Capstone Project - Predicting Depression from Plasma Measurements

Zsombor Szoke-Kovacs

2022-05-31

Contents

Introduction **2**

Project Background **2**

Project Overview **2**

Methods and Data Analysis Workflow **2**

Data Preparation **2**

Data Analysis **4**

Data Distribution 4

Two-Sample t-Test 7

Linearity Between the Two Arms 11

Model Fitting **11**

Support Vector Machine 16

Random Forest Model 19

Validation **20**

SVM 20

k-NN 21

RMF 21

Results and Conclusions **21**

Future Perspectives **22**

References **22**

Introduction

Major depressive disorder (MDD) is the most common type of mood disorders, affecting as much as 121 million people worldwide (Szoke-Kovacs et al. 2020). MDD is characterized by multiple symptoms including anhedonia, depressed mood, suicidal thoughts, loss of appetite, and psychomotor retardation (Sharma, Santra, and Dutta 2015). Despite the available treatment options (e.g. SSRIs, (Szoke-Kovacs et al. 2020)) and a somewhat limited knowledge about the biology and risk factors of MDD, there is still a lack of information about the pathobiochemical background of the disease. Therefore, in order to provide a better support and an early diagnosis of patients suffering from the disease, it is important to map the molecular alteration and hallmarks of MDD.

Project Background

The Genome and Biomedical Sciences Facility from the University of California have collected cerebrospinal fluid (CSF) and peripheral blood from more than 600 subjects from the general population to map molecular and biochemical properties. Participants then completed the Beck Depression Inventory (BDI), a well-established questionnaire for measuring severity of depression. Subjects who were identified as suffering from MDD (based on their BDI score), were gender and age matched with a control group, and an untargeted analysis of primary metabolites was performed. The primary obejctive of the study was to identify metabolites that could help predicting and/or diagnosing MDD in patients.

Project Overview

In this Capstone Project, I have chosen to work with the data acquired by the Genome and Biomedical Sciences Facility from the University of California from human blood plasma. The data was downloaded from the **https://www.metabolomicsworkbench.org/data/DRCCMetadata.php?Mode=Study&StudyID=ST000062&StudyType=MS&ResultType=1** website, and was converted into a dataframe, which was used for the analysis. An initial distribution analysis was done, followed by two-sample t-test-, correlation-, and k-means clustering analysis. The model building was done on the selected parameters that showed a significant difference between the two arms. The three models used were support vector machines, random forest models, and k-nearest neighbour models. Due to the nature of the measured parameters, normalization of the data was also done in some instances.

Methods and Data Analysis Workflow

During my analysis I applied Shapiro-Wilk's test for mapping distribution of the parameters. This was followed by two-sample t-test. From this latter analysis parameters showing a significant difference were selected and a correlation analysis was done. Next, k-means clustering analysis was done, to see if the clusters can be separated well between the two arms. Next, support vector machine and k-NN models were fit on a standardized data of the selected parameters, and random forest was fit on the non-standardized data.

Data Preparation

First the data was downloaded from the Metabolomics website (see url above). Next, the data set was edited; some of the measured molecules have been removed from the list for simplicity and also due to these did not have names (only ID numbers). A simplified data sheet was used for the analysis.

```
library("readxl")
library("dplyr")
library("ggplot2")
library('ggfortify')
library('corrplot')
library('stats')
library('purrr')
library('caTools')
library('e1071')
library('randomForest')
library('stringr')
library('class')
set.seed(1234)
# Here, I set the file to a path on my computer, but once this is saved to a
# different computer, the path will need to be updated. The simplified data sheet
# can also be downloaded from my git repository:
# https://github.com/zsk2021/CapstoneProject--PredictingDepression
file_original <- "~/Desktop/HarvardX, EdX, Data Science/Capstone Project/Capstone Project - Chosen Proj
temp_file <- read_excel(file_original)
# Converting the temp file into a transposed data frame.
file_df <- as.data.frame(t(temp_file))
# removing unnecessary rows/columns
data <- file_df[c(-2,-3,-4),c(-1,-3,-4,-5,-7)]
```

New column names are introduced:

```
# Adding new column names for the molecules and the arms (groups):
data[1,1] <- 'Samples'
data[1,2] <- 'Arm'
colnames(data) <-data[1,]
# Remove first row from the data frame:
data <- data[-1,]
# By investigating the data, we can see that measurements come from two groups:
# Group 1 (control) and Group 2 (patients diagnosed with depression):
data %>% group_by(Arm) %>% summarise(n = n())
```

```
## # A tibble: 2 x 2
##   Arm                    n
##   <chr>              <int>
## 1 Group 1 - Score 0     48
## 2 Group 2 - Score 50    49
```

To analyse the relationship between the different measurements between the two groups, I first separate the two arms and remove unnecessary columns:

```
group_1 <- data %>% group_by(Arm) %>% filter(Arm == "Group 1 - Score 0")
# A data set without columns containing other characters than numbers is created:
group_1_truncated <- group_1[, c(-1, -2)]
# Same for group 2:
group_2 <- data %>% group_by(Arm) %>% filter(Arm == "Group 2 - Score 50")
group_2_truncated <- group_2[, c(-1, -2)]
```

Data Analysis

Data Distribution

I first look at the distribution of the data for each measurement from the two arms, by generating graphs with the codes below. Due to the code below would result in a high number of plots (144 plots per arm), I commented this out in the .rmd file.

```
# Group 1:

#for (i in group_1_truncated){
# plot <- group_1_truncated %>% ggplot(aes(x = as.numeric(i))) +
#  geom_density()
# print(plot)
#}

# Group 2

#for (i in group_2_truncated){
#  plot <- group_2_truncated %>% ggplot(aes(x = as.numeric(i))) +
#    geom_density()
#  print(plot)
#}
```

Instead, I used Shapiro-Wilk's method (http://www.sthda.com/english/wiki/normality-test-in-r) to get a value of the normality for each measured parameter. The null hypothesis for this test is that "the sample distribution is normal". So, if the p-value is >0.05, that implies that the distribution of the data is not significantly different from the normal distribution. In other words, if the p-value is >0.05 we can assume normality. First, I loop through the truncated and transposed list and generate Shapiro-Wilk's test for each column in the data set. I use the magicfor library to record p-values in a vector:
Note: p-values are not printed in the .rmd file, as specified by the "results= 'hide' " argument.

```
library(magicfor)
# Group 1:
magic_for(print)
for (c in group_1_truncated){
  shap_test <- shapiro.test(as.numeric(c))
  output <- shap_test$p.value
  print(output)
}
# Saving the printed p-values as a vector:
pvalues_group_1 <- magic_result_as_vector()
# Binding vector to the original data, so the last row is the p-value from the
# Shapiro-Wilk's test:
group_1_truncated_with_pvalues <- rbind(group_1_truncated, pvalues_group_1)

# Group 2:
magic_for(print)
for (c in group_2_truncated){
  shap_test <- shapiro.test(as.numeric(c))
  output <- shap_test$p.value
  print(output)
}
```

```
# Saving the printed p-values as a vector:
pvalues_group_2 <- magic_result_as_vector()
# Binding vector to the original data, so the last row is the p-value from the
# Shapiro-Wilk's test:
group_2_truncated_with_pvalues <- rbind(group_2_truncated,pvalues_group_2)
# Remove magicalization:
magic_free()
```

The last row in these two data frames are the Shapiro-Wilk's p-values. The original column numbers are 48 and 49, respectively (see above).

```
group_1_truncated_with_pvalues %>%
  summarise(Arm = 'Group 2',
            nrow = dim(group_1_truncated_with_pvalues)[1],
            ncol = dim(group_1_truncated_with_pvalues)[2])
```

```
## # A tibble: 1 x 3
##   Arm      nrow  ncol
##   <chr>   <int> <int>
## 1 Group 2    49   143
```

```
group_2_truncated_with_pvalues %>%
  summarise(Arm = 'Group 2',
            nrow = dim(group_2_truncated_with_pvalues)[1],
            ncol = dim(group_2_truncated_with_pvalues)[2])
```

```
## # A tibble: 1 x 3
##   Arm      nrow  ncol
##   <chr>   <int> <int>
## 1 Group 2    50   143
```

Now, that I have the p-values for the Shapiro-Wilk's test for the measured parameters from each arm, I transpose the data frames, so the p-values are in a separate column and the data frame is in tidy format:

```
# Group 1 - transpose
group_1_tidy <- as.data.frame(t(group_1_truncated_with_pvalues))
# adding one extra row that will be used for the column names
group_1_tidy<- add_row(group_1_tidy, .before = 1)
# using sample names for column names
group_1_tidy[1,1:48] <- group_1$Samples
# adding name for extra column
group_1_tidy[1,49] <- "Shapiro-Wilk's p-values"
# use first row as column names
colnames(group_1_tidy) <- group_1_tidy[1,]
group_1_tidy <- group_1_tidy[-1,]
# The last column is the Shapiro-Wilk's p-values
group_1_tidy %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy)[1],
                           ncol = dim(group_1_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 1  143   49
```

```
# Group 2 -transpose
group_2_tidy <- as.data.frame(t(group_2_truncated_with_pvalues))
# adding on extra row that will be used for the column names
group_2_tidy<- add_row(group_2_tidy, .before = 1)
# using sample names for column names
group_2_tidy[1,1:49] <- group_2$Samples
# adding name for extra column
group_2_tidy[1,50] <- "Shapiro-Wilk's p-values"
# use first row as column names
colnames(group_2_tidy) <- group_2_tidy[1,]
group_2_tidy <- group_2_tidy[-1,]
# The last column is the Shapiro-Wilk's p-values
group_2_tidy %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy)[1],
                           ncol = dim(group_2_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 2  143   50
```

Now, that I have the data for the two arms, together with the p-values for normal distribution, I filter the data to keep the measured parameters, where the distribution was approximately normal. In other words, I keep all measured parameters, where the p-value was >0.05:

```
group_1_tidy <- group_1_tidy %>% filter(`Shapiro-Wilk's p-values`>0.05)
group_2_tidy <- group_2_tidy %>% filter(`Shapiro-Wilk's p-values`>0.05)
```

There are 97 and 112 measured parameters where the p-value is >0.05 in Group 1 and Group 2, respectively. Group 1 has 48 patients, whereas Group 2 has 49. The extra column in each data frame is the Shapiro-Wilk's p-value.

```
group_1_tidy %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy)[1],
                           ncol = dim(group_1_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   97   49
```

```
group_2_tidy %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy)[1],
                           ncol = dim(group_2_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 2  112   50
```

Due to the number of the normally distributed measured parameters are different in the two groups, I will work with the list from the control group (Group 1 - baseline), where the normally distributed parameters were 97 (as opposed to Group 2 where it was 112). I use semi_join to keep only the records from Group 2, that have a match in Group 1.

```
# Adding row names as an extra column, so I can use semi_join:
group_1_tidy <- cbind(group_1_tidy, rownames = rownames(group_1_tidy))
group_2_tidy <- cbind(group_2_tidy, rownames = rownames(group_2_tidy))
# We should have one extra column in each data frame:
group_1_tidy %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy)[1],
                           ncol = dim(group_1_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   97   50
```

```
group_2_tidy %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy)[1],
                           ncol = dim(group_2_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 2  112   51
```

```
# Keep everything from Group 1 with a match in Group 2:
group_1_tidy <- semi_join(group_1_tidy, group_2_tidy, by = "rownames")
# Keep everything from Group 2 with a match in Group 1:
group_2_tidy <- semi_join(group_2_tidy, group_1_tidy, by = "rownames")
```

Investigating the dimensions of the two newly generated data frames, we can see, that both arms have 78 measured parameters, as well as 48 and 49 sample count (plus the two columns with p-values and row names), respectively.

```
group_1_tidy %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy)[1],
                           ncol = dim(group_1_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   78   50
```

```
group_2_tidy %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy)[1],
                           ncol = dim(group_2_tidy)[2])
```

```
##       Arm nrow ncol
## 1 Group 2   78   51
```

Two-Sample t-Test

In this next section, I will calculate two-sample t-tests for the selected parameters, so I can see if there is a significant difference in any normally distributed parameters between the two groups. First, I transpose the data frame generated above and remove unnecessary rows.

```
# Transpose tidy data, so I can loop through the columns: Group 1
group_1_tidy_t <- as.data.frame(t(group_1_tidy))
# Removing last two rows with p-values and row names; these are not needed for the
# calculation of two-sample t-test.
group_1_tidy_t <- group_1_tidy_t[c(-49,-50),]
# Transpose tidy data, so I can loop through the columns: Group 2
group_2_tidy_t <- as.data.frame(t(group_2_tidy))
# Removing last two rows with p-values and row names; these are not needed for the
# calculation of two-sample t-test.
group_2_tidy_t <- group_2_tidy_t[c(-50,-51),]
# The two data set has 78 measured parameters and 48 and 49 samples, respectively:
group_1_tidy_t %>% summarise(Arm = 'Group 1', nrow = dim(group_1_tidy_t)[1],
                             ncol = dim(group_1_tidy_t)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   48   78
```

```
group_2_tidy_t %>% summarise(Arm = 'Group 2', nrow = dim(group_2_tidy_t)[1],
                             ncol = dim(group_2_tidy_t)[2])
```

```
##       Arm nrow ncol
## 1 Group 2   49   78
```

Now, that I have the two data frames with the same measured parameters in both, and all of the measurements show approximately normal distribution, I can test the vectors for significant differences.
Note: p-values are not printed in the .rmd file, as specified by the "results= 'hide' " argument.

```
# Two-sample t-test by looping through the columns:
magic_for(print)
for (j in seq(ncol(group_1_tidy_t))){
  testresults <- t.test(as.numeric(group_1_tidy_t[,j]), as.numeric(group_2_tidy_t[,j]))
  print(testresults$p.value)
}
```

Saving p-values from the two-sample t-test into a data frame:

```
twosample_ttest <- magic_result_as_dataframe()
magic_free()
# Adding the names of the measured parameters to the p-values:
colnames(twosample_ttest)[1] <- 'rownames'
twosample_ttest$`rownames` <- colnames(group_1_tidy_t)
# Filtering out measured parameters that showed significant differences between
# the two groups:
twosample_ttest_significant <- twosample_ttest %>% filter(`testresults$p.value` <= 0.05)
twosample_ttest_significant
```

```
##               rownames testresults$p.value
## 1         stearic acid        4.942357e-05
## 2             sorbitol        4.571605e-02
## 3        shikimic acid        1.102079e-02
## 4              ribitol        4.199180e-02
## 5       pseudo uridine        1.947676e-03
## 6         nicotinamide        5.894853e-04
## 7         myo-inositol        1.060330e-03
## 8              mannose        2.861927e-02
## 9              lyxitol        2.000258e-03
## 10  heptadecanoic acid        1.549329e-04
## 11       glutaric acid        4.067713e-02
## 12       glutamic acid        1.122825e-02
## 13         citric acid        2.823496e-02
## 14        behenic acid        9.529080e-04
## 15 alpha-ketoglutarate        1.629588e-03
```

There are 15 measured parameters that show normal distribution, and there is a significant difference between the two groups: *## Correlation Analysis*
From the previous section, I have a set of measured parameters that show significant differences of the mean

between the two arms. To see the actual relationship between the two groups, I will use correlation analysis for the 15 parameters. Initially, I will subset the two data frames group_1_tidy and group_2_tidy to only consist of the 15 parameters of interest.

```
group_1_final <- semi_join(group_1_tidy, twosample_ttest_significant, by = 'rownames')
group_1_final <- group_1_final[,c(-49,-50)]

group_2_final <- semi_join(group_2_tidy, twosample_ttest_significant, by = 'rownames')
group_2_final <- group_2_final[,c(-50,-51)]
```

Here I have two data frames from the two arms, one control and one diagnosed with depression, where only the parameters of interest are included. The data frames consist of 48 and 49 patients, respectively:

```
group_1_final %>% summarise(Arm = 'Group 1', nrow = dim(group_1_final)[1],
                            ncol = dim(group_1_final)[2])
```

```
##       Arm nrow ncol
## 1 Group 1   15   48
```

```
group_2_final %>% summarise(Arm = 'Group 2', nrow = dim(group_2_final)[1],
                            ncol = dim(group_2_final)[2])
```

```
##       Arm nrow ncol
## 1 Group 2   15   49
```

Now, that I have the two data frames with the 15 measured parameters that showed approximately normal distribution and significant differences between the two groups, I will merge the two arms, and will generate a new data frame with all of the subjects and the 15 measured parameters. I will use this data frame for my further work:

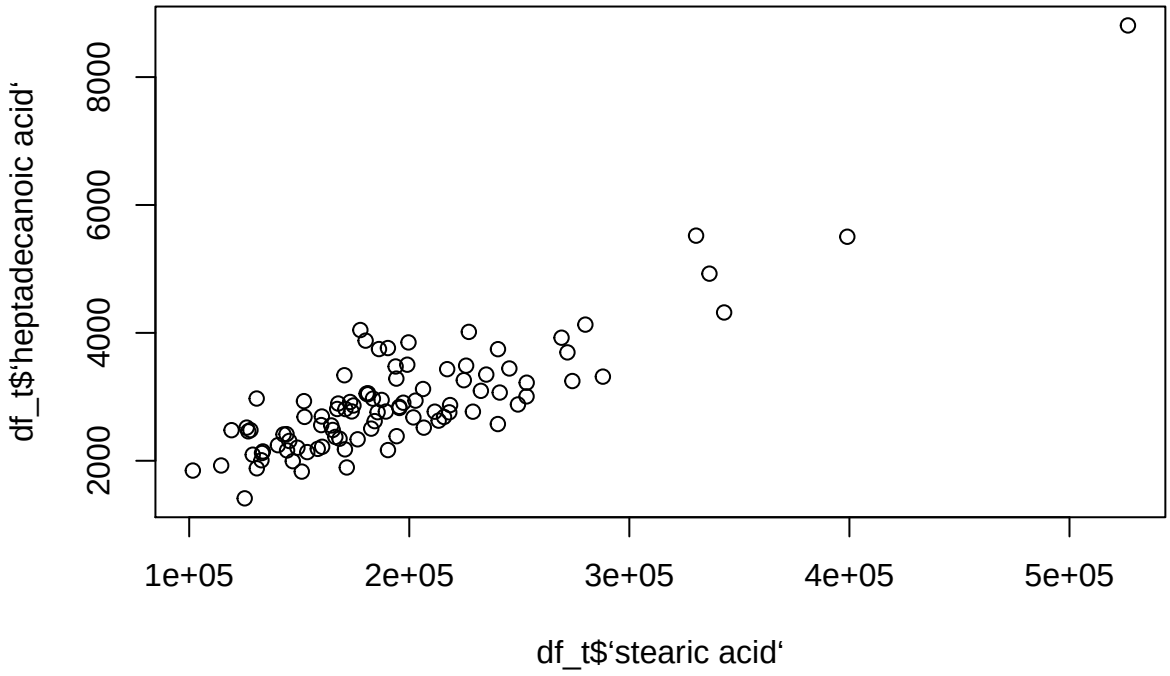
```
# First, I create a new column in both data frames, so I can merge these with
# left_join()
group_1_final <- cbind(group_1_final, rownames = rownames(group_1_final))
group_2_final <- cbind(group_2_final, rownames = rownames(group_2_final))
# Merging the two data frames by rownames:
df <- left_join(group_1_final, group_2_final, by = "rownames")
# Adding rownames based on the rownames column
rownames(df) <- df$rownames
# Removing rownames column results in the final data set df:
df <- subset(df, select = -rownames)
```

In the above section, I generated my data set with all 15 parameters and the entire cohort. I now transpose this and will do a correlation analysis to see if any of these parameters are correlated:

```
df_t <- as.data.frame(t(df))
```

A quick plotting of the data shows that there is a potential correlation between stearic acid and heptadecanoic acid: commented out code, otherwise too many plots would have been printed.

```
# plot(df_t)
plot(df_t$`stearic acid`, df_t$`heptadecanoic acid`)
```




```
# A correlation analysis between the two parameters shows a strong positive correlation
# with a value of 0.862:
cor.test(as.numeric(df_t$`stearic acid`), as.numeric(df_t$`heptadecanoic acid`))
```

```
##
##  Pearson's product-moment correlation
##
## data:  as.numeric(df_t$‘stearic acid‘) and as.numeric(df_t$‘heptadecanoic acid‘)
## t = 16.583, df = 95, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8002618 0.9058056
## sample estimates:
##       cor
## 0.8621075
```

```
# Here, I convert all df_t to numeric, so I can do a correlation analysis:
df_num <-as.data.frame(sapply(df_t, as.numeric))
# This also shows, that the only correlation is between stearic acid and heptadecanoic
# acid:
cor_15_param <- as.data.frame(cor(df_num))
cor_15_param %>% filter(cor_15_param >= 0.7)
```

```
##                     stearic acid  sorbitol shikimic acid    ribitol
## stearic acid           1.0000000 0.2083188     0.2492042 0.15060595
## heptadecanoic acid     0.8621075 0.1501909     0.3128487 0.09976116
##                     pseudo uridine nicotinamide myo-inositol   mannose
## stearic acid           -0.19706050    0.3878188   0.04917488 0.1807085
## heptadecanoic acid     -0.05231672    0.2501691   0.13140503 0.2167668
##                        lyxitol heptadecanoic acid glutaric acid glutamic acid
## stearic acid       0.02011295          0.8621075  -0.034593242    0.05591120
## heptadecanoic acid 0.01543889          1.0000000  -0.005699254   -0.04747537
##                     citric acid behenic acid alpha-ketoglutarate
## stearic acid         0.08475043    0.4184708          0.02048958
## heptadecanoic acid   0.17991391    0.3845567         -0.01077332
```

Linearity Between the Two Arms

I have also looked at the other features, whether these are linearly separable between the two arms. Values from each measured parameters are plotted on y and the arm is plotted on x.

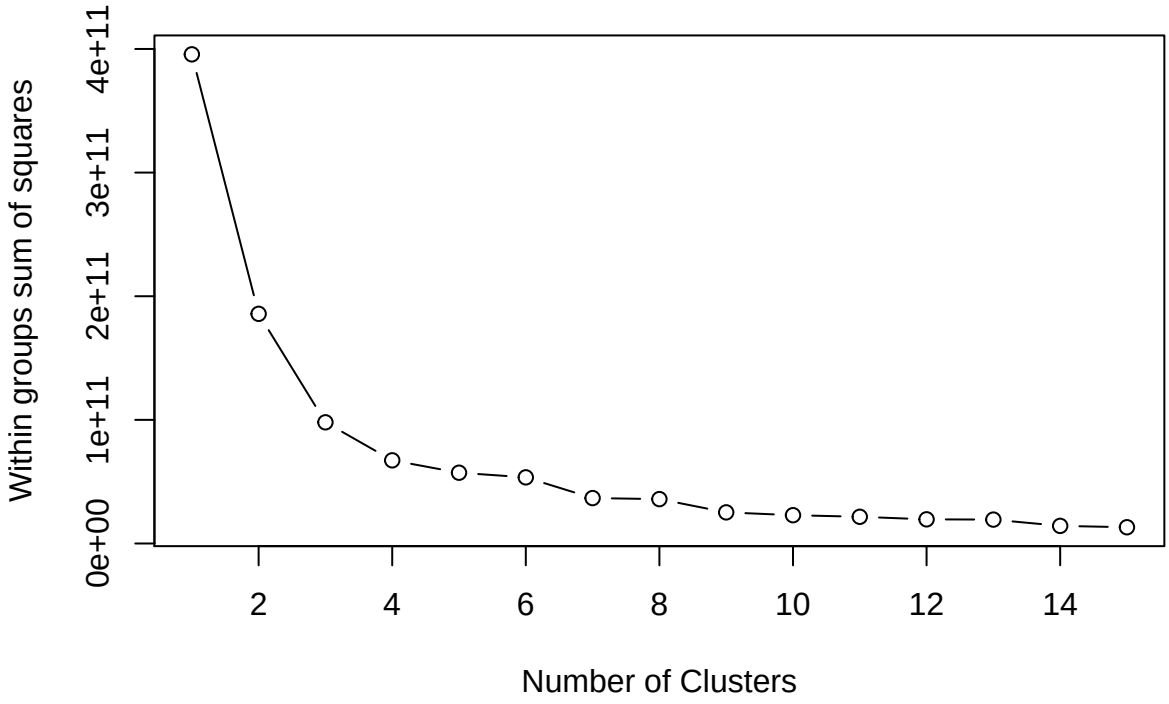
```
# code is commented to avoid the printing of an excess amount of graphs:
# for (v in data[,c(-1,-2)]){
#  plot(as.numeric(v), col = as.factor(data$Arm))
# }
```

Based on the plots generated by the above code, the values do not show a linear association between the two arms. Hence, I will not use linear regression in my analysis.

Model Fitting

In this section of my project, I will fit different models on the data set that contains the 15 normally distributed/significantly different parameters. ## *K-means Clustering*
I used the df_t data set, that is a transposed format of the 15 significant measurements from the two arms. Parameters are in the columns, and samples in rows. First, lets find the optimum number of clusters: we use the wssplot() function, of which the code was copied from this website:
https://www.projectpro.io/data-science-in-r-programming-tutorial/k-means-clustering-techniques-tutorial

```
wssplot <- function(data, nc=15, seed=1234){
                  wss <- (nrow(data)-1)*sum(apply(data,2,var))
                      for (i in 2:nc){
                set.seed(seed)
                    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
             plot(1:nc, wss, type="b", xlab="Number of Clusters",
                          ylab="Within groups sum of squares")
             wss
       }
wssplot(df_num)
```




```
##  [1] 395635828037 185760569486  98013679493  67263552575  57250341995
##  [6]  53490348440  36742712995  35905094218  25183107256  22902610178
## [11]  21609713450  19544771548  19333076704  14248401371  13151570164
```

From the above plot, we can see that the optimum number of clusters 3 (the smallest possible number, where the plot shows an elbow shape). So, we will apply the cluster numbers 3 in our k-means cluster analysis.

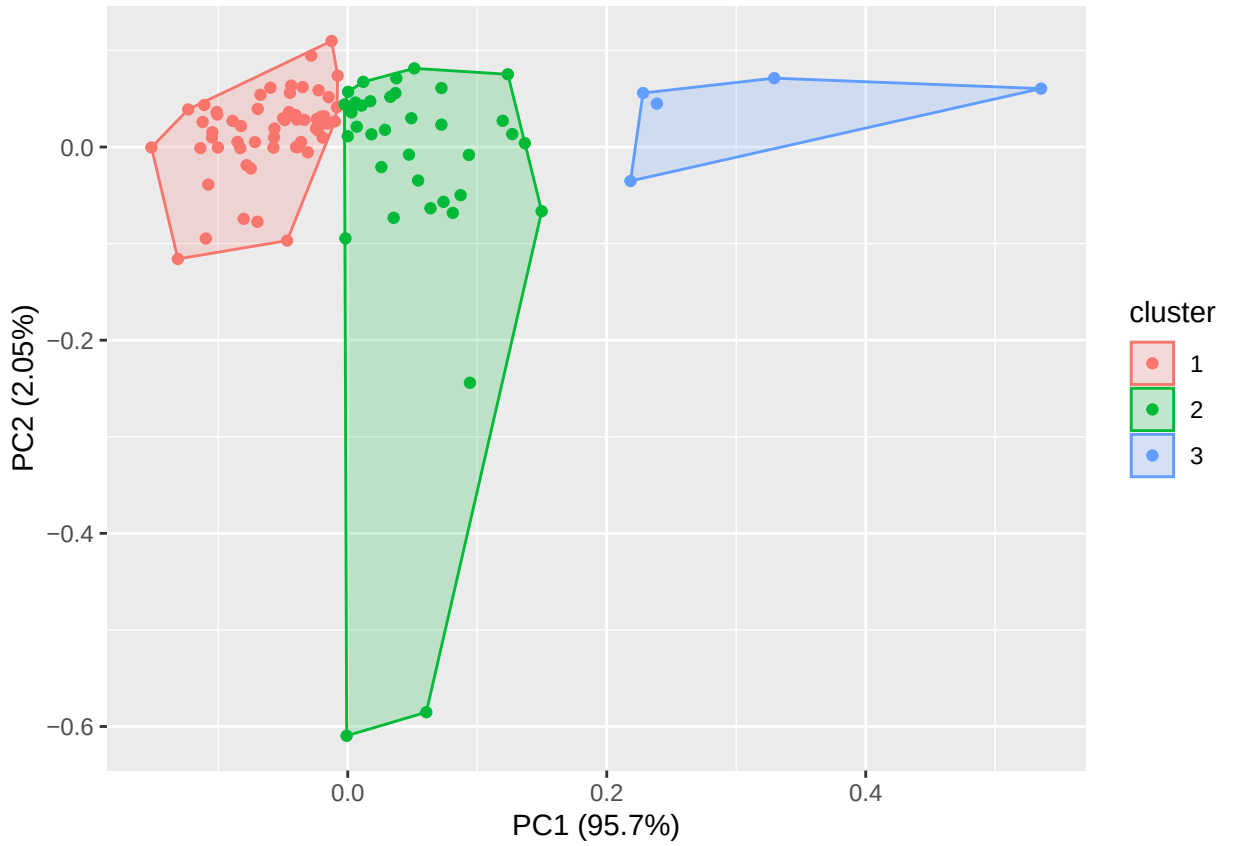
```
KM_3 <- kmeans(df_num, 3)
print(KM_3)
```

```
## K-means clustering with 3 clusters of sizes 56, 36, 5
##
## Cluster means:
##   stearic acid  sorbitol shikimic acid  ribitol pseudo uridine nicotinamide
## 1     158133.1  984.8479      376.7243 342.5996       1425.287     159.2324
## 2     226722.3 3758.6777      404.5479 566.9193       1330.800     428.6950
## 3     387081.3 4226.8173      616.2581 434.8046       1148.915     944.4025
##   myo-inositol  mannose  lyxitol heptadecanoic acid glutaric acid glutamic acid
## 1     8138.416 15893.98 1085.131           2557.490      90.20281      4888.893
## 2     8756.183 14255.69 1170.427           3158.250      85.15203      9104.517
## 3     7566.377 19499.29 1016.876           5815.383      77.25767      4250.937
##   citric acid behenic acid alpha-ketoglutarate
## 1    29348.89     592.8050            175.9726
## 2    29345.78     758.3372            178.3793
## 3    29944.92     927.8507            216.9687
```

```
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 2 2 1 2
## [39] 1 1 2 2 2 1 2 1 1 1 2 3 1 2 1 2 2 2 2 2 2 2 1 2 2 2 2 3 2 2 1 2 2 1 1 2 2 1
## [77] 2 2 3 3 2 1 3 1 1 1 1 1 1 1 1 2 1 1 2 2 1
##
## Within cluster sum of squares by cluster:
## [1] 33485743443 36586389829 27941546221
##  (between_SS / total_SS =  75.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

There are two ways to evaluate cluster analysis: 1.) looking at the cluster plot or or 2.) look at the cluster centers. First we look at the cluster plot by using the autoplot() function:

```
autoplot(KM_3, df_num, frame = TRUE)
```



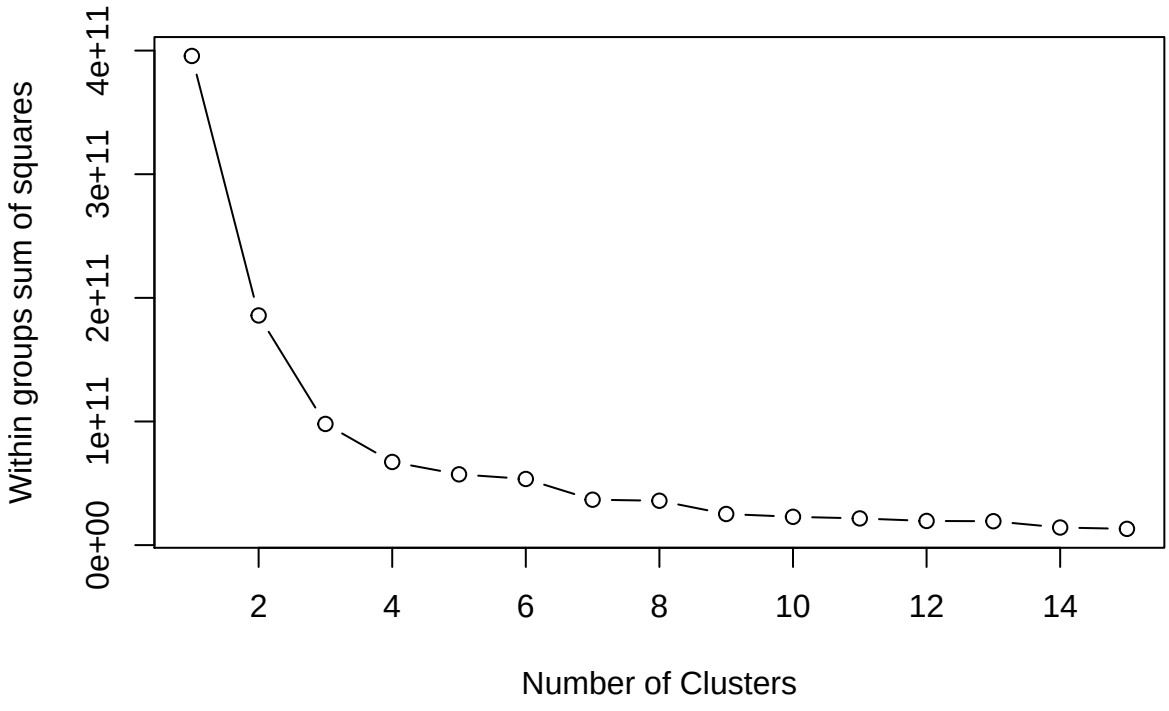
From the above plot, we can see that the clusters 1 and 2 overlap and there is no clear separation between the classes. As the number of observation increases, the cluster plot becomes more 'busy', therefore, another way to evaluate the k-means cluster analysis and see the distinctiveness of the clusters is to look at the center of the particular clusters. Centroids can be derived from the k-means analysis object:

```
KM_3$centers
```

```
##   stearic acid  sorbitol shikimic acid  ribitol pseudo uridine nicotinamide
## 1     158133.1  984.8479      376.7243 342.5996       1425.287     159.2324
## 2     226722.3 3758.6777      404.5479 566.9193       1330.800     428.6950
## 3     387081.3 4226.8173      616.2581 434.8046       1148.915     944.4025
##   myo-inositol  mannose  lyxitol heptadecanoic acid glutaric acid glutamic acid
## 1     8138.416 15893.98 1085.131           2557.490      90.20281      4888.893
## 2     8756.183 14255.69 1170.427           3158.250      85.15203      9104.517
## 3     7566.377 19499.29 1016.876           5815.383      77.25767      4250.937
##   citric acid behenic acid alpha-ketoglutarate
## 1    29348.89     592.8050            175.9726
## 2    29345.78     758.3372            178.3793
## 3    29944.92     927.8507            216.9687
```

From the above values, we can see that the centers of the selected parameters are different, suggesting that the clusters are distinct in nature. A good separation for the clusters in the case of stearic acid, nicotinamide, myo-inositol, mannose, heptadecanoic acid, and glutamic acid can be seen. Now, if I repeat the same analysis with only two clusters (based on the 2 arms and also on a potential elbow on the below plot at cluster 2), I can see a better separation for the centroids in the case of stearic acid, sorbitol, nicotinamide, heptadecanoic acid, glutamic acid, and behenic acid.

```
wssplot(df_num)
```



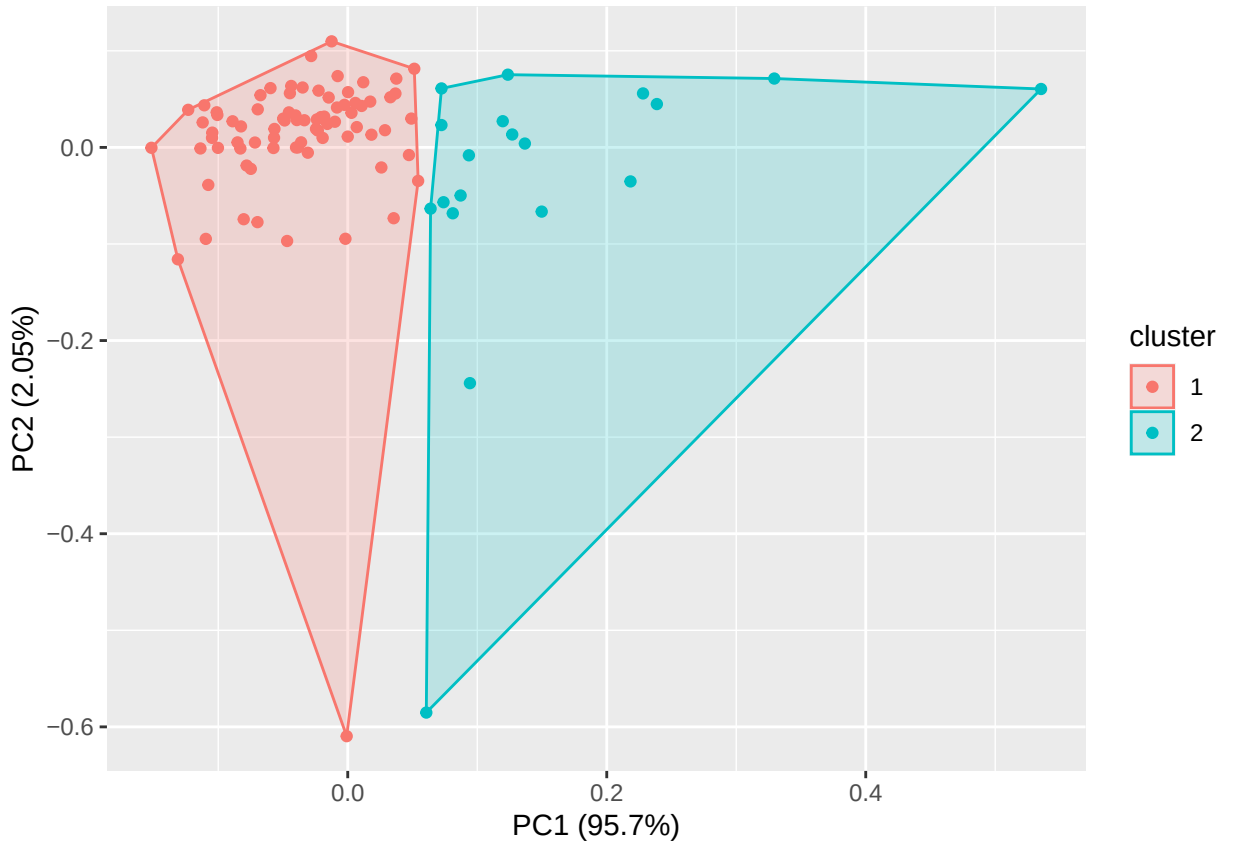

```
##  [1] 395635828037 185760569486  98013679493  67263552575  57250341995
```

```
##  [6]  53490348440  36742712995  35905094218  25183107256  22902610178
## [11]  21609713450  19544771548  19333076704  14248401371  13151570164
```

```
KM_2 <- kmeans(df_num, 2)
print(KM_2)
```

```
## K-means clustering with 2 clusters of sizes 78, 19
##
## Cluster means:
##   stearic acid sorbitol shikimic acid  ribitol pseudo uridine nicotinamide
## 1     172353.6 1680.032      382.2930 403.4542       1394.712     217.5227
## 2     289962.4 4239.760      469.6168 542.0666       1299.050     637.1195
##   myo-inositol  mannose  lyxitol heptadecanoic acid glutaric acid glutamic acid
## 1     8367.910 15398.69 1105.755           2695.191      88.42112      5481.488
## 2     8216.252 15771.91 1144.115           3987.812      84.54062     10275.749
##   citric acid behenic acid alpha-ketoglutarate
## 1    29588.69     630.8289            175.5881
## 2    28515.39     838.5168            192.8996
##
## Clustering vector:
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
## [39] 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 1 1 1 2 2 2 1 1 2 2 1 2 2 2 1 1 1 1 1 1 2 2 1
## [77] 2 2 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##
## Within cluster sum of squares by cluster:
## [1]  82453715161 101353923427
##  (between_SS / total_SS =  53.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
autoplot(KM_2, df_num, frame = TRUE)
```




```
KM_2$centers
```

```
##   stearic acid sorbitol shikimic acid  ribitol pseudo uridine nicotinamide
## 1     172353.6 1680.032      382.2930 403.4542       1394.712     217.5227
## 2     289962.4 4239.760      469.6168 542.0666       1299.050     637.1195
##   myo-inositol  mannose  lyxitol heptadecanoic acid glutaric acid glutamic acid
## 1     8367.910 15398.69 1105.755           2695.191      88.42112      5481.488
## 2     8216.252 15771.91 1144.115           3987.812      84.54062     10275.749
##   citric acid behenic acid alpha-ketoglutarate
## 1    29588.69     630.8289            175.5881
## 2    28515.39     838.5168            192.8996
```

However, if we compare the within cluster sum of squares for the first and the second run (3 and 2 clusters, respectively), we can see that the analysis with 2 clusters is a less good fit (53.5%) than that of the 3 clusters (75.2%). Hence, I will be using the parameters selected by the k-means analysis ran with 3 clusters; these are stearic acid, nicotinamide, myo-inositol, mannose, heptadecanoic acid, and glutamic acid.

Support Vector Machine

Creating the Train and Test Data Sets

```
# Creating a training and test set from the data frame df_num.
# First I add the arm to the data set, so the split can be done based on these features:
df_num_sp <- cbind(df_num, Arm = data$Arm)
```

```
index <- sample.split(df_num_sp$Arm, SplitRatio = .7)
SVM_tr <- subset(df_num_sp, index == TRUE)
SVM_val <- subset(df_num_sp, index == FALSE)
# Splitting the training set into further training and test sets:
index_train <- sample.split(SVM_tr$Arm, SplitRatio = .5)
SVM_train <- subset(SVM_tr, index_train == TRUE)
SVM_test <- subset(SVM_tr, index_train == FALSE)
```

Due to that we are predicting a categorical variable, using numeric variables, a good approach is to fit a classification model. In this example, the arm is the predicted variables, whereas the parameters are the predictor variables. Now, that I have the parameters that show the best separation, I will train the SVM on the training set. Here I train the SVM to predict the Arm, based on the variables that showed significant difference between the two arms, and gave the best cluster separation: stearic acid, nicotinamide, myo-inositol, mannose, heptadecanoic acid, and glutamic acid.

```
# SVM model with linear kernel: raw values
svm_model_linear <- svm(as.factor(Arm)~
                          as.numeric(SVM_train$`stearic acid`) +
                          as.numeric(SVM_train$nicotinamide) +
                          as.numeric(SVM_train$`myo-inositol`) +
                          as.numeric(SVM_train$mannose) +
                          as.numeric(SVM_train$`heptadecanoic acid`) +
                          as.numeric(SVM_train$`glutamic acid`),
                        data = SVM_train, method = "C", kernel = "linear",
                        gamma = 1, cost = 2)
# Getting the mean of the correctly predicted arm on the train data set:
predict_tr_linear <- predict(svm_model_linear, SVM_train)
classification_accuracy_SVM_tr <- mean(predict_tr_linear == SVM_train$Arm)
classification_accuracy_SVM_tr
```

```
## [1] 0.9117647
```

```
# Getting the mean of the correctly predicted arm on the test data set:
predict_test_linear <- predict(svm_model_linear, SVM_test)
classification_accuracy_SVM_test <- mean(predict_test_linear == SVM_test$Arm)
classification_accuracy_SVM_test
```

```
## [1] 0.9117647
```

Next, I will apply SVM on the standardized data set. I will use the scale() function, that calculates the Z-score for each variable.

```
# Creating a standardized training and test set from the data frame df_num:
df_num_st <- as_tibble(scale(df_num))
df_num_st <- cbind(df_num_st, Arm = data$Arm)
index_st <- sample.split(df_num_st$Arm, SplitRatio = .7)
SVM_tr_st <- subset(df_num_st, index == TRUE)
SVM_val_st <- subset(df_num_st, index == FALSE)
# Splitting the standardized training set into further training and test sets:
index_train_st <- sample.split(SVM_tr_st$Arm, SplitRatio = .5)
SVM_train_st <- subset(SVM_tr_st, index_train_st == TRUE)
SVM_test_st <- subset(SVM_tr_st, index_train_st == FALSE)
```

```
# SVM model with linear kernel: standardized data
svm_model_linear_st <- svm(as.factor(Arm)~
                           as.numeric(SVM_train_st$`stearic acid`) +
                           as.numeric(SVM_train_st$nicotinamide) +
                           as.numeric(SVM_train_st$mannose) +
                           as.numeric(SVM_train_st$`myo-inositol`) +
                           as.numeric(SVM_train_st$`heptadecanoic acid`) +
                           as.numeric(SVM_train_st$`glutamic acid`),
                         data = SVM_train_st, method = "C", kernel = "linear",
                         gamma = 1, cost = 2)
# Getting the mean of the correctly predicted arm on the standardized train data set:
predict_tr_linear_st <- predict(svm_model_linear_st, SVM_train_st)
classification_accuracy_SVM_st_tr <- mean(predict_tr_linear_st == SVM_train_st$Arm)
classification_accuracy_SVM_st_tr
```

```
## [1] 0.8235294
```

```
# Getting the mean of the correctly predicted arm on the standardized test data set:
predict_test_linear_st <- predict(svm_model_linear_st, SVM_test_st)
classification_accuracy_SVM_st_test <- mean(predict_test_linear_st == SVM_test_st$Arm)
classification_accuracy_SVM_st_test
```

```
## [1] 0.8235294
```

Standardized data was also used to perform k-means clustering analysis, but this resulted in a low within cluster sum of squares values for k = 2 (~14%) and k = 4 (~27%). ## *k-NN model In this section, I will fit a k-NN model using a standardized data set

```
# Re-initializing the scaled data set used for the k-means calculation above
df_num_st <- as_tibble(scale(df_num))
df_num_st <- cbind(df_num_st, Arm = data$Arm)
df_num_st <- df_num_st %>% select(1, 6, 7, 8, 10, 12, 16)
index_st <- sample.split(df_num_st$Arm, SplitRatio = .7)
knn_tr <- subset(df_num_st, index == TRUE)
knn_valid <- subset(df_num_st, index == FALSE)
index_knn_tr <- sample.split(knn_tr$Arm, SplitRatio = .5)
knn_train <- subset(knn_tr, index_knn_tr == TRUE)
knn_test <- subset(knn_tr, index_knn_tr == FALSE)
pred_knn <- knn(knn_train[,-7], knn_test[,-7], knn_train[,7], k = 21)
# Next we validate the predicted labels with the actual labels:
CFM_knn <- table(pred_knn, knn_test[,7])
CFM_knn
```

```
##
## pred_knn             Group 1 - Score 0 Group 2 - Score 50
##   Group 1 - Score 0                 17                 13
##   Group 2 - Score 50                 0                  4
```

From the above table we can clearly see that the model did not do a very good job predicting samples from group 2. The overall accuracy can be seen below:

```
classification_accuracy_knn <- sum(diag(CFM_knn))/sum(CFM_knn)
classification_accuracy_knn
```

```
## [1] 0.6176471
```

Random Forest Model

Another classification model is the Random Forest (RF) model, that is an evolved form of the Decision Trees (DTs). In this section, I will build a RF model for the 15 selected parameters. I will use the data set converted into numbers df_num. First, I will bind the 'Arm' predicted variable to the data set, then I will transform the column titles, so these do not consist of any special characters or spaces:

```
# Binding the Arm variables to the df_num data set:
df_num_RMF <- cbind(df_num, Arm = data$Arm)
# Selecting parameters highlighted by the k-nearest model:
df_num_RMF <- df_num_RMF %>% select(1, 6, 7, 8, 10, 12, 16)
# Converting the parameter names, so these do not consist of space or any illegal characters:
df_colnames <- sub(' ', '_', colnames(df_num_RMF))
df_colnames <- sub('-', '_', df_colnames)
# Adding back the converted names to the column headers.
colnames(df_num_RMF) <- df_colnames
```

Next, I will create a new set of train and test sets:

```
# Creating a training and test set from the data frame df_num_RMF:
index_RMF <- sample.split(df_num_RMF$Arm, SplitRatio = .7)
RMF_tr <- subset(df_num_RMF, index == TRUE)
RMF_val <- subset(df_num_RMF, index == FALSE)
# Splitting the train set into further train and test sets:
index_train_RMF <- sample.split(RMF_tr$Arm, SplitRatio = .5)
RMF_train <- subset(RMF_tr, index_train == TRUE)
RMF_test <- subset(RMF_tr, index_train == FALSE)
```

Building the RMF model:

```
RMF <- randomForest(as.factor(Arm)~nicotinamide +
                      stearic_acid +
                      mannose +
                      heptadecanoic_acid +
                      glutamic_acid +
                      myo_inositol,
                    data = RMF_train)

# Now that we have built the RMF model, we use the predict function to get the
# predicted values for the data set:
predict_RMF_tr <- predict(RMF, RMF_train)
# Adding the predicted arms to the data set:
RMF_train$predict_RMF_tr <- predict_RMF_tr
# Compare the predictions and the actual values to see the accuracy of the model
# using the table() function (building a confusion matrix):
CFM_RMF_tr <- table(RMF_train$Arm, RMF_train$predict_RMF_tr)
CFM_RMF_tr
```

```
##
##                       Group 1 - Score 0 Group 2 - Score 50
##   Group 1 - Score 0                  17                   0
##   Group 2 - Score 50                  0                  17
```

```
# Calculating the accuracy of the testing data can be measured by adding together
# all the diagonal values, and dividing with the sum of all values:
classification_accuracy_RMF_tr <- sum(diag(CFM_RMF_tr)/sum(CFM_RMF_tr))
classification_accuracy_RMF_tr
```

```
## [1] 1
```

Testing the model on the test set:

```
predict_RMF_test <- predict(RMF, RMF_test)
RMF_test$predict_RMF_test <- predict_RMF_test
CFM_RMF_test <- table(RMF_test$Arm, RMF_test$predict_RMF_test)
CFM_RMF_test
```

```
##
##                       Group 1 - Score 0 Group 2 - Score 50
##   Group 1 - Score 0                  11                   6
##   Group 2 - Score 50                  2                  15
```

```
classification_accuracy_RMF_test <- sum(diag(CFM_RMF_test)/sum(CFM_RMF_test))
classification_accuracy_RMF_test
```

```
## [1] 0.7647059
```

A standardized data has also been tested for building an RMF model, however, this also led to a low overall accuracy.

Validation

SVM

Validating the SVM model (linear and radial):

```
# Non-standardized Values - Linear
predict_valid_linear <- predict(svm_model_linear, SVM_val)
classification_accuracy_SVM_val <- mean(predict_valid_linear == SVM_val$Arm)
classification_accuracy_SVM_val
```

```
## [1] 0.7352941
```

```
# Standardized Values - Linear
predict_valid_linear_st <- predict(svm_model_linear_st, SVM_val_st)
classification_accuracy_SVM_st_val <- mean(predict_valid_linear_st == SVM_val_st$Arm)
classification_accuracy_SVM_st_val
```

```
## [1] 0.6470588
```

k-NN

```
pred_knn_valid <- knn(knn_train[,-7], knn_valid[,-7], knn_train[,7], k = 15)
# Next we validate the predicted labels with the actual labels:
CFM_knn_valid <- table(pred_knn_valid, knn_valid[,7])
CFM_knn_valid
```

```
##
## pred_knn_valid       Group 1 - Score 0 Group 2 - Score 50
##   Group 1 - Score 0                 14                 10
##   Group 2 - Score 50                 0                  5
```

```
classification_accuracy_knn_valid <- sum(diag(CFM_knn_valid))/sum(CFM_knn_valid)
classification_accuracy_knn_valid
```

```
## [1] 0.6551724
```

RMF

Validating the RMF model:

```
predict_valid_RMF <- predict(RMF, RMF_val)
RMF_val$predict_valid_RMF <- predict_valid_RMF
CFM_RMF_val <- table(RMF_val$Arm, RMF_val$predict_valid_RMF)
CFM_RMF_val
```

```
##
##                      Group 1 - Score 0 Group 2 - Score 50
##   Group 1 - Score 0                 11                  3
##   Group 2 - Score 50                 6                  9
```

```
classification_accuracy_RMF_val <- sum(diag(CFM_RMF_val)/sum(CFM_RMF_val))
classification_accuracy_RMF_val
```

```
## [1] 0.6896552
```

Results and Conclusions

The project looked into the accuracy of different models in the prediction of major depressive disorder in 97 patients. Altogether more than 140 parameters were measured from blood plasma, followed by the testing for significant difference of the mean and correlation analysis between two arms. From this analysis, 15 parameters were selected and k-means clustering was performed. This latter clustering analysis revealed 6 parameters that look promising for the potential differentiation of the two arms. The 6 selected parameters were used to fit support vector machine, k-NN and random forest models. The results are summarized below:

```
model_results_tibble <- tibble(Models = c("SVM Linear",
                                          "SVM Standardized Linear",
                                          "RMF", "k-NN"),
                               TrainFit = c(classification_accuracy_SVM_tr,
                                            classification_accuracy_SVM_st_tr,
                                            classification_accuracy_RMF_tr,
                                            '-'),
                               TestFit = c(classification_accuracy_SVM_test,
                                           classification_accuracy_SVM_st_test,
                                           classification_accuracy_RMF_test,
                                           classification_accuracy_knn),
                               Validation = c(classification_accuracy_SVM_val,
                                              classification_accuracy_SVM_st_val,
                                              classification_accuracy_RMF_val,
                                              classification_accuracy_knn_valid)) %>%
  mutate(TestFit = sprintf("%0.4f", TestFit))

model_results_tibble
```

```
## # A tibble: 4 x 4
##   Models                   TrainFit          TestFit Validation
##   <chr>                    <chr>             <chr>        <dbl>
## 1 SVM Linear               0.911764705882353 0.9118       0.735
## 2 SVM Standardized Linear  0.823529411764706 0.8235       0.647
## 3 RMF                      1                 0.7647       0.690
## 4 k-NN                     -                 0.6176       0.655
```

From the above results we can clearly see, that the lowest accuracy was produced by the k-NN model. SVM was fit on raw data as well as standardized values. These resulted in very similar accuracy values, around 70%. SVM was also fit with radial kernel, but this resulted in about 50% accuracy. The k-NN model was fit with k values ranging from 3 to 21 and produced one of the lowest overall accuracy.

Future Perspectives

A possible cause of the low accuracy of the different models could be that there is no real relationship in the measured parameters between the two arms, resulting in a high error rate in differentiating between healthy and depressed patients. To be able to separate the two groups based on these biochemical markers, a higher number of participants would be necessary. This latter could potentially result in a better separation between the arms. Another approach would be to identify new bio marker groups which are specific to patients diagnosed with MDD. Furthermore, additional factors, such as the BDI score, or lifestyle parameters could help fine tuning the models, leading to higher accuracy rates.

References

Sharma, Horrick, Soumava Santra, and Aloke Dutta. 2015. "Triple Reuptake Inhibitors as Potential Next-Generation Antidepressants: A New Hope?" *Future Medicinal Chemistry* 7 (17): 2385–2406.

Szoke-Kovacs, Zsombor, CSABA More, RITA Szoke-Kovacs, ENDRE Mathe, and Ede Frecska. 2020. "Selective Inhibition of the Serotonin Transporter in the Treatment of Depression: Sertraline, Fluoxetine and Citalopram." *Neuropsychopharmacologia Hungarica: A Magyar. Pszichofarmakologiai Egyesulet Lapja* 22 (1): 4–15.