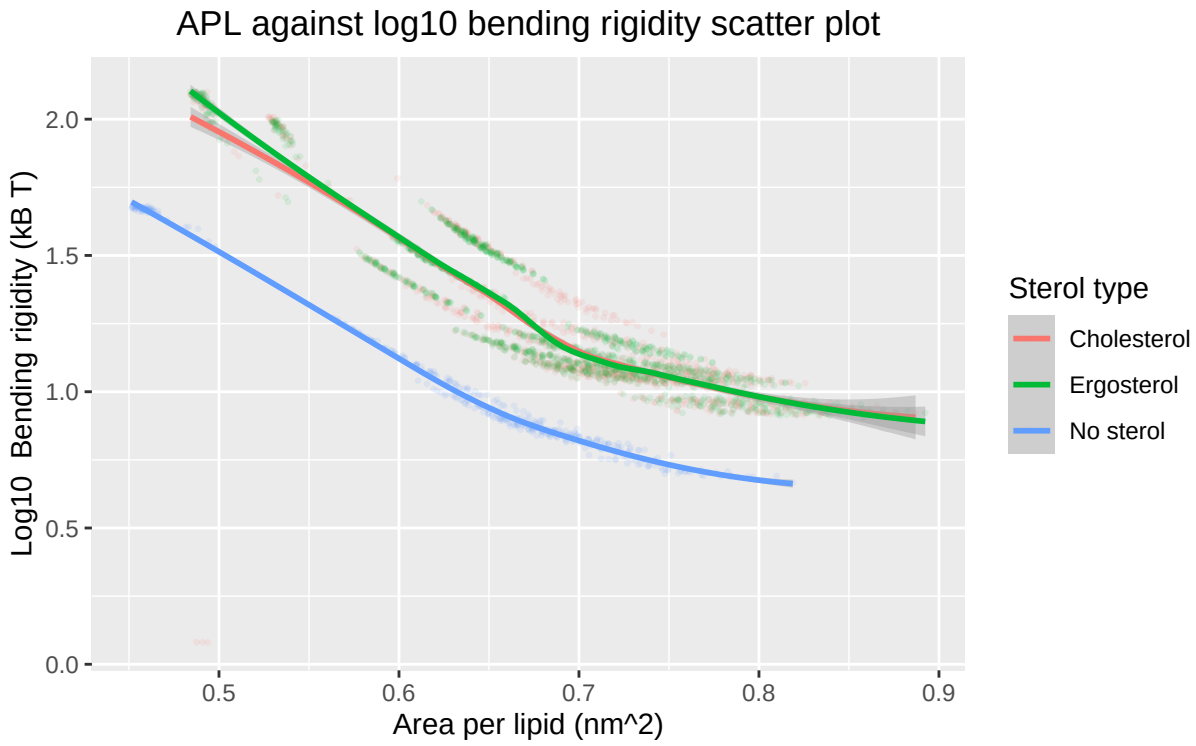


Figure 4: APL against log transformed bending regidity

Looking at figure 4 we can see that there indeed seem to be an almost linear correlation between the log10 transformed bending rigidity and the area per lipid. There does still seem to be a sleight curve in all of the trend lines when the area per lipid increases.

Here I'll plot the temperature against the bending rigidity while colouring based on saturation index, and deciding the shape based on the sterol type.

```
# Create rectangles to fit over the plot
rects <- data.frame(xstart = seq(0,600,300), xend = seq(300,900,300), col = letters[1:3])

# Create the plot
chart <- ggplot(nona_data, aes(temperature, bending, colour = factor(satur.index),
                               shape = factor(sterol.type) )) +
  geom_jitter(alpha=1/10) +
  xlab(get_des_by_ab(codebook, "temperature")) +
```

```
  ylab(get_des_by_ab(codebook, "bending"))  +
  ggtitle("Temperature against bending regidity,
          coloured by saturation index, shaped on sterol type") +
  theme(plot.title = element_text(size=10)) +
  labs(colour = get_des_by_ab(codebook, "sterol.type"),
       shape = get_des_by_ab(codebook, "satur.index"))

# Print the plot and add 2 boxes for the 2 groups in the temperature variable
chart +
  geom_rect(aes(xmin = -Inf, xmax = 312.9, ymin = -Inf, ymax = Inf),
            color="orange",
            fill = "green", alpha = 0.0001) +
  geom_rect(aes(xmin = 313.1, xmax = Inf, ymin = -Inf, ymax = Inf),
            color="purple",
            fill = "yellow", alpha = 0.0001) +
  labs(tag = "Figure 5: Temperature against bending regidity") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.margin = margin(t = 10, r = 10, b = 40, l = 10),
        plot.tag.position = c(0.35, -0.1)
        )
```

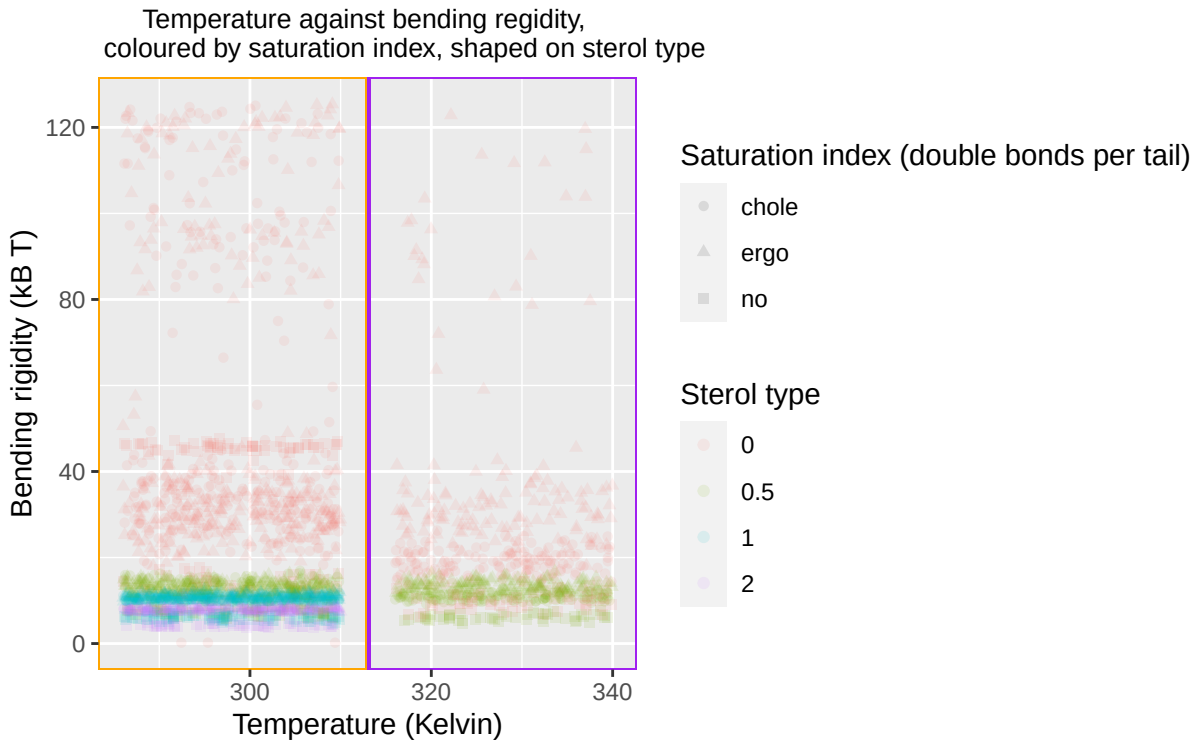


Figure 5: Temperature against bending regidity

When looking at the plot in figure 5, there seem to be a clear separation between a saturation index of 0 while the rest of the values seem to be clustered quite close together. There does seem to be a small separation between 0.5, 1 and 2 values. An interesting observation is that there don't seem to be any data point with a saturation index of more than 0.5 in the 328 kelvin group (the purple box).

Here the APL will be plot against the bending rigidity again, but here the colour is decided based on the

saturation index and the shape on the sterol type.

```
# Create plots
chart2 <- ggplot(nona_data, aes(APL, bending, colour = factor(satur.index), shape = factor(sterol.type)
  geom_jitter(alpha=0.1, size=0.5) +
  xlab(get_des_by_ab(codebook, "APL")) +
  ylab(get_des_by_ab(codebook, "bending"))  +
  ggtitle("APL against bending regidity,
          coloured on saturation index, shaped on sterol type") +
  theme(plot.title = element_text(size=10)) +
  guides(colour = guide_legend(override.aes = list(alpha = 1)))
chart2 + labs(colour = get_des_by_ab(codebook, "satur.index"), shape = get_des_by_ab(codebook, "sterol.
  labs(tag = "Figure 6: APL against bending rigidity coloured on saturation index") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.margin = margin(t = 10, r = 10, b = 40, l = 10),
        plot.tag.position = c(0.4, -0.1)
        )
```

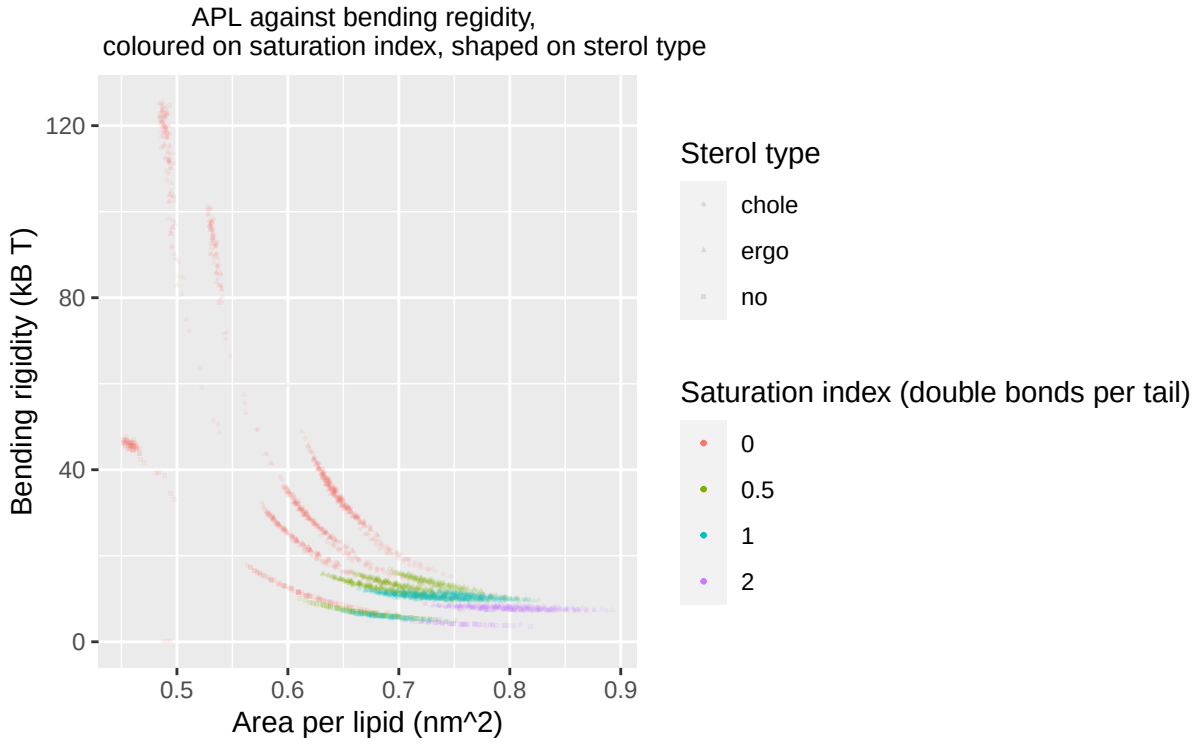


Figure 6: APL against bending rigidity coloured on saturation index

In figure 6 we can see the same kind of correlation as before, but now we can see that there also seem to be some kind of clustering based on the saturation index.

Here is a plot that shows the compressibility plotted against the bending rigidity.

```
# Create plot
chart <- ggplot(nona_data, aes(compress, bending, colour = factor(sterol.type) )) +
  geom_point(alpha=2/10, size=0.5) +
  xlab(get_des_by_ab(codebook, "compress")) +
```

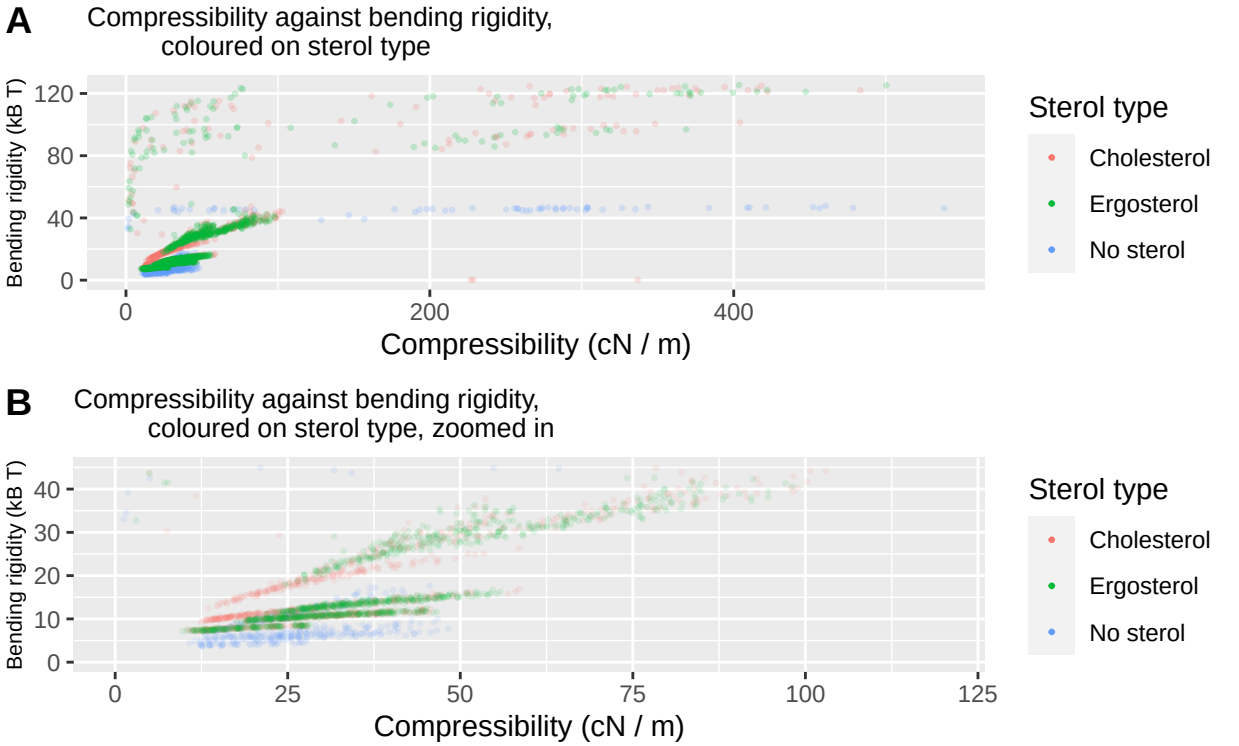
```
  ylab(get_des_by_ab(codebook, "bending")) +
  theme(axis.title.y = element_text(size=8)) +
  ggtitle("Compressibility against bending rigidity,
          coloured on sterol type") +
  scale_color_manual(name=get_des_by_ab(codebook, "sterol.type"),
                     labels = c("Cholesterol",
                                "Ergosterol",
                                "No sterol"),
                     values = hue_pal()(3)) +
  theme(plot.title = element_text(size=10)) +
  guides(colour = guide_legend(override.aes = list(alpha = 1)))

chart2 <- ggplot(nona_data, aes(compress, bending, colour = factor(sterol.type) )) +
  geom_point(alpha=1/10, size=0.5) +
  xlab(get_des_by_ab(codebook, "compress")) +
  ylab(get_des_by_ab(codebook, "bending")) +
  theme(axis.title.y = element_text(size=8)) +
  ggtitle("Compressibility against bending rigidity,
          coloured on sterol type, zoomed in") +
  scale_color_manual(name=get_des_by_ab(codebook, "sterol.type"),
                     labels = c("Cholesterol",
                                "Ergosterol",
                                "No sterol"),
                     values = hue_pal()(3)) +
  scale_x_continuous(limits=c(0,120)) +
  scale_y_continuous(limits=c(0, 45))  +
  theme(plot.title = element_text(size=10)) +
  guides(colour = guide_legend(override.aes = list(alpha = 1)))

# Arrange plots
plots <- ggarrange(chart,
                   chart2,
                   labels = c("A", "B"),
                   ncol = 1, nrow = 2)
```

```
## Warning: Removed 261 rows containing missing values (geom_point).
```

```
title <- expression(atop(bold("Figure 7:"),
                         scriptstyle("Compressibility against bending rigidity, full and zoomed")))
annotate_figure(plots,
                top=text_grob(title))
```

Figure 7:

Compressibility against bending rigidity, full and zoomed



In plot A of figure 7 we can't really make out much of a pattern in the dense clout to the left, but when looking at the rest of the data points there does seem to be some grouping based on whether or not no sterol or a sterol is present. When zooming in in plot B we can now see the same kind of grouping happen again based on either a sterol present or no sterol present.

3.3 Heatmap

Here a heatmap is made to look for correlation between variables.

```
# Get correlation
correlation <- nona_data[,9:14]
melted_cormat <- melt(cor(correlation))

ggplot(data = melted_cormat, aes(x=Var1, y=Var2, fill=value)) +
  geom_tile() +
  ggtitle("Heatmap of the variables") +
  labs(tag = "Figure 8: Heatmap of the independent varaibles") +
  theme(plot.title = element_text(hjust = 0.5),
        plot.margin = margin(t = 10, r = 10, b = 40, l = 10),
        plot.tag.position = c(0.5, -0.1)
        )
```

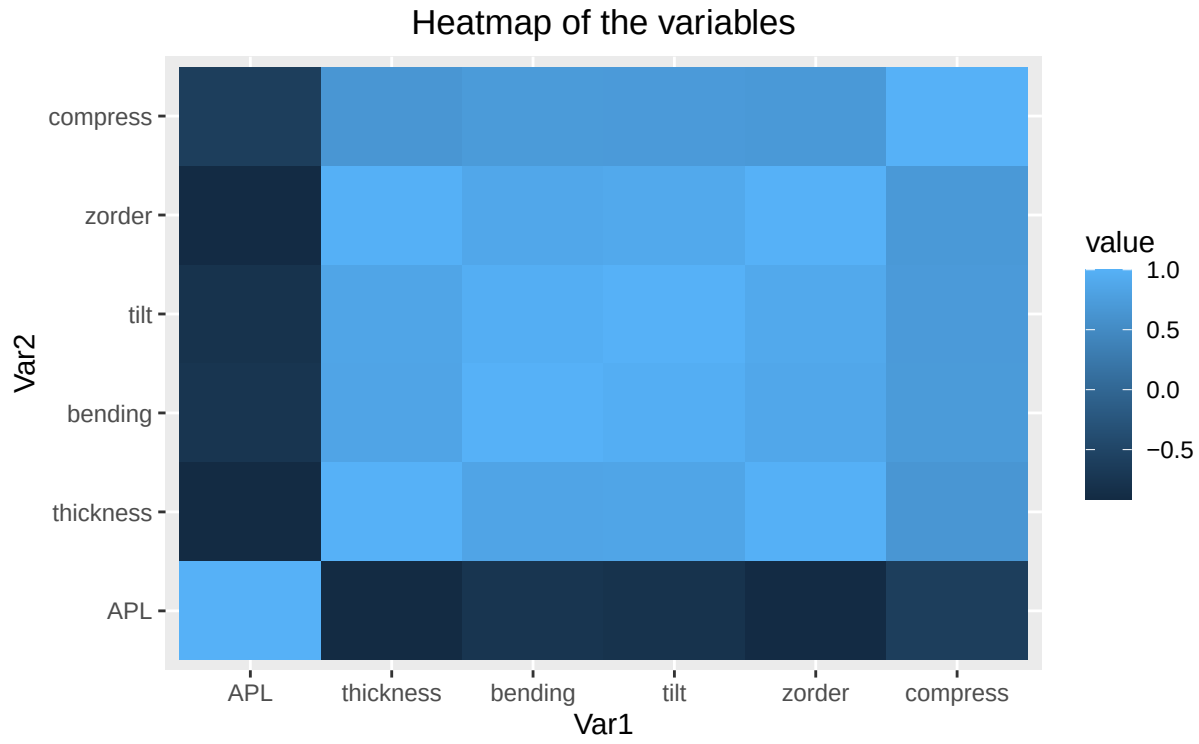


Figure 8: Heatmap of the independent varaibles

The results shown in figure 8 are very interesting. It seems like all of the variables seem to be quite heavily correlated, except for APL, which is negatively correlated. It also seem like the comprehensibility is slightly less correlated then the rest of the variables. But thickness, bending, tilt and z order all seem to be heavily correlated.

3.4 PCA

Here we'll make a PCA plot to look for possible clusters.

```
# PCA plot results
pca_res <- prcomp(correlation, scale. = TRUE, center = TRUE)
print(pca_res)
```

```
## Standard deviations (1, .., p=6):
## [1] 2.24944886 0.69482659 0.55396674 0.31774787 0.20132585 0.09391948
##
## Rotation (n x k) = (6 x 6):
##                  PC1         PC2        PC3          PC4         PC5
## APL       -0.4039636 -0.38324528 -0.4127355 -0.705249607  0.14566451
## thickness  0.4238589  0.31048244  0.2156260 -0.511891894 -0.01429709
## bending    0.4152014 -0.14463396 -0.5644389  0.026515944 -0.69807879
## tilt       0.4196997 -0.08623976 -0.5085343  0.252592319  0.68632375
## zorder     0.4334754  0.22183399  0.0793969 -0.419528906  0.14134816
## compress   0.3474421 -0.82410989  0.4468250 -0.008899588  0.01561576
##                   PC6
## APL        0.031475796
```

```
## thickness  0.644369420
## bending   -0.008970044
## tilt       0.151540515
## zorder    -0.748732960
## compress   0.012300032
```

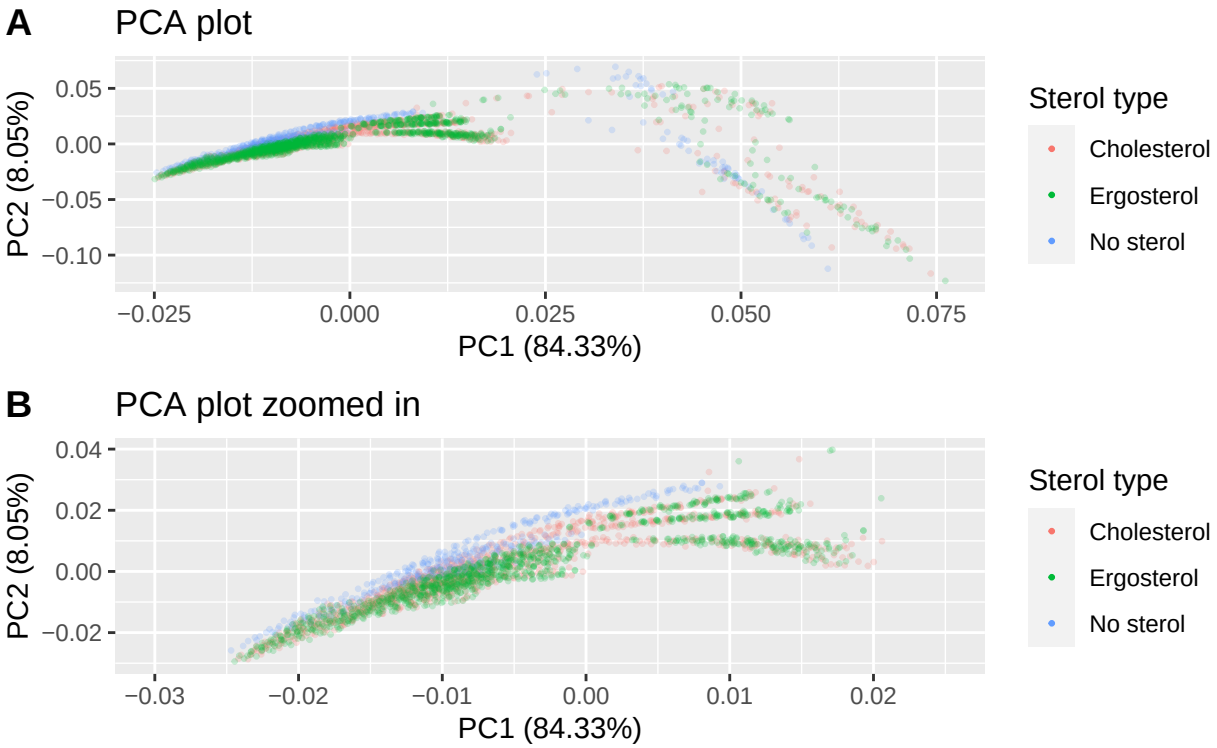
```
# Create PCA plot
plots <- ggarrange(autoplot(pca_res, data = nona_data,
                            colour = 'sterol.type', alpha=2/10,
                            size = 5/10) +
                     ggtitle("PCA plot") +
                     scale_color_manual(name=get_des_by_ab(codebook, "sterol.type"),
                                        labels = c("Cholesterol",
                                                   "Ergosterol",
                                                   "No sterol"),
                                        values = hue_pal()(3)) +
                     guides(colour = guide_legend(override.aes = list(alpha = 1))),
                   autoplot(pca_res, data = nona_data,
                            colour = 'sterol.type', alpha=1/5,
                            size = 1/2) +
                     ggtitle("PCA plot zoomed in") +
                     scale_color_manual(name=get_des_by_ab(codebook, "sterol.type"),
                                        labels = c("Cholesterol",
                                                   "Ergosterol",
                                                   "No sterol"),
                                        values = hue_pal()(3)) +
                     scale_x_continuous(limits=c(-0.03,0.025)) +
                     scale_y_continuous(limits=c(-0.03,0.04)) +
                     guides(colour = guide_legend(override.aes = list(alpha = 1))),
                   labels = c("A", "B"),
                   ncol = 1, nrow = 2)
```

```
## Warning: Removed 271 rows containing missing values (geom_point).
```

```
# Add title
title <- expression(atop(bold("Figure 9:"),
                         scriptstyle("PCA plot using autoplot")))
annotate_figure(plots,
                top=text_grob(title))
```

Figure 9:

PCA plot using autoplot



In plot A of figure 9 we can hardly see any clusters at all, it does seem to have a dense cloud to the right. Plot B cuts off the data point on the side and zooms in on the dense cloud of dots visible in plot A, a total of 271 dots aren't visible because of this. In this plot we can't really make out much of a pattern or cluster. It does, however, seem like no sterol cluster is sleightly seperated from the other sterol types.

Here is another way to make a PCA plot that is a bit more advanced seeing as it will also give us a way to identify correlation based on variables in the clustering.

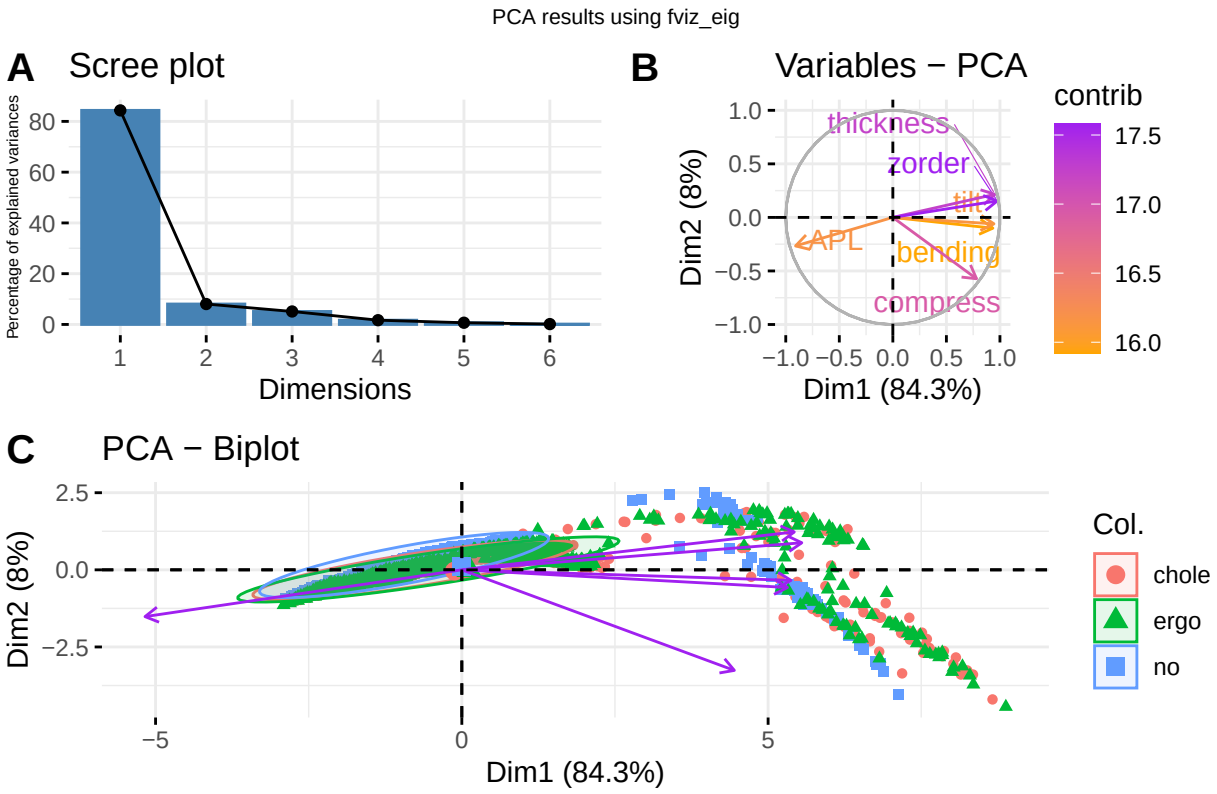
```
# Create PCA plot
res.pca <- prcomp(correlation, scale = TRUE)
plot1 <- fviz_eig(res.pca) +
  theme(axis.title.y = element_text(size=6))
plot2 <- fviz_pca_var(res.pca,
                      col.var = "contrib", # Colour by contributions to the PC
                      gradient.cols = c("orange", "purple"),
                      repel = TRUE     # Avoid text overlapping
                      )
plot3 <- fviz_pca_biplot(res.pca, repel = FALSE,
                         col.var = "purple", # Variables colour
                         col.ind = nona_data$sterol.type,  # Individuals colour
                         label = FALSE,
                         addEllipses =  TRUE,
                         ellipse.type = "t"
                         )
# Arrange plots
plot <- arrangeGrob(plot1,  plot2,
                    plot3,
```

```
                    ncol = 2, nrow = 2,
                    layout_matrix = rbind(c(1,2), c(3,3)))
```

```
## Warning in MASS::cov.trob(data[, vars]): Probable convergence failure
```

```
plot <- as_ggplot(plot) +
  draw_plot_label(label = c("A", "B", "C"), size = 15,
                  x = c(0, 0.5, 0), y = c(1, 1, 0.5))
title <- expression(atop(bold("Figure 10:"),
                         scriptstyle("PCA results using fviz_eig")))
annotate_figure(plot,
                top=text_grob(title))
```

Figure 10:



The PCA plots, B and C, shown in figure 10 confirms what we could already see in the heat map, that is that most of the variables seem to be heavily correlated, seeing as they are pointing in the same direction, except for APL and compress. We can also see that thickness and z order are correlated just like tilt and bending. The PCA plot also still looks the same, only now it has as ellipse for clusters that was calculated using a t test. The ellipses all seem to overlap so there doesn't seem to be much clustering.

3.5 K means

Here we'll use K-means clustering as a different technique to find possible clusters.

```r
# Calculate K-means, set seed for reproducibility
set.seed(666)
km.res <- kmeans(scale(correlation), 3, nstart = 25)

# Dimension reduction using PCA
res.pca <- prcomp(correlation,  scale = TRUE)
ind.coord <- as.data.frame(get_pca_ind(res.pca)$coord)
ind.coord$cluster <- factor(km.res$cluster)
ind.coord$sterol.type <- nona_data$sterol.type
kable(head(ind.coord), caption="kmeans results") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 4: kmeans results

Dim.1	Dim.2	Dim.3	Dim.4	Dim.5	Dim.6	cluster	sterol.type
-1.385930	-0.1910350	-0.1129858	0.0445117	0.0120431	0.0276672	3	chole
-1.468677	-0.2591053	-0.1694398	0.0073676	0.0181114	-0.0044196	3	chole
-1.331954	-0.1460367	-0.0861045	0.0548778	0.0031295	0.0500533	3	chole
-1.709264	-0.4213788	-0.3268945	-0.0624653	0.0414548	-0.1137487	3	chole
-1.205414	-0.0421895	-0.0158859	0.0909438	-0.0115402	0.1168805	3	chole
-1.610915	-0.3573446	-0.2555837	-0.0356559	0.0387487	-0.0660767	3	chole

The results in table 4 are printed to give a quick overview of the results of the K-means and to check for any possible problems. The row with sterol type is added so we can look whether or not this has anything to do with the found clusters.

```r
# Percentage of variance explained by dimensions
eigenvalue <- round(get_eigenvalue(res.pca), 1)
variance.percent <- eigenvalue$variance.percent
kable(head(eigenvalue), caption="Variance explained by dimensions") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 5: Variance explained by dimensions

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	5.1	84.3	84.3
Dim.2	0.5	8.0	92.4
Dim.3	0.3	5.1	97.5
Dim.4	0.1	1.7	99.2
Dim.5	0.0	0.7	99.9
Dim.6	0.0	0.1	100.0

In table 5 we can see the amount of variance is explained by ever dimension. Just like in the previous PCA plot we can see that over 80% of the variance is explained by a single variable, which is odd, to say the least.

```r
# Make pca plot based on K-means clusters
ggscatter(
  ind.coord, x = "Dim.1", y = "Dim.2",
  color = "cluster", palette = "npg", ellipse = TRUE, ellipse.type = "convex",
  shape = "sterol.type", size = 1.5,  legend = "right", ggtheme = theme_bw(),
  xlab = paste0("Dim 1 (", variance.percent[1], "% )" ),
  ylab = paste0("Dim 2 (", variance.percent[2], "% )" )
) +
  stat_mean(aes(color = cluster), size = 4) +
  labs(tag = "Figure 11: PCA plot based on K-means clustering",
       shape = get_des_by_ab(codebook, "sterol.type")) +
  theme(plot.title = element_text(hjust = 0.5),
        plot.margin = margin(t = 10, r = 10, b = 40, l = 10),
        plot.tag.position = c(0.46, -0.1)
        )
```

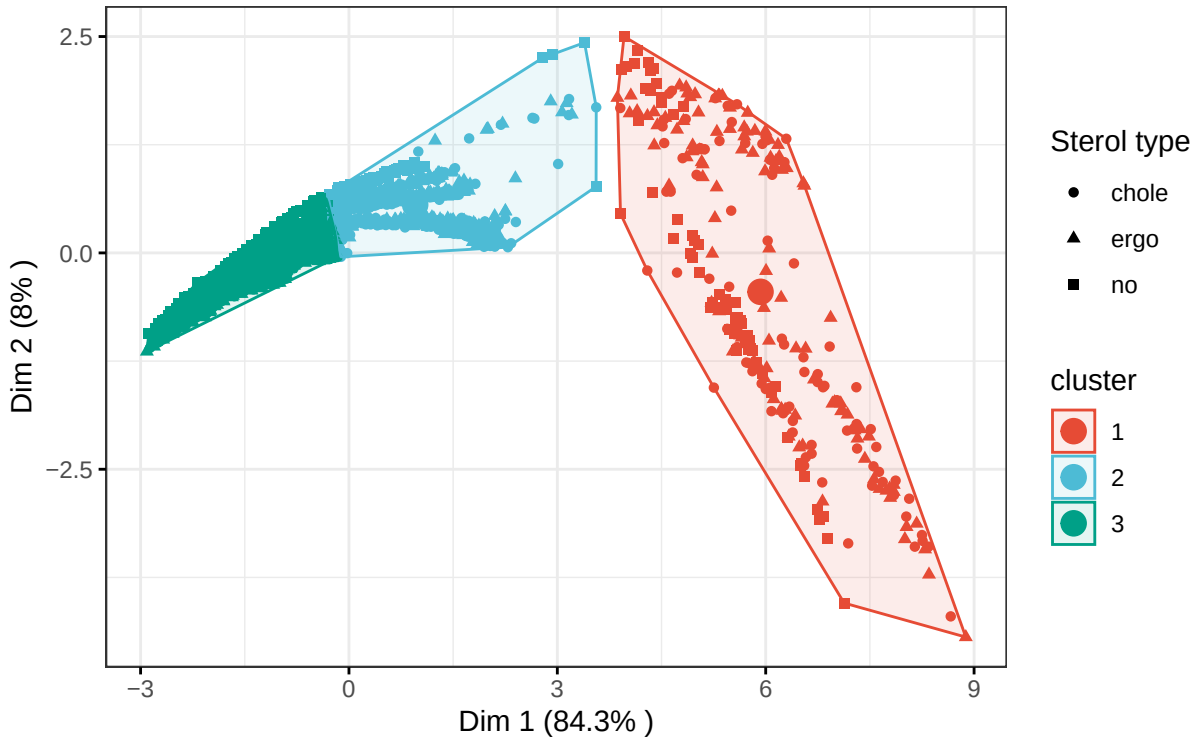


Figure 11: PCA plot based on K−means clustering

This plot shown in figure 11 still doesn't seem very promising seeing as there seem to be three clear clusters, but this seems to be more because of the large spread of the data. The clusters also seem to have nothing to do with the sterol type of the membrane.

4 Research question

Is it possible to use machine learning to reliably predicts the sterol type in a membrane with a higher than 80% accuracy, given the area per lipid, the bending rigidity and the compressibility?

5 Data Cleaning

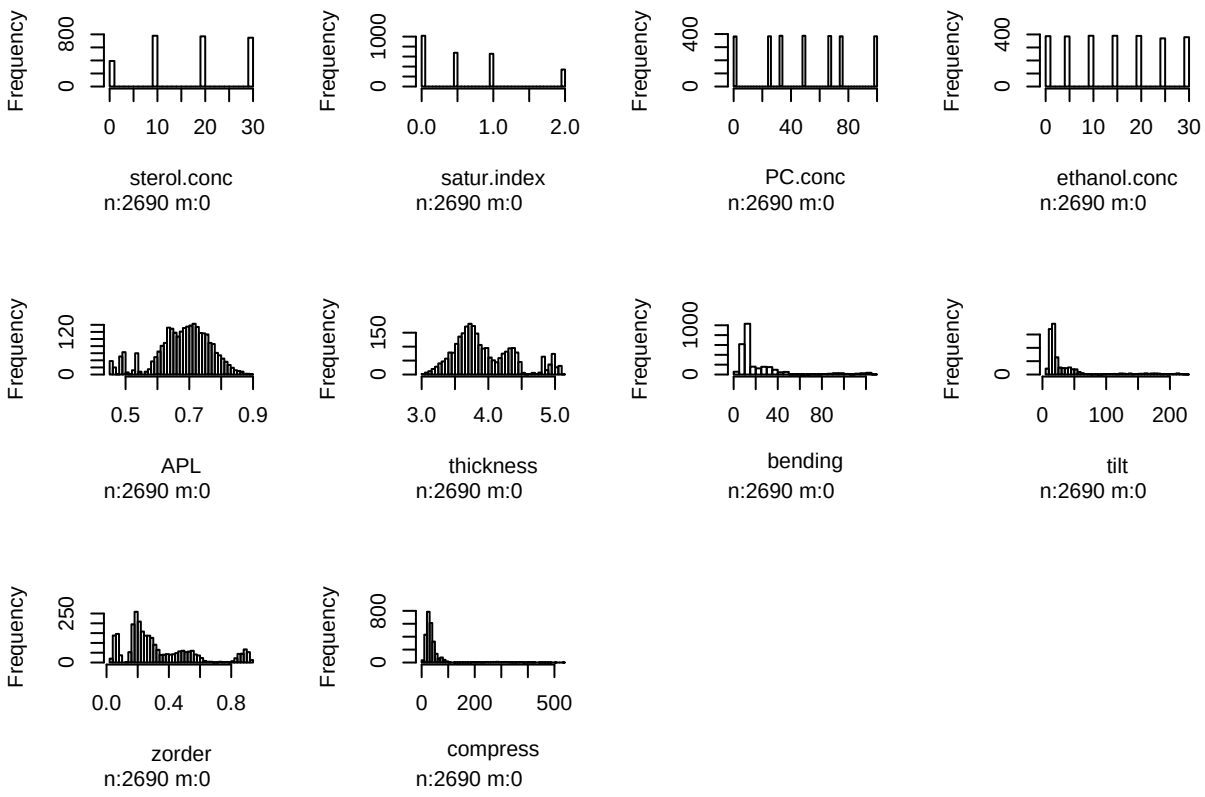
Here is a quick code snippet to show us the skewness of the data. This will be useful when deciding whether or not to use standardization or normalization on the columns.

```
apply(nona_data[c("temperature","sterol.conc","satur.index",
                      "PC.conc","ethanol.conc","APL", "thickness", "bending",
                      "tilt", "zorder","compress")], 2, skewness, na.rm =TRUE)
```

```
##   temperature   sterol.conc   satur.index       PC.conc  ethanol.conc
##  1.1257170406 -0.1719109112  0.9075759350  0.0008402866  0.0149496771
##           APL     thickness       bending          tilt        zorder
## -0.6069614189  0.7652531496  2.6743851360  2.6321330960  1.1463852300
##      compress
##  3.9327898578
```

A lot of the data seems to be extremely skewed. Lets plot it in histograms to get a better look.

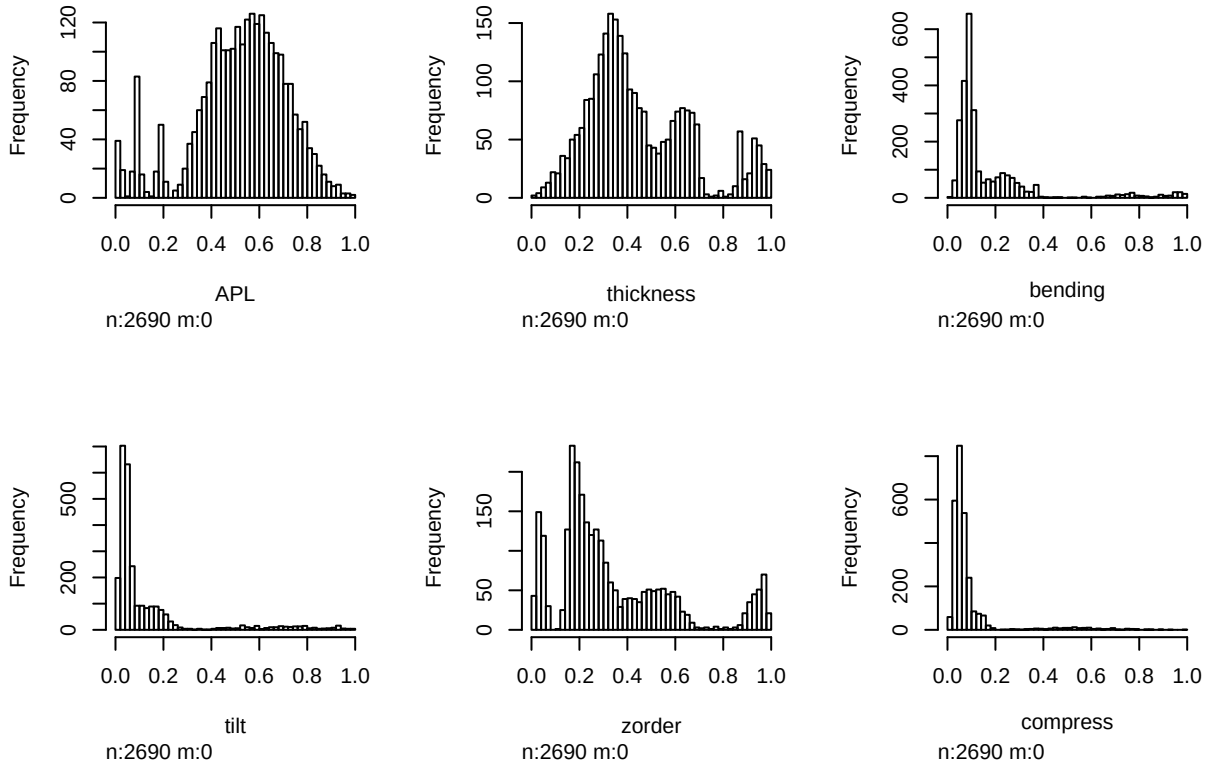
```
hist.data.frame(nona_data[c("temperature","sterol.conc","satur.index",
                            "PC.conc","ethanol.conc","APL", "thickness", "bending",
                            "tilt", "zorder","compress")])
```



The data seems to be skewed a lot in almost every column. If we were planning on using a machine learning algorithms that are based on the assumption that the data has a Gaussian distribution we would need to use normalization (transformation) to force the data in to such a distribution. To achieve this we could use a log

or cube root transformation. We probably won't be using any algorithms that assume that our distribution is Gaussian, which is why we won't make any transformations at first. Distributions like bending, tilt and compress have very long tails due to outliers. These could be log transformed. However they do not follow a power law distribution (they aren't fat tailed, the tail is just long), and values do not span across multiple orders of magnitude. This is why they won't be log transformed. Seeing as most of the distributions aren't Gaussian we should use min_max normalization(scaling). This does lessen/suppress the impact of outliers, it also isn't as effective with outliers. We could also use z score normalization (standardization), but this assumes that our distributions are Gaussian, which they are not. It could still be used. However, due to the heavy skew and the high amount of outliers in some column, the scale probably won't be balanced. We could also look at box cox transformation, but I wasn't taught this method, so I'm not confident about using is. The reason we use normalization is because we might want to use and algorithm like k nearest neighbor(KNN), which is an algorithm that is very sensitive to data scale.

```
scale_min_max <- function(x) (x - min(x)) / (max(x) - min(x))
nona_copy <- nona_data[c("APL", "thickness", "bending",
                                              "tilt", "zorder","compress")]

scaled_data <- as.data.frame(lapply(nona_copy, FUN=scale_min_max))
hist.data.frame(scaled_data)
```




```
# scaled_data <- as.data.frame(lapply(nona_copy, FUN=scale))
# hist.data.frame(scaled_data)

apply(scaled_data, 2, skewness, na.rm =TRUE)
```

```
##        APL  thickness    bending       tilt     zorder   compress
```

```
## -0.6069614  0.7652531  2.6743851  2.6321331  1.1463852  3.9327899
```

After min max scaling we can see that the distributions stay the same but scale on all of the variables now ranges from zero to one. In the future we could always try different forms of normalization and transformation to see whether or not our model improves. Now the only thing left to do is add the sterol type label.

```
scaled_data$sterol.type <- nona_data$sterol.type
kable(head(scaled_data), caption="Cleaned data") %>%
  kable_styling(latex_options = c("hold_position"), full_width = TRUE)
```

Table 6: Cleaned data

APL	thickness	bending	tilt	zorder	compress	sterol.type
0.6474103	0.2983812	0.0789085	0.0361181	0.1640910	0.0426240	chole
0.6685354	0.2831408	0.0793886	0.0340809	0.1603082	0.0428898	chole
0.6361482	0.3096582	0.0803025	0.0371130	0.1674086	0.0418494	chole
0.7216122	0.2350388	0.0784531	0.0290836	0.1506323	0.0403502	chole
0.6079452	0.3370028	0.0819857	0.0406205	0.1715037	0.0406188	chole
0.6999791	0.2554618	0.0771624	0.0313995	0.1544628	0.0420252	chole

```
# Write new data to csv
write.csv(scaled_data, "data/clean_data.csv", row.names = FALSE)
```

Only the sterol type was added as the label, seeing as the other variables where considered labels by the assignment, and we're interested in the sterol type in the membrane. Looking at the PCA plot and the heatmap it would seem like some of the variables are heavily correlated. However, it was decided not to remove any of the variables as of this time.

Here is a PCA plot that is made using the data after applying log and cube transformations to reduce skewness.

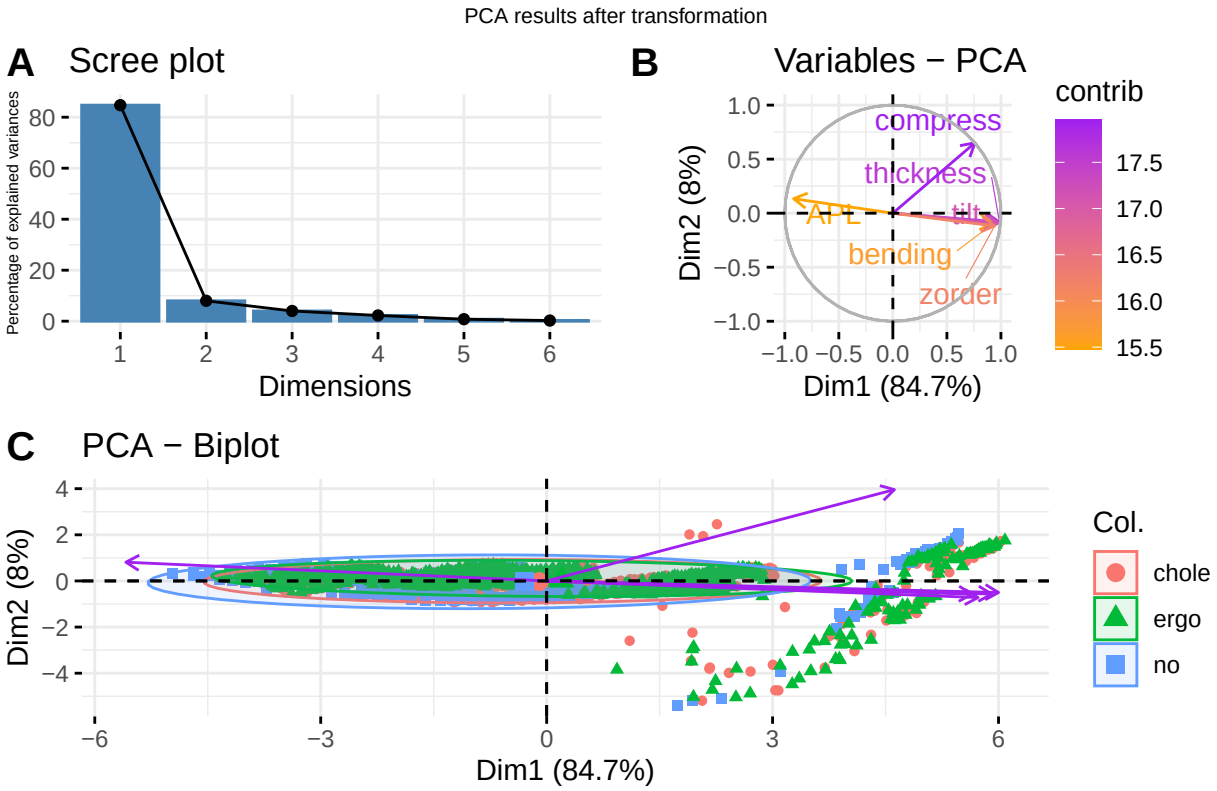
```
# Create PCA plot
new_help <- log10(correlation[2:6])
new_help$APL <- correlation$APL^2
apply(new_help, 2, skewness, na.rm =TRUE)
```

```
##  thickness    bending       tilt     zorder   compress        APL
##  0.5275196  0.7523384  1.2834053 -0.4950526  1.0032745 -0.2422874
```

```
res.pca <- prcomp(new_help, scale = TRUE)
plot1 <- fviz_eig(res.pca) +
  theme(axis.title.y = element_text(size=6))
plot2 <- fviz_pca_var(res.pca,
                      col.var = "contrib", # Colour by contributions to the PC
                      gradient.cols = c("orange", "purple"),
                      repel = TRUE     # Avoid text overlapping
                      )
plot3 <- fviz_pca_biplot(res.pca, repel = FALSE,
                         col.var = "purple", # Variables colour
                         col.ind = nona_data$sterol.type,  # Individuals colour
                         label = FALSE,
                         addEllipses =  TRUE,
```

```
                          ellipse.type = "t"
                          )
# Arrange plots
plot <- arrangeGrob(plot1,  plot2,
                    plot3,
                    ncol = 2, nrow = 2,
                    layout_matrix = rbind(c(1,2), c(3,3)))

plot <- as_ggplot(plot) +
  draw_plot_label(label = c("A", "B", "C"), size = 15,
                  x = c(0, 0.5, 0), y = c(1, 1, 0.5))
title <- expression(atop(bold("Figure 14:"),
                          scriptstyle("PCA results after transformation")))
annotate_figure(plot,
                top=text_grob(title))
```

Figure 14:



As we can see in figure 14, reducing the skewness had absolutely no effect, besides increasing the variance expained by the first dimension and flipping the PCA upside down.

```
kable(head(data), caption="Original data") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

```
kable(head(scaled_data), caption="Cleaned data") %>%
  kable_styling(latex_options = c("scale_down", "hold_position"))
```

Table 7: Original data

temperature	sterol.type	sterol.conc	other.phosph	tails	satur.index	PC.conc	ethanol.conc	APL	thickness	bending	tilt	zorder	compress
298	chole	20	PE	PI	1	33	20	0.736911	3.628973	10.08020	14.7578	0.175499	24.18218
298	chole	20	PE	PI	1	50	20	0.746226	3.596991	10.14030	14.3075	0.172084	24.32495
298	chole	20	PE	PI	1	25	20	0.731945	3.652637	10.25470	14.9777	0.178494	23.76607
298	chole	20	PE	PI	1	100	20	0.769630	3.496052	10.02320	13.2029	0.163349	22.96077
298	chole	20	PE	PI	1	0	20	0.719509	3.710018	10.46540	15.7530	0.182191	23.10503
298	chole	20	PE	PI	1	75	20	0.760091	3.538908	9.86163	13.7148	0.166807	23.86050

Table 8: Cleaned data

APL	thickness	bending	tilt	zorder	compress	sterol.type
0.6474103	0.2983812	0.0789085	0.0361181	0.1640910	0.0426240	chole
0.6685354	0.2831408	0.0793886	0.0340809	0.1603082	0.0428898	chole
0.6361482	0.3096582	0.0803025	0.0371130	0.1674086	0.0418494	chole
0.7216122	0.2350388	0.0784531	0.0290836	0.1506323	0.0403502	chole
0.6079452	0.3370028	0.0819857	0.0406205	0.1715037	0.0406188	chole
0.6999791	0.2554618	0.0771624	0.0313995	0.1544628	0.0420252	chole

5.1 Min Max

This is code to make a csv file containing the minimums and maximums of every column so that we can use it in the java wrapper.

```
columns <- c("APL", "thickness", "bending",
             "tilt", "zorder","compress")

min_max1 <- tibble("Column" = columns,
                   "Min" = apply(nona_data[, columns], 2, min),
                   "Max" = apply(nona_data[, columns], 2, max))
write.csv(min_max1, "Min_Max/min_max", row.names = FALSE)
```

6 Weka

6.1 Rescaling data

Here we rescale all of the possible labels to factors so when it is exported to a .arff file it will be encoded correctly.

```
scale_min_max <- function(x) (x - min(x)) / (max(x) - min(x))
nona_copy <- nona_data

#str(nona_copy)

nona_copy[1:8] <- lapply(nona_copy[1:8], factor)
str(nona_copy)
```

```
## tibble [2,690 x 14] (S3: tbl_df/tbl/data.frame)
##  $ temperature : Factor w/ 2 levels "298","328": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sterol.type : Factor w/ 3 levels "chole","ergo",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
##  $ sterol.conc : Factor w/ 4 levels "0","10","20",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ other.phosph: Factor w/ 1 level "PE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tails       : Factor w/ 5 levels "DI","DO","DP",..: 4 4 4 4 4 4 4 2 2 2 ...
##  $ satur.index : Factor w/ 4 levels "0","0.5","1",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ PC.conc     : Factor w/ 7 levels "0","25","33",..: 3 4 2 7 1 6 5 7 4 3 ...
##  $ ethanol.conc: Factor w/ 7 levels "0","5","10","15",..: 5 5 5 5 5 5 5 2 2 2 ...
##  $ APL         : num [1:2690] 0.737 0.746 0.732 0.77 0.72 ...
##  $ thickness   : num [1:2690] 3.63 3.6 3.65 3.5 3.71 ...
##  $ bending     : num [1:2690] 10.1 10.1 10.3 10 10.5 ...
##  $ tilt        : num [1:2690] 14.8 14.3 15 13.2 15.8 ...
##  $ zorder      : num [1:2690] 0.175 0.172 0.178 0.163 0.182 ...
##  $ compress    : num [1:2690] 24.2 24.3 23.8 23 23.1 ...
##  - attr(*, "na.action")= 'omit' Named int [1:153] 281 400 401 402 403 404 405 406 414 415 ...
##   ..- attr(*, "names")= chr [1:153] "281" "400" "401" "402" ...
```

```
nona_copy_scaled <- nona_copy

index <- sapply(nona_copy_scaled, is.numeric)
nona_copy_scaled[index] <- lapply(nona_copy_scaled[index], scale_min_max)

#scaled_data <- as.data.frame(lapply(nona_copy, FUN=scale_min_max))
#hist.data.frame(scaled_data)
#str(nona_copy)
```

Here is a function to put a specific column as the last column for the encoding in the .arff file.

```
library(RWeka)
Mutate_df <- function(data, last_wanted_column){
  data %>%
    relocate(all_of(last_wanted_column), .after = last_col()) %>%
    select(tail(names(.), 7)) %>%
    #mutate_at(last_wanted_column, as.factor) %>%
    write.arff(file = paste0("data/scaled_data_", last_wanted_column, ".arff"))
}
Mutate_df(nona_copy_scaled, "tails")
Mutate_df(nona_copy_scaled, "sterol.type")
Mutate_df(nona_copy_scaled, "sterol.conc")
```

Here is some code to make a .arff file.

```
library(RWeka)
write.arff(scaled_data, file = "data/scaled_data.arff")
```

6.2 Quality metrics

The two most important metrics that we'll be using are the accuracy and the confusion matrix. The accuracy is important becuase it tells us how accurate the algorithm is. The confusion matrix is also very important seeing as this can tell us if the model may be prediction some classes with high accuracy while have a very poor accuracy on other classes.

6.3 Performance

The speed of the algorithm isn't of the utmost importance, we would rather have a very accurate model that is slow as hell, then a really fast one at the cost of accuracy. Scalability wise, it should be able to deal with a rather large amount of data. We could use online classification, but it seems like this wouldn't really be productive seeing as the distribution or the classification method won't change over time. With batch classification we keep full control over the data that we use when training, while with online classification we are at risk that due to a feed of bad data that could be entered, which would then change the model. To combat this we would need to filter all of the data that goes in, which is feasible but a lot of work. It would save us resources but this isn't really a concern to start with. Batch learning is also more time intesive than online learning, but this also isn't really a concern.

6.4 Output

Some of the subsections are named experiment followed by a number. This is because this name corresponds to the file where the experiment settings can be found to rerun the experiment in weka. The experiment setups can be found in the Experiment folder. Experiment 1 can be found in the Experiment.exp file.

6.4.1 Sterol prediction

The dataset used in this section can be found under data/scaled_data_sterol.type.arff.

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 0 -col-name-width 0 -row-name-width 0 -mean-width 2 -stddev-width 2 -sig-width 1 -count-width 0 -print-row-names -enum-col-names"
Analysing: Percent_correct
Datasets: 1
Resultsets: 9
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 10/10/21, 4:18 PM

Table 9: Sterol prediction accuracy

Dataset	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
R-data-frame	42.94	56.65 ○	59.51 ○	47.83 ○	57.30 ○	60.20 ○	57.07 ○	58.03 ○	70.03 ○

○, • statistically significant improvement or degradation

Table 10: Sterol prediction accuracy (Key)

(1) rules.ZeroR '' 48055541465867954
(2) rules.OneR '-B 50' -3459427003147861443
(3) lazy.IBk '-K 100 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(4) bayes.NaiveBayes '' 5995231201785697655
(5) trees.J48 '-C 0.25 -M 200' -217733168393644444
(6) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(7) 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\"' -6585883636378691736
(8) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0' 7397710626304705059
(9) trees.LMT '-I -1 -M 15 -W 0.0' -1113212459618104943

In table 9 we can see the results of specific machine learning models in predicting the sterol type of a given membrane, the settings for every algorithm can be found in 10. These aren't the result we where hoping for. It seems like there is no way of descerning between sterol type, seeing as the highest achieved accuracy

is equal to around 61%. This is quite poor to say the least. Maybe we should take a closer look at the confusion matrix of some of the algorithms to see if we can find out what's going on here. When we look at the confusion matrices of algorithems like LMT, RandomForest and SimpleLogistic, we get a pretty good idea of what is going on here. The confusion matrices of RandomForest and SimpleLogistic are show in table 11. It seems like the algorithms are very capable of predicting whether or not there is a sterol present in the membrane or not. They just can't distinguish between ergosterol and cholesterol. We can test this hypothesis even further by removing all of the instances that have no sterol in the membrane. If what we saw here is indeed what is going on, we would expect to see around a 50% accuracy. If this is the case we could change the encoding in the dataset from "no," "ergo" and "chole" to just "no" and combine ergo and chole into "yes." The entire column can then be changed to just "sterol.present."

Table 11: Confusion matrix from RandomForest and SimpleLogistics, sterol type

RandomForest			
a	b	c	<– classified as
608	535	0	a = chole
261	893	1	b = ergo
1	4	387	c = no

SimpleLogistics			
a	b	c	<– classified as
624	511	8	a = chole
585	564	6	b = ergo
18	2	372	c = no

Here we'll do some quick recoding of the sterol column, and run random forest to see the results. The newly generated dataset will consist of one that has the no sterol column removed and one were ergosterol and cholesterol are recoded to "Present." The data structure will be printed as a way of verifying whether or not everything right.

```
Mutate_df2 <- function(data, last_wanted_column, removed){
  data %>%
    relocate(all_of(last_wanted_column), .after = last_col()) %>%
    select(tail(names(.), 7)) %>%
    #mutate_at(last_wanted_column, as.factor) %>%
    write.arff(file = paste0("data/scaled_data_recoded_",
                             last_wanted_column, removed,
                             ".arff"))
}

Sterol_present <- nona_copy_scaled
names(Sterol_present)[names(Sterol_present) == "sterol.type"] <- "sterol.present"
Sterol_present$sterol.present <- Sterol_present$sterol.present %>%
  fct_collapse(yes = c("ergo","chole"))
Sterol_no_removed <- filter(nona_copy_scaled, `sterol.type` != "no")

str(Sterol_present)
```

```
## tibble [2,690 x 14] (S3: tbl_df/tbl/data.frame)
##  $ temperature   : Factor w/ 2 levels "298","328": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sterol.present: Factor w/ 2 levels "yes","no": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sterol.conc   : Factor w/ 4 levels "0","10","20",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ other.phosph  : Factor w/ 1 level "PE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tails         : Factor w/ 5 levels "DI","DO","DP",..: 4 4 4 4 4 4 4 2 2 2 ...
##  $ satur.index   : Factor w/ 4 levels "0","0.5","1",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ PC.conc       : Factor w/ 7 levels "0","25","33",..: 3 4 2 7 1 6 5 7 4 3 ...
##  $ ethanol.conc  : Factor w/ 7 levels "0","5","10","15",..: 5 5 5 5 5 5 5 2 2 2 ...
##  $ APL           : num [1:2690] 0.647 0.669 0.636 0.722 0.608 ...
```

```
##  $ thickness     : num [1:2690] 0.298 0.283 0.31 0.235 0.337 ...
##  $ bending       : num [1:2690] 0.0789 0.0794 0.0803 0.0785 0.082 ...
##  $ tilt          : num [1:2690] 0.0361 0.0341 0.0371 0.0291 0.0406 ...
##  $ zorder        : num [1:2690] 0.164 0.16 0.167 0.151 0.172 ...
##  $ compress      : num [1:2690] 0.0426 0.0429 0.0418 0.0404 0.0406 ...
##  - attr(*, "na.action")= 'omit' Named int [1:153] 281 400 401 402 403 404 405 406 414 415 ...
##   ..- attr(*, "names")= chr [1:153] "281" "400" "401" "402" ...
```

```
str(Sterol_no_removed)
```

```
## tibble [2,298 x 14] (S3: tbl_df/tbl/data.frame)
##  $ temperature : Factor w/ 2 levels "298","328": 1 1 1 1 1 1 1 1 1 1 ...
##  $ sterol.type : Factor w/ 3 levels "chole","ergo",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ sterol.conc : Factor w/ 4 levels "0","10","20",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ other.phosph: Factor w/ 1 level "PE": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tails       : Factor w/ 5 levels "DI","DO","DP",..: 4 4 4 4 4 4 4 2 2 2 ...
##  $ satur.index : Factor w/ 4 levels "0","0.5","1",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ PC.conc     : Factor w/ 7 levels "0","25","33",..: 3 4 2 7 1 6 5 7 4 3 ...
##  $ ethanol.conc: Factor w/ 7 levels "0","5","10","15",..: 5 5 5 5 5 5 5 2 2 2 ...
##  $ APL         : num [1:2298] 0.647 0.669 0.636 0.722 0.608 ...
##  $ thickness   : num [1:2298] 0.298 0.283 0.31 0.235 0.337 ...
##  $ bending     : num [1:2298] 0.0789 0.0794 0.0803 0.0785 0.082 ...
##  $ tilt        : num [1:2298] 0.0361 0.0341 0.0371 0.0291 0.0406 ...
##  $ zorder      : num [1:2298] 0.164 0.16 0.167 0.151 0.172 ...
##  $ compress    : num [1:2298] 0.0426 0.0429 0.0418 0.0404 0.0406 ...
##  - attr(*, "na.action")= 'omit' Named int [1:153] 281 400 401 402 403 404 405 406 414 415 ...
##   ..- attr(*, "names")= chr [1:153] "281" "400" "401" "402" ...
```

```
Mutate_df2(Sterol_present, "sterol.present", "")
Mutate_df2(Sterol_no_removed, "sterol.type", "_removed_no")
```

Everything seems to have gone right with the encoding. Now lets run experiment 1 again on the new files. Before running the experiment the @relation in both files were changed to "Sterol_Present" in scaled_data_recoded_sterol.present.arff and "Removed_No" in scaled_data_recoded_sterol.type_removed_no.arff. This is because the write.arff function does not correctly give a name to the given file. The relation argument also doesn't seem to work correctly.

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05
-result-matrix "weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 2
-col-name-width 0 -row-name-width 0 -mean-width 0 -stddev-width 0 -sig-width 0
-count-width 0 -print-row-names -enum-col-names"
Analysing: Percent_correct
Datasets: 2
Resultsets: 9
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 10/29/21, 5:10 PM

After the recording we find that we can get a staggering 99.75% percent accuracy when predicting whether or not a sterol is present in the membrane using the RandomForest algorithm. But then when looking at specifically distinguishing between ergosterol and cholesterol we only get a measly 57.54% accuracy as the highest accuracy. These results are visible in 12, the keys are the same as we saw in 12. What we have inadvertently done here is created evidence to support the theory that there in no significant difference in

Table 12: Recoded sterol type prediction accuracy

Dataset	(6)	(1)	(2)	(3)	(4)	(5)	(7)	(8)	(9)
Sterol-Present	99.75	85.43 •	93.36 •	92.93 •	86.91 •	92.78 •	90.35 •	98.99 •	99.94
Removed-No	53.30	50.26 •	55.67	55.06	53.88	57.31 ◦	51.43	52.90	57.54 ◦

◦, • statistically significant improvement or degradation

the effect of cholesterol compared to ergosterol on membrane characteristics. We could choose to pursue this even further, but that isn't really the goal of this project. Because of these findings I have decided to use the RandomForest algorithm, seeing as is it significantly better than every other algorithm except for LMT. I will use the model that predicts whether or not a sterol is present in the membrane, and not what specific sterol it is. This is because as we could see in the results, predicting the specific sterol type in a membrane gives absolutely awful results. There is however, one big drawback to this. This being that the dataset now contains 2298 instances with a sterol present and only 392 that don't contain a sterol in the membrane.

6.4.2 Tails prediction

The dataset used in this section can be found under data/scaled_data_tails.arff.

6.4.2.1 Experiment 1

Testing a multitude of machine learning algorithms.

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 0 -col-name-width 0 -row-name-width 0 -mean-width 2 -stddev-width 2 -sig-width 1 -count-width 0 -print-row-names -enum-col-names"
Analysing: Percent_correct
Datasets: 1
Resultsets: 9
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 10/10/21, 4:26 PM

Table 13: Accuracy tails prediction

Dataset	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
R-data-frame	37.81	82.28 ◦	82.39 ◦	77.83 ◦	81.03 ◦	99.47 ◦	79.71 ◦	99.42 ◦	99.94 ◦

◦, • statistically significant improvement or degradation

Table 14: Accuracy tails prediction (Key)

(1) rules.ZeroR '' 48055541465867954
(2) rules.OneR '-B 50' -3459427003147861443
(3) lazy.IBk '-K 100 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"' -3080186098777067172
(4) bayes.NaiveBayes '' 5995231201785697655
(5) trees.J48 '-C 0.25 -M 200' -217733168393644444
(6) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(7) 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\"' -6585883636378691736
(8) functions.SimpleLogistic '-I 0 -M 500 -H 50 -W 0.0' 7397710626304705059
(9) trees.LMT '-I -1 -M 15 -W 0.0' -1113212459618104943

The results can be found in table 13, the keys/commands for every algorithm can be found in table 14. It looks like RandomForest and LMT score very high. Lets set RandomForest as our base now to compare which is significantly better.

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 2 -col-name-width 0

-row-name-width 0 -mean-width 0 -stddev-width 0 -sig-width 0 -count-width 0 -print-row-names -enum-col-names”
Analysing: Percent_correct
Datasets: 1
Resultsets: 9
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 12-10-21 12:48

Table 15: Tails prediction, RandomForest as base of t test

Dataset	(6)	(1)	(2)	(3)	(4)	(5)	(7)	(8)	(9)
R-data-frame	99.47	37.81 •	82.28 •	82.39 •	77.83 •	81.03 •	79.71 •	99.42	99.94 ◦

◦, • statistically significant improvement or degradation

Table 16: Tails prediction, RandomForest base (Key)

(1) rules.ZeroR ” 48055541465867954
(2) rules.OneR ’-B 50’ -3459427003147861443
(3) lazy.IBk ’-K 100 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"’ -3080186098777067172
(4) bayes.NaiveBayes ” 5995231201785697655
(5) trees.J48 ’-C 0.25 -M 200’ -217733168393644444
(6) trees.RandomForest ’-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1’ 1116839470751428698
(7) 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\"’ -6585883636378691736
(8) functions.SimpleLogistic ’-I 0 -M 500 -H 50 -W 0.0’ 7397710626304705059
(9) trees.LMT ’-I -1 -M 15 -W 0.0’ -1113212459618104943

Table 15 shows that RandomForest is significantly better than most of the algorithms. However, LMT is significantly better, and SimpleLogistic isn’t significantly better nor significantly worse. Even though LMT seems to be the best at the moment, I don’t think I’ll use it. This is for the simple reason that I do not have enough knowledge on the algorithm to understand how it works, which means I do not have enough confidence in using it. It’s also understandable that RandomForest came does quite well, seeing as it is technically a form of ensemble learning.

6.4.2.2 Enseble Learning (Experiment 4) Testing ensemble learning algorithms with the best scoring algorithms.

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix “weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 0 -col-name-width 0 -row-name-width 0 -mean-width 0 -stddev-width 0 -sig-width 0 -count-width 0 -print-row-names -enum-col-names”
Analysing: Percent_correct
Datasets: 1
Resultsets: 6
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 12-10-21 14:41

Here we look at the area under the curve for all of the algoritms to check for a difference, the results are shown in table 17. It uses the same algorithms/keys as in table 16, also shown in table 18 for convenience.

These are the results for the area under the curve for the same experiment.

Table 17: Ensemble learning tails prediction accuracy

Dataset	(1)	(2)	(3)	(4)	(5)	(6)
R-data-frame	99.47	99.41	99.69	94.41 •	99.29	99.48

◦, • statistically significant improvement or degradation

Table 18: Ensemble learning tail prediction (Key)

(1) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(2) meta.Vote '-S 1 -B \"trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1\" -B \"functions.SimpleLogistic -I 0 -M 500 -H 50 -W 0.0\" -B \"lazy.IBk -K 100 -W 0 -A \\\"weka.core.neighboursearch.LinearNNSearch -A \\\\\\\"weka.core.EuclideanDistance -R first-last\\\\\\\"\\\"\" -R AVG' -637891196294399624
(3) meta.Stacking '-X 10 -M \"trees.J48 -C 0.25 -M 2\" -S 1 -num-slots 1 -B \"trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1\" -B \"functions.SimpleLogistic -I 0 -M 500 -H 50 -W 0.0\" -B \"lazy.IBk -K 100 -W 0 -A \\\"weka.core.neighboursearch.LinearNNSearch -A \\\\\\\"weka.core.EuclideanDistance -R first-last\\\\\\\"\\\"\"' 5134738557155845452
(4) meta.MultiClassClassifier '-M 0 -R 2.0 -S 1 -W functions.Logistic – -R 1.0E-8 -M -1 -num-decimal-places 4' -3879602011542849141
(5) meta.Bagging '-P 100 -S 1 -num-slots 1 -I 10 -W trees.RandomForest – -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' -115879962237199703
(6) meta.AdaBoostM1 '-P 100 -S 1 -I 10 -W trees.RandomForest – -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' -1178107808933117974

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -V
-result-matrix "weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 0
-col-name-width 0 -row-name-width 0 -mean-width 0 -stddev-width 0 -sig-width 0
-count-width 0 -show-stddev -print-row-names -enum-col-names"
Analysing: Area_under_ROC
Datasets: 1
Resultsets: 6
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 13-10-21 13:39

Table 19: AUC, tail predictions

Dataset	(1)	(2)	(3)	(4)	(5)	(6)
R-data-frame	1.00±0	1.00±0	1.00±0	1.00±0	1.00±0	1.00±0

◦, • statistically significant improvement or degradation

These results shown in table 19 are quit weird. All of the AUC (Area Under the Curve) values are 1.00 with a standard deviation of 0. Which is quite interesting, seeing as the accuracy isn't 100%. This probably has to do with the fact that there are 5 different labels, of which some can be predicted with a 100% accuracy. It probably chooses the first label as positive, which in this case would be DI, which indeed has an area under the curve of 1.

6.4.2.3 Attribute selection (Experiment 3) These are the result of attribute selection on the random forest algorithm.

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05
-result-matrix "weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 2
-col-name-width 0 -row-name-width 0 -mean-width 0 -stddev-width 0 -sig-width 0
-count-width 0 -print-row-names -enum-col-names"
Analysing: Percent_correct
Datasets: 1
Resultsets: 2
Confidence: 0.05 (two tailed)

Sorted by: -
Date: 10/25/21, 6:15 PM

Table 20: Attribute selection accuracy

Dataset	(1)	(2)
R-data-frame	99.47	99.54

○, ● statistically significant improvement or degradation

Table 21: Attribute selection accuracy (Key)

(1) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(2) meta.AttributeSelectedClassifier '-E \"WrapperSubsetEval -B trees.RandomForest -F 5 -T 0.01 -R 1 -E DEFAULT – -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1\" -S \"BestFirst -D 1 -N 5\" -W trees.RandomForest – -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' -1151805453487947577

As we can see in table 20, attribute selection doesn't make a significant difference in results. This would lead to the conclusion that it is probably best to run the model with less attributes, seeing as less input would be better. However, for the final program it isn't actually that useful to use less attributes seeing as we'll be using a multitude of models to predict multiple components of the given membrane. Most of these models use different attributes, so we'll probably still need all of them as input for our program.

6.4.2.4 A comment on hyperparamater selection/optimization

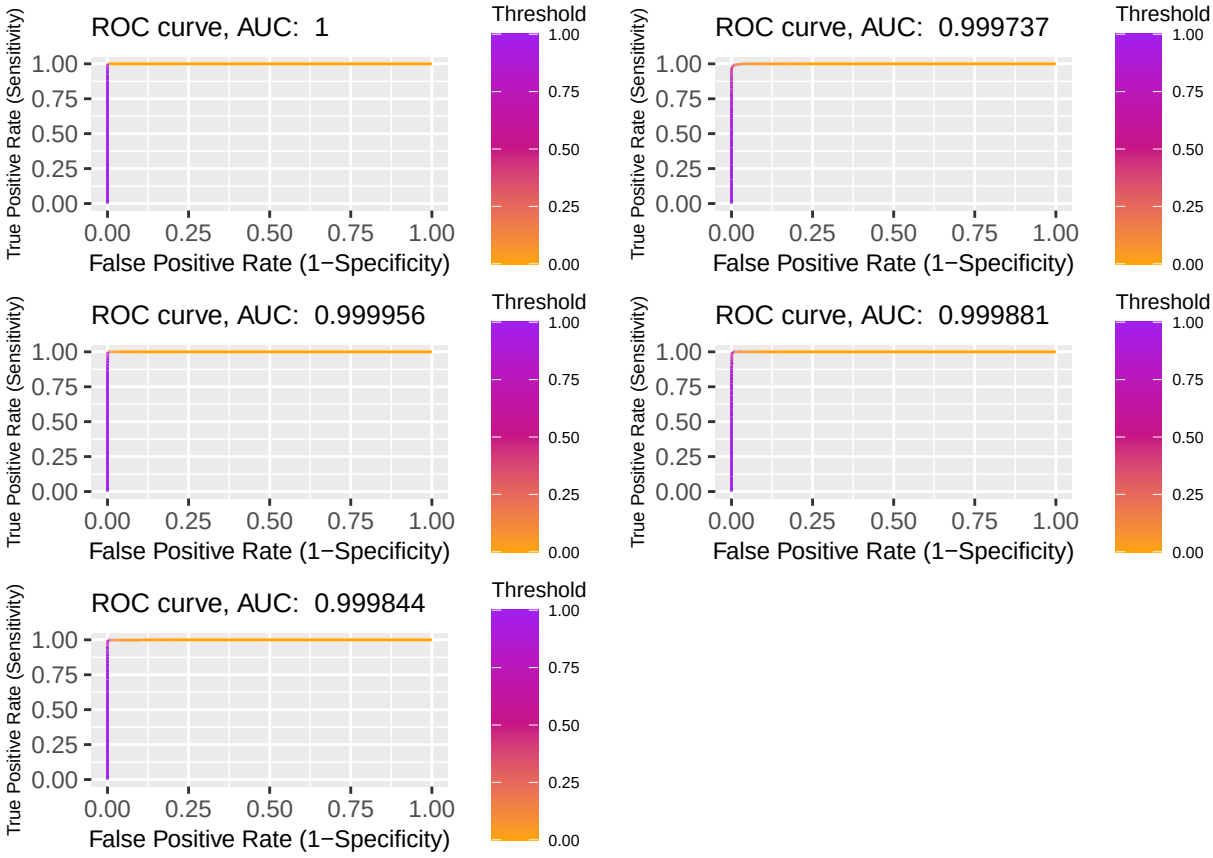
I would love to be able to do some parameter optimization on the LMT and RandomForest algorithms. However, whenever I try to do this, Weka won't let me. It constantly gives errors, which is the reason why there isn't a complete subchapter on this subject.

6.4.2.5 ROC curve

Here we'll plot the ROC curve of the RandomForest algorithm.

```
#library(pROC)

AUC_calculator <- function(TPR, FPR){
  dFPR <- c(diff(FPR), 0)
  dTPR <- c(diff(TPR), 0)
  abs(sum(TPR * dFPR) + sum(dTPR * dFPR)/2)
}

ROC_variable_maker <- function(filehandle){
  open_file <- RWeka::read.arff(filehandle)
  AUC <- with(open_file, AUC_calculator(`True Positive Rate`, `False Positive Rate`))
  #print(head(open_file))
  plot <- ggplot(open_file)+
    geom_path(aes(x=`False Positive Rate`,
                  y=`True Positive Rate`,
                  color=Threshold)) +
    scale_color_gradient2(low="orange", mid="mediumvioletred",
                          high="purple", midpoint = 0.5) +
    labs(title= paste("ROC curve, AUC: ", round(AUC, 6)),
         x = "False Positive Rate (1-Specificity)",
         y = "True Positive Rate (Sensitivity)") +
    theme(plot.title = element_text(size=10),
          axis.title.x=element_text(size=9),
```

```
          axis.title.y=element_text(size=7),
          legend.title = element_text(size = 8),
          legend.text = element_text(size = 6))
  return(plot)

}

ROC_creator <- function (filefolder){
  files <- list.files(path=filefolder, pattern="*.arff", full.names=TRUE, recursive=FALSE)
  #print(files)
  plots <- lapply(files, ROC_variable_maker)

  n <- length(plots)
  nCols <- floor(sqrt(n))
  do.call("grid.arrange", c(plots, ncol= nCols))
}

ROC_creator("./Experiment/Results_Explorer/ROC_Curve")
```



All of the ROC curves look extremely good seeing as they are all at what is basically a right angle. We can also see this in the area under the curve, which is close to one for all for all of the different labels.

6.4.3 Sterol concentration prediction

This section will be a bit less extensive than the tails prediction because of certain time restraints. The keys are again the exact same ones shown in table 10 because the same experiment was used. The dataset used

in this section can be found under data/scaled_data_sterol.conc.arff.

6.4.3.1 Experiment 1 These are the results of running the experiment 1 configuration on the dataset with sterol concentration on as the last column.

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05 -result-matrix "weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 0 -col-name-width 0 -row-name-width 0 -mean-width 0 -stddev-width 0 -sig-width 0 -count-width 0 -print-row-names -enum-col-names"
Analysing: Percent_correct
Datasets: 1
Resultsets: 9
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 10/23/21, 2:29 PM

Table 22: Sterol concentration prediction accuracy

Dataset	(9)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
R-data-frame	99.94	37.81 •	82.28 •	82.39 •	77.83 •	81.03 •	99.47 •	79.71 •	99.42 •

◦, • statistically significant improvement or degradation

Table 23: Sterol concentration prediction accuracy (Key)

(1) rules.ZeroR ” 48055541465867954
(2) rules.OneR ’-B 50’ -3459427003147861443
(3) lazy.IBk ’-K 100 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\"\"’ -3080186098777067172
(4) bayes.NaiveBayes ” 5995231201785697655
(5) trees.J48 ’-C 0.25 -M 200’ -217733168393644444
(6) trees.RandomForest ’-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1’ 1116839470751428698
(7) 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\"’ -6585883636378691736
(8) functions.SimpleLogistic ’-I 0 -M 500 -H 50 -W 0.0’ 7397710626304705059
(9) trees.LMT ’-I -1 -M 15 -W 0.0’ -1113212459618104943

Looking at table 22, it seems like the LMT tree gives significantly better results then any of the other models. Let’s take a look whether or not we can get some sort of improvements with ensemble learning. Here I will use the LMT tree, seeing as I now have a better understanding of it due to some papers. However, due to time restraints, I am unable to redo all of the experiments for the tails prediction and make a new model. It is also important to note that the minimum number of instances was set to 100 in a separate run to try and prevent overfitting. Funnily enough, this didn’t change the the model at all. The model at a minimum number of instances is the exact same as the one at 100. Both of the models can be found under Experiment/Results_Explorer/LMT_Models. The model also doesn’t seem to be overfitted at 15 seeing as these results are from 10 fold cross-validation and the tree doesn’t look to complex. The resulting tree is printed below to illustrate that the tree is indeed not very complex, which would be a sign of overfitting.

=== Classifier model (full training set) ===

Logistic model tree

bending <= 0.054892: LM_1:116/232 (215)
bending > 0.054892
| thickness <= 0.894825
| | compress <= 0.089242
| | | APL <= 0.578692: LM_2:116/580 (894)
| | | APL > 0.578692: LM_3:116/580 (1029)
| | compress > 0.089242: LM_4:116/464 (384)
| thickness > 0.894825: LM_5:116/348 (168)

Number of Leaves : 5

Size of the Tree : 9

6.4.3.2 Ensemble learners (Experiment 4) Here are the results of making ensemble learners using the best performing algorithms.

Tester: weka.experiment.PairedCorrectedTTester -G 4,5,6 -D 1 -R 2 -S 0.05
-result-matrix "weka.experiment.ResultMatrixLatex -mean-prec 2 -stddev-prec 0
-col-name-width 0 -row-name-width 0 -mean-width 0 -stddev-width 0 -sig-width 0
-count-width 0 -print-row-names -enum-col-names"
Analysing: Percent_correct
Datasets: 1
Resultsets: 6
Confidence: 0.05 (two tailed)
Sorted by: -
Date: 10/23/21, 7:29 PM

Table 24: Ensemble learning sterol concentration prediction accuracy

Dataset	(1)	(2)	(3)	(4)	(5)	(6)
R-data-frame	99.73	99.64	99.80	99.70	99.72	36.69 •

○, • statistically significant improvement or degradation

Table 25: Ensemble learning sterol concentration prediction accuracy (Key)

(1) trees.LMT '-I -1 -M 15 -W 0.0' -1113212459618104943
(2) trees.RandomForest '-P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1' 1116839470751428698
(3) meta.Vote '-S 1 -B \"trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1\" -B \"trees.LMT -I -1 -M 15 -W 0.0\" -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\\\"\" -R AVG' -637891196294399624
(4) meta.Stacking '-X 10 -M \"trees.J48 -C 0.25 -M 2\" -S 1 -num-slots 1 -B \"trees.RandomForest -P 100 -I 100 -num-slots 1 -K 0 -M 1.0 -V 0.001 -S 1\" -B \"trees.LMT -I -1 -M 15 -W 0.0\" -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\\\"\"' 5134738557155845452
(5) meta.Bagging '-P 100 -S 1 -num-slots 1 -I 10 -W trees.LMT – -I -1 -M 15 -W 0.0' -115879962237199703
(6) meta.AdaBoostM1 '-P 100 -S 1 -I 10 -W trees.DecisionStump' -1178107808933117974

Now this is where stuff gets really interesting. The results of the experiment can be found in table 24, the keys for the experiment can be found in table 25. No algorithm is significantly better than just LMT, except for adaboost, which preforms horrendously bad. But now, for some reason, it also shows that RandomForest does not actually perform significantly worse than LMT. These result are contradictory to what we found in the previous experiment.

6.5 Temporary results

After running a multitude of test in weka to try and make a model to predict sterol type, I found something quite interesting. It seems that it is almost impossible to predict the sterol type in a membrane. By this I mean it is very hard to distinguish between ergosterol and cholesterol. It is however completely possible to predict whether or not there is a sterol present in the membrane. After running some test with the all of the instances with no sterol removed we see that almost every algorithm go's to 50% accuracy, meaning it's basically as good as zeroR. The highest accuracy we achieved (with the no sterol group removed) was a mere 57.54% This gives us an interesting conclusion to the temporary research question. No, it is not possible to predict the sterol in a membrane given certain measurements. It is however possible to distinguish between no sterol and a sterol. This leads me to conclude that there is just very little to no difference between ergosterol

and cholesterol, seeing as when testing to predict the sterol concentration we get multiple algorithms with a higher than 99.5% accuracy using the algorithms as used on sterol type (again on the data with instances with no sterol removed), and only given the measurements.

6.6 A word on attribute selection

The attribute selection algorithms that can be used in weka all take very long to complete. Some results of attribute selection on the final algorithms can be fournd in the Experiments/Results_Explorer folder. However due to time restraints I could not test all of these algorithm on statistical significance. However, for the final program this doesn't really matter seeing as in the results we can see that for every algorithm different attributes where chosen. But when looking at the attributes needed to run all three algorithm we would still need all of the attributes. This means that for our java wrapper it doesn't really matter. However, if I decide that I want the wrapper to be able to only predict specific membrane components, then these models would probably be better. This is due to the fact that gathering the values for the given attributes isn't an easy task, so less attributes would definitely be better in that case.

7 EDA Discussion

I need to make this very clear. The EDA was made with a specific question in mind. This probably exposed the exploration steps to an subconscious bias when making plots and exploring relations between variables.

8 Refereces

Czub, Jacek, and Maciej Baginski. 2006. "Comparative Molecular Dynamics Study of Lipid Membranes Containing Cholesterol and Ergosterol." *Biophysical Journal* 90 (7): 2368–82.

Hung, Wei-Chin, Ming-Tao Lee, Hsien Chung, Yi-Ting Sun, Hsiung Chen, Nicholas E Charron, and Huey W Huang. 2016. "Comparative Study of the Condensing Effects of Ergosterol and Cholesterol." *Biophysical Journal* 110 (9): 2026–33.

Landwehr, Niels, Mark Hall, and Eibe Frank. 2005. "Logistic Model Trees." *Machine Learning* 59 (1-2): 161–205.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Waikato, University of. 2020. *Weka 3: Machine Learning Software in Java*. Hamilton, New Zealand: University of Waikato. https://www.cs.waikato.ac.nz/ml/weka/.