Class Project 2 – Clustering and Predictive Analysis

CSCI 6444 Introduction to big data and analytics

Professor: Stephen Kaisler

Group 2
Shaival Vora(G37099269)
Nithin Raghava Aitha(G43935136)

TASK 1

Analyze the dataset

The first step in the project is to download and import the dataset in R Studio. The dataset we are using for this project is “ObesityDataSet_raw_and_data_sinthetic.csv”.

Variable Name	Role	Type	Demographic	Description	Units	Missing Values
Gender	Feature	Categorical	Gender			no
Age	Feature	Continuous	Age			no
Height	Feature	Continuous				no
Weight	Feature	Continuous				no
family_history_with_overweight	Feature	Binary		Has a family member suffered or suffers from overweight?		no
FAVC	Feature	Binary		Do you eat high caloric food frequently?		no
FCVC	Feature	Integer		Do you usually eat vegetables in your meals?		no
NCP	Feature	Continuous		How many main meals do you have daily?		no
CAEC	Feature	Categorical		Do you eat any food between meals?		no
SMOKE	Feature	Binary		Do you smoke?		no
CH2O	Feature	Continuous		How much water do you drink daily?		no
SCC	Feature	Binary		Do you monitor the calories you eat daily?		no
FAF	Feature	Continuous		How often do you have physical activity?		no
TUE	Feature	Integer		How much time do you use technological devices such as cell phone, videogames, television, computer and others?		no
CALC	Feature	Categorical		How often do you drink alcohol?		no
MTRANS	Feature	Categorical		Which transportation do you usually use?		no
NObeyesdad	Target	Categorical		Obesity level		no

```
#########################################################################################################################################
# Task 1
# Loading Data set  and exploration
# For this data set, plot the data using pairwise plotting to get a sense of the relationships between the attributes.
#
#########################################################################################################################################

# Read the Data set
clusteringDataset <- read.csv("/Users/shaivalvora/CSCI_6444_BigData/Project2/ObesityDataSet_raw_and_data_sinthetic.csv")
clusteringDataset[1:10,]
```

Load All the necessary Library required for this Project

```
install.packages("glmnet")
library(dplyr)
library(ggplot2)
library(readr)
library(tidyr)
library(caret)
library(cluster)
library(corrplot)
library(glmnet)
library(rstatix)
library(gmodels)
library(psych)
library(nnet)
```

Assigning Variable names to the vector for this we will assign the names to a variable . Check the length and assign to names(clusteringDataset)

```
> clusteringDataset.names = c("Gender", "Age", "Height", "Weight","family_history_with_overweight","FAVC","FCVC","NCP","CAE
C","SMOKE","CH2O","SCC","FAF","TUE","CALC","MTRANS","NObeyesdad")
> names(clusteringDataset) <- clusteringDataset.names
```

The str() function is used to display the internal structure of R objects.

```
> # Get the structure of the date set
> str(clusteringDataset)
'data.frame':   2111 obs. of  17 variables:
 $ Gender                        : num  1 1 2 2 2 2 1 2 2 2 ...
 $ Age                           : num  21 21 23 27 22 29 23 22 24 22 ...
 $ Height                        : num  1.62 1.52 1.8 1.8 1.78 1.62 1.5 1.64 1.78 1.72 ...
 $ Weight                        : num  64 56 77 87 89.8 53 55 53 64 68 ...
 $ family_history_with_overweight: num  2 2 2 1 1 1 2 1 2 2 ...
 $ FAVC                          : num  1 1 1 1 1 2 2 1 2 2 ...
 $ FCVC                          : num  2 3 2 3 2 2 3 2 3 2 ...
 $ NCP                           : num  3 3 3 3 1 3 3 3 3 3 ...
 $ CAEC                          : num  4 4 4 4 4 4 4 4 4 4 ...
 $ SMOKE                         : num  1 2 1 1 1 1 1 1 1 1 ...
 $ CH2O                          : num  2 3 2 2 2 2 2 2 2 2 ...
 $ SCC                           : num  1 2 1 1 1 1 1 1 1 1 ...
 $ FAF                           : num  0 3 2 2 0 0 1 3 1 1 ...
 $ TUE                           : num  1 0 1 0 0 0 0 0 1 1 ...
 $ CALC                          : num  3 4 2 2 4 4 4 4 2 3 ...
 $ MTRANS                        : num  4 4 4 5 4 1 3 4 4 4 ...
 $ NObeyesdad                    : num  2 2 2 6 7 2 2 2 2 2 ...
>
```

Identify all the non numeric data and we have to convert the data into numeric values

```
> # Identifying all the  non numeric columns
> non_numeric_columns = sapply(clusteringDataset, Negate(is.numeric))
> non_numeric_columns
                      Gender                          Age                       Height
                       FALSE                        FALSE                        FALSE
                      Weight family_history_with_overweight                         FAVC
                       FALSE                        FALSE                        FALSE
                        FCVC                          NCP                         CAEC
                       FALSE                        FALSE                        FALSE
                       SMOKE                         CH2O                          SCC
                       FALSE                        FALSE                        FALSE
                         FAF                          TUE                         CALC
                       FALSE                        FALSE                        FALSE
                      MTRANS                    NObeyesdad
                       FALSE                        FALSE
>
```

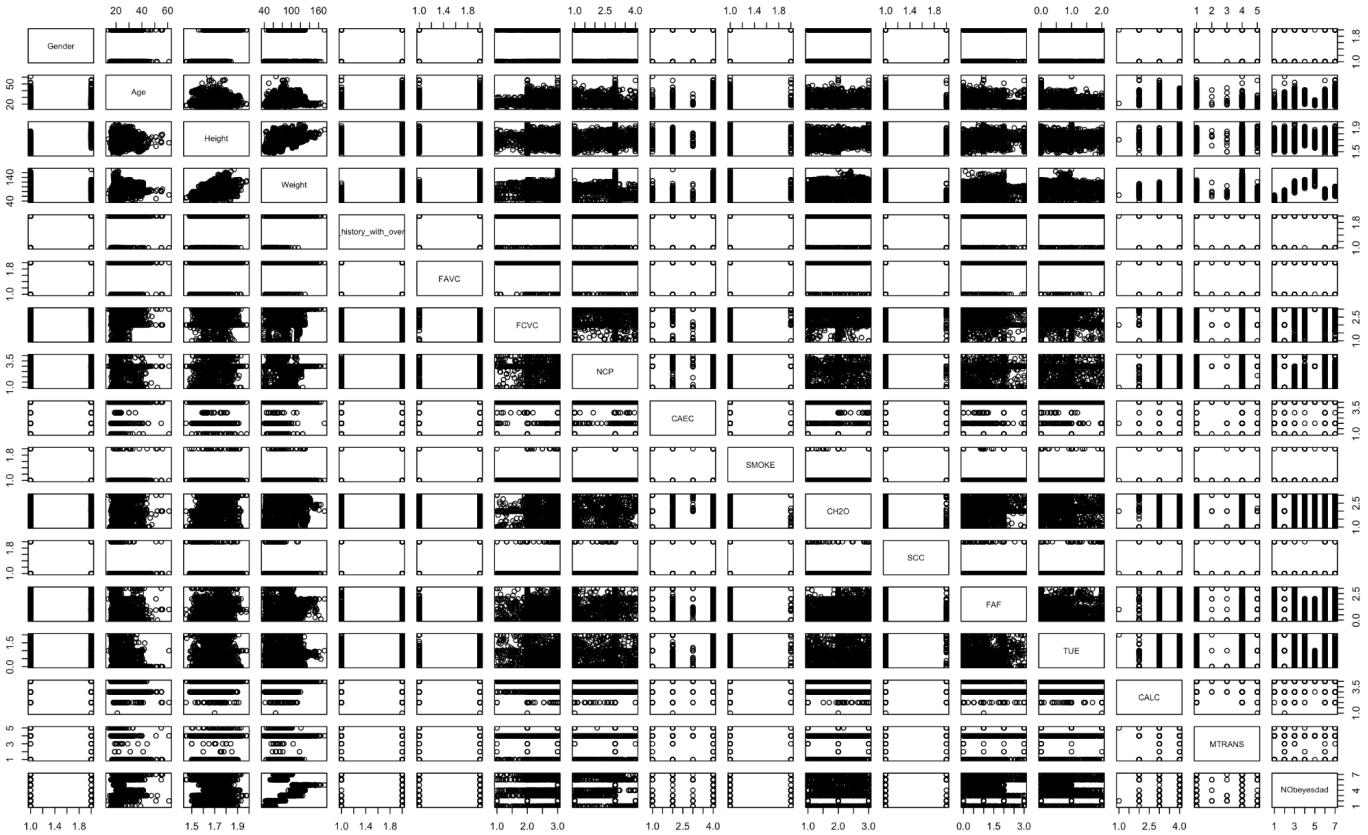
we need to convert non-numeric columns into numeric format. The function as.numeric() is used for this purpose, transforming character data into numeric data.

lapply(): It is a function in R that applies another function to each element of a list (or vector) and returns a list of the same length as the input list, where each element of the output list is the result of applying the specified function to the corresponding element of the input list.

```
> # Convert all the non numeric data into a numeric data
> clusteringDataset[non_numeric_columns] = lapply(clusteringDataset[non_numeric_columns], function(x) as.numeric(as.factor(x)))
> clusteringDataset
   Gender Age Height Weight family_history_with_overweight FAVC FCVC NCP CAEC SMOKE CH2O SCC FAF TUE CALC MTRANS NObeyesdad
1       1  21   1.62   64.0                               2    1    2   3    4     1    2   1   0   1    3      4          2
2       1  21   1.52   56.0                               2    1    3   3    4     2    3   2   3   0    4      4          2
3       2  23   1.80   77.0                               2    1    2   3    4     1    2   1   2   1    2      4          2
4       2  27   1.80   87.0                               1    1    3   3    4     1    2   1   2   0    2      5          6
5       2  22   1.78   89.8                               1    1    2   1    4     1    2   1   0   0    4      4          7
6       2  29   1.62   53.0                               1    2    2   3    4     1    2   1   0   0    4      1          2
7       1  23   1.50   55.0                               2    2    3   3    4     1    2   1   1   0    4      3          2
8       2  22   1.64   53.0                               1    1    2   3    4     1    2   1   3   0    4      4          2
9       2  24   1.78   64.0                               2    2    3   3    4     1    2   1   1   1    2      4          2
10      2  22   1.72   68.0                               2    2    2   3    4     1    2   1   1   1    3      4          2
11      2  26   1.85  105.0                               2    2    3   3    2     1    3   1   2   2    4      4          3
12      1  21   1.72   80.0                               2    2    2   3    2     1    2   2   2   1    4      4          7
13      2  22   1.65   56.0                               1    1    3   3    4     1    3   1   2   0    4      4          2
14      2  41   1.80   99.0                               1    2    2   3    4     1    2   1   2   1    2      1          3
15      2  23   1.77   60.0                               2    2    3   1    4     1    1   1   1   1    4      4          2
16      1  22   1.70   66.0                               2    1    3   3    1     1    2   2   2   1    4      4          2
17      2  27   1.93  102.0                               2    2    2   1    4     1    1   1   1   0    4      4          7
18      1  29   1.53   78.0                               1    2    2   1    4     1    2   1   0   0    3      1          3
19      1  30   1.71   82.0                               2    2    3   4    2     2    1   1   0   0    3      1          7
20      1  23   1.65   70.0                               2    1    2   1    4     1    2   1   0   0    4      4          6
21      2  22   1.65   80.0                               2    1    2   3    4     1    2   1   3   2    3      5          7
22      1  52   1.69   87.0                               2    2    3   1    4     2    2   1   0   0    3      1          3
23      1  22   1.65   60.0                               2    2    3   3    4     1    2   1   1   0    4      1          2
24      1  22   1.60   82.0                               2    2    1   1    4     1    2   1   0   2    4      4          3
25      2  21   1.85   68.0                               2    2    2   3    4     1    2   1   0   1    4      4          2
26      2  20   1.60   50.0                               2    1    2   4    2     2    2   1   3   2    3      4          2
27      2  21   1.70   65.0                               2    2    2   1    2     1    2   1   1   2    1      5          2
28      1  23   1.60   52.0                               1    2    2   4    2     1    2   1   2   1    4      1          2
29      2  19   1.75   76.0                               2    2    3   3    4     1    2   2   3   1    4      4          2
30      2  23   1.68   70.0                               1    2    2   3    4     1    2   1   2   2    2      5          2
31      2  29   1.77   83.0                               1    2    1   4    2     1    3   1   0   1    3      3          6
32      1  31   1.58   68.0                               2    1    2   1    4     1    1   1   1   0    4      4          7
33      1  24   1.77   76.0                               1    1    2   3    4     1    3   1   1   1    4      5          2
34      2  39   1.79   90.0                               1    1    2   1    4     1    2   1   0   0    4      4          7
35      2  22   1.65   62.0                               1    2    2   4    2     1    2   1   2   0    4      4          2
36      1  21   1.50   65.0                               2    1    2   3    4     1    2   1   2   2    4      4          7
37      1  22   1.56   49.0                               1    2    2   3    4     1    1   2   2   0    3      5          2
38      1  21   1.60   48.0                               1    2    2   3    4     1    1   1   1   0    4      4          2
39      2  23   1.65   67.0                               2    2    2   3    4     1    2   1   1   1    4      4          2
40      1  21   1.75   88.0                               2    2    2   3    4     1    3   1   3   0    4      4          7
41      1  21   1.67   75.0                               2    2    2   3    4     1    2   1   1   0    4      4          6
42      2  23   1.68   60.0                               1    1    2   4    2     1    2   1   0   0    3      5          2
```

Use Pairwise plotting to understand the relationship between the attributes of the obesity dataset

```
9
0  # Do a pair wise plotting to get the relation between all the types of data
1  plot(clusteringDataset)
2  pairs(clusteringDataset)
3
```



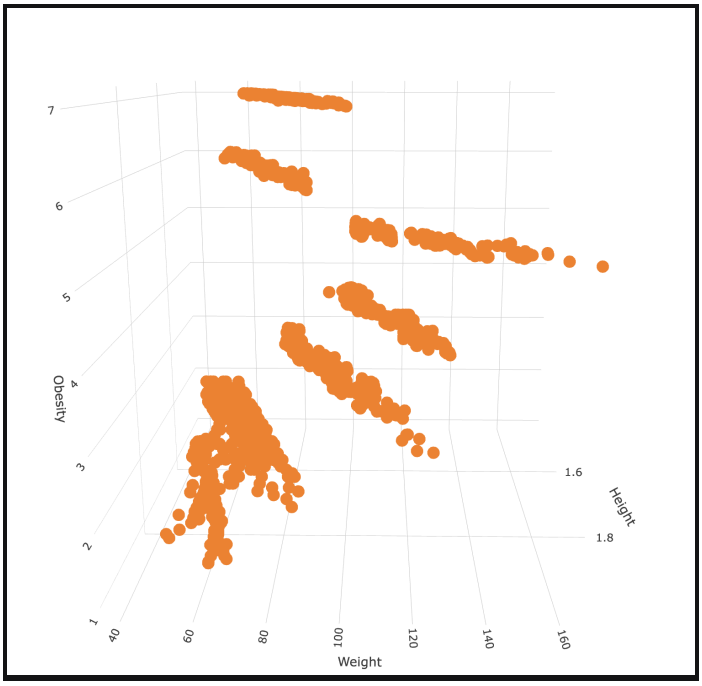
By examining the pair plot of the clusteringDataset, we can gain valuable insights into the relationships between various attributes related to obesity

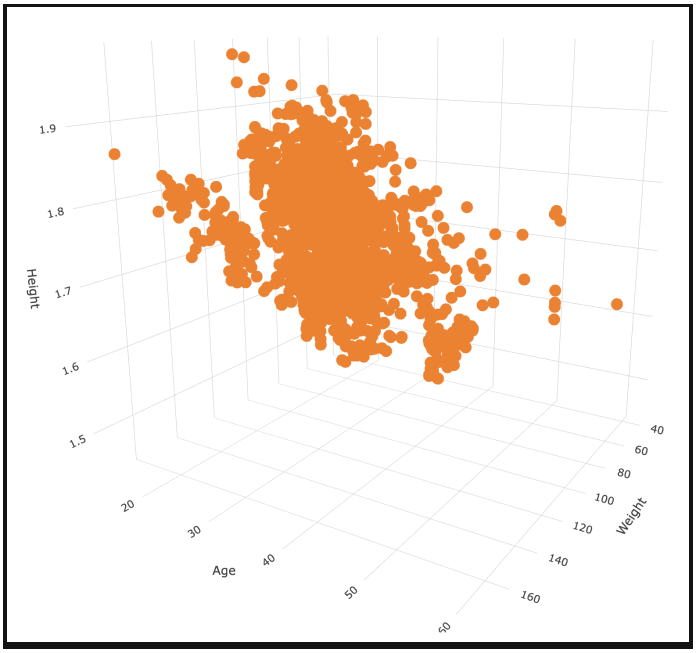
We can identify potential linear relationships between attributes by looking for diagonal lines in the scatter plots such as Age vs. NObeyesdad and Height vs. NObeyesdad. The absence of a diagonal pattern suggests a non-linear relationship such as Height vs. Weight and Age vs. Weight

3D Plotting

3d Plotting of the data using three parameters Height, Weight and NObeyesdad to get more insight into the relationship between attributes

```
> plot3D = plot_ly(clusteringDataset,
+                  x = ~Height,
+                  y = ~Weight,
+                  z = ~NObeyesdad,
+                  type = "scatter3d",
+                  mode = "markers")
> plot3D = plot3D %>% add_markers()
> plot3D = plot3D %>%  layout(scene = list(xaxis = list(title = 'Height'),
+                                          yaxis = list(title = 'Weight'),
+                                          zaxis = list(title = 'Obesity')))
> plot3D
```





TASK 2

Prepare the data set

str(): After the Numeric conversion we will again see the structure of the data set

```
> # Get the summery and description of the data set and check the structure of the data
> str(clusteringDataset)
'data.frame':   2111 obs. of  17 variables:
 $ Gender                        : num  1 1 2 2 2 2 1 2 2 2 ...
 $ Age                           : num  21 21 23 27 22 29 23 22 24 22 ...
 $ Height                        : num  1.62 1.52 1.8 1.8 1.78 1.62 1.5 1.64 1.78 1.72 ...
 $ Weight                        : num  64 56 77 87 89.8 53 55 53 64 68 ...
 $ family_history_with_overweight: num  2 2 2 1 1 1 2 1 2 2 ...
 $ FAVC                          : num  1 1 1 1 1 2 2 1 2 2 ...
 $ FCVC                          : num  2 3 2 3 2 2 3 2 3 2 ...
 $ NCP                           : num  3 3 3 3 1 3 3 3 3 3 ...
 $ CAEC                          : num  4 4 4 4 4 4 4 4 4 4 ...
 $ SMOKE                         : num  1 2 1 1 1 1 1 1 1 1 ...
 $ CH2O                          : num  2 3 2 2 2 2 2 2 2 2 ...
 $ SCC                           : num  1 2 1 1 1 1 1 1 1 1 ...
 $ FAF                           : num  0 3 2 2 0 0 1 3 1 1 ...
 $ TUE                           : num  1 0 1 0 0 0 0 0 1 1 ...
 $ CALC                          : num  3 4 2 2 4 4 4 4 2 3 ...
 $ MTRANS                        : num  4 4 4 5 4 1 3 4 4 4 ...
 $ NObeyesdad                    : num  2 2 2 6 7 2 2 2 2 2 ...
> 
```

summery(): This function provides a concise summary of each variable in the dataset. For numeric variables, it will typically show the minimum, maximum, mean, median, and quartiles (like 1st quartile and 3rd quartile).

```
> summary(clusteringDataset)
     Gender           Age             Height          Weight       family_history_with_overweight      FAVC      
 Min.   :1.000   Min.   :14.00   Min.   :1.450   Min.   : 39.00   Min.   :1.000                  Min.   :1.000  
 1st Qu.:1.000   1st Qu.:19.95   1st Qu.:1.630   1st Qu.: 65.47   1st Qu.:2.000                  1st Qu.:2.000  
 Median :2.000   Median :22.78   Median :1.700   Median : 83.00   Median :2.000                  Median :2.000  
 Mean   :1.506   Mean   :24.31   Mean   :1.702   Mean   : 86.59   Mean   :1.818                  Mean   :1.884  
 3rd Qu.:2.000   3rd Qu.:26.00   3rd Qu.:1.768   3rd Qu.:107.43   3rd Qu.:2.000                  3rd Qu.:2.000  
 Max.   :2.000   Max.   :61.00   Max.   :1.980   Max.   :173.00   Max.   :2.000                  Max.   :2.000  
      FCVC            NCP             CAEC           SMOKE            CH2O            SCC             FAF       
 Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.   :0.0000  
 1st Qu.:2.000   1st Qu.:2.659   1st Qu.:4.000   1st Qu.:1.000   1st Qu.:1.585   1st Qu.:1.000   1st Qu.:0.1245  
 Median :2.386   Median :3.000   Median :4.000   Median :1.000   Median :2.000   Median :1.000   Median :1.0000  
 Mean   :2.419   Mean   :2.686   Mean   :3.671   Mean   :1.021   Mean   :2.008   Mean   :1.045   Mean   :1.0103  
 3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:4.000   3rd Qu.:1.000   3rd Qu.:2.477   3rd Qu.:1.000   3rd Qu.:1.6667  
 Max.   :3.000   Max.   :4.000   Max.   :4.000   Max.   :2.000   Max.   :3.000   Max.   :2.000   Max.   :3.0000  
      TUE             CALC           MTRANS        NObeyesdad  
 Min.   :0.0000   Min.   :1.00   Min.   :1.000   Min.   :1.000  
 1st Qu.:0.0000   1st Qu.:3.00   1st Qu.:4.000   1st Qu.:2.000  
 Median :0.6253   Median :4.00   Median :4.000   Median :4.000  
 Mean   :0.6579   Mean   :3.63   Mean   :3.365   Mean   :4.016  
 3rd Qu.:1.0000   3rd Qu.:4.00   3rd Qu.:4.000   3rd Qu.:6.000  
 Max.   :2.0000   Max.   :4.00   Max.   :5.000   Max.   :7.000  
> 
```

describe(): This function is similar to summary and it will give some additional statistics like standard deviation or skewness for numeric variables.

```
> describe(clusteringDataset)
                               vars    n  mean    sd median trimmed   mad   min    max  range  skew kurtosis   se
Gender                            1 2111  1.51  0.50   2.00    1.51  0.00  1.00   2.00   1.00 -0.02    -2.00 0.01
Age                               2 2111 24.31  6.35  22.78   23.34  4.78 14.00  61.00  47.00  1.53     2.81 0.14
Height                            3 2111  1.70  0.09   1.70    1.70  0.10  1.45   1.98   0.53 -0.01    -0.57 0.00
Weight                            4 2111 86.59 26.19  83.00   85.82 32.22 39.00 173.00 134.00  0.26    -0.70 0.57
family_history_with_overweight    5 2111  1.82  0.39   2.00    1.90  0.00  1.00   2.00   1.00 -1.64     0.70 0.01
FAVC                              6 2111  1.88  0.32   2.00    1.98  0.00  1.00   2.00   1.00 -2.40     3.74 0.01
FCVC                              7 2111  2.42  0.53   2.39    2.46  0.57  1.00   3.00   2.00 -0.43    -0.64 0.01
NCP                               8 2111  2.69  0.78   3.00    2.77  0.00  1.00   4.00   3.00 -1.11     0.38 0.02
CAEC                              9 2111  3.67  0.78   4.00    3.87  0.00  1.00   4.00   3.00 -2.13     3.06 0.02
SMOKE                            10 2111  1.02  0.14   1.00    1.00  0.00  1.00   2.00   1.00  6.70    42.95 0.00
CH2O                             11 2111  2.01  0.61   2.00    2.01  0.67  1.00   3.00   2.00 -0.10    -0.88 0.01
SCC                              12 2111  1.05  0.21   1.00    1.00  0.00  1.00   2.00   1.00  4.36    17.02 0.00
FAF                              13 2111  1.01  0.85   1.00    0.94  1.19  0.00   3.00   3.00  0.50    -0.62 0.02
TUE                              14 2111  0.66  0.61   0.63    0.59  0.72  0.00   2.00   2.00  0.62    -0.55 0.01
CALC                             15 2111  3.63  0.55   4.00    3.70  0.00  1.00   4.00   3.00 -1.17     0.46 0.01
MTRANS                           16 2111  3.37  1.26   4.00    3.55  0.00  1.00   5.00   4.00 -1.28    -0.20 0.03
NObeyesdad                       17 2111  4.02  1.95   4.00    4.02  2.97  1.00   7.00   6.00  0.01    -1.19 0.04
>
```

head(): This function displays the first few rows of the dataset.

```
> head(clusteringDataset)
  Gender Age Height Weight family_history_with_overweight FAVC FCVC NCP CAEC SMOKE CH2O SCC FAF TUE CALC MTRANS
1      1  21   1.62   64.0                               2    1    2   3    4     1    2   1   0   1    3      4
2      1  21   1.52   56.0                               2    1    3   3    4     2    3   2   3   0    4      4
3      2  23   1.80   77.0                               2    1    2   3    4     1    2   1   2   1    2      4
4      2  27   1.80   87.0                               1    1    3   3    4     1    2   1   2   0    2      5
5      2  22   1.78   89.8                               1    1    2   1    4     1    2   1   0   0    4      4
6      2  29   1.62   53.0                               1    2    2   3    4     1    2   1   0   0    4      1
  NObeyesdad
1          2
2          2
3          2
4          6
5          7
6          2
>
```

Normalize Function

In the Normalization step we will keep the response variable NObeyesdad intact, this means we will not convert this column into numeric values. For this we will use

```
6          2
> # Normalize features, keeping the response variable "NObeyesdad" intact
> features <- setdiff(names(clusteringDataset), "NObeyesdad")
> clusteringDataset.features <- clusteringDataset[features]
>
```

Here the setdiff function in R takes two sets and returns the difference between them. the first set is all column names (names(clusteringDataset)) and the second set is just the name "NObeyesdad".By subtracting "NObeyesdad", you end up with a vector named features containing the names of all other columns in the dataset.

Normalize part: The formula ((x-min(x))/(max(x)-min(x))) subtracts the minimum value of the vector from each element and divides by the range (max-min), ensuring all values fall within 0 and 1.

By normalizing features, we can ensure they contribute more meaningfully to the clustering process and prevent any single feature from dominating due to its scale.

```
> # Scaling with normalize: The lapply() method applies to the entire data frame and all selected columns
> # Normalization function.
> normalize_function <- function(x) {((x-min(x))/(max(x)-min(x)))}
> clusteringDataset.features.norm <- as.data.frame(lapply(clusteringDataset.features, normalize_function))
> clusteringDataset.norm <- cbind(clusteringDataset.features.norm, NObeyesdad = clusteringDataset$NObeyesdad)
>
```

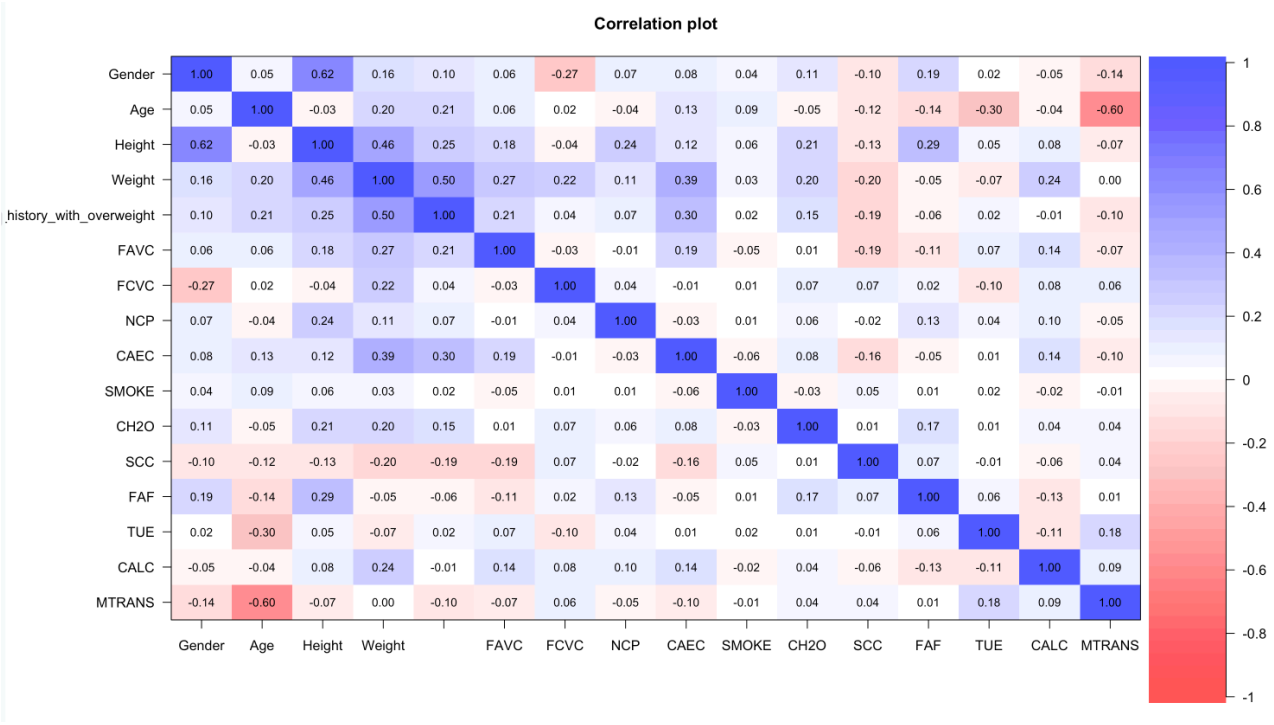
First 10 rows of the normalized data

```
> clusteringDataset.norm[1:10,]
   Gender       Age     Height    Weight family_history_with_overweight FAVC FCVC       NCP CAEC SMOKE CH2O SCC       FAF TUE       CALC MTRANS NObeyesdad
1       0 0.1489362 0.32075472 0.1865672                              1    0  0.5 0.6666667    1     0  0.5   0 0.0000000 0.5 0.6666667   0.75          2
2       0 0.1489362 0.13207547 0.1268657                              1    0  1.0 0.6666667    1     1  1.0   1 1.0000000 0.0 1.0000000   0.75          2
3       1 0.1914894 0.66037736 0.2835821                              1    0  0.5 0.6666667    1     0  0.5   0 0.6666667 0.5 0.3333333   0.75          2
4       1 0.2765957 0.66037736 0.3582090                              0    0  1.0 0.6666667    1     0  0.5   0 0.6666667 0.0 0.3333333   1.00          6
5       1 0.1702128 0.62264151 0.3791045                              0    0  0.5 0.0000000    1     0  0.5   0 0.0000000 0.0 1.0000000   0.75          7
6       1 0.3191489 0.32075472 0.1044776                              0    1  0.5 0.6666667    1     0  0.5   0 0.0000000 0.0 1.0000000   0.00          2
7       0 0.1914894 0.09433962 0.1194030                              1    1  1.0 0.6666667    1     0  0.5   0 0.3333333 0.0 1.0000000   0.50          2
8       1 0.1702128 0.35849057 0.1044776                              0    0  0.5 0.6666667    1     0  0.5   0 1.0000000 0.0 1.0000000   0.75          2
9       1 0.2127660 0.62264151 0.1865672                              1    1  1.0 0.6666667    1     0  0.5   0 0.3333333 0.5 0.3333333   0.75          2
10      1 0.1702128 0.50943396 0.2164179                              1    1  0.5 0.6666667    1     0  0.5   0 0.3333333 0.5 0.6666667   0.75          2
>
```

Correlation

The correlation matrix and its heatmap visualization can help you understand the linear relationships between the normalized features. Strong positive correlations (values close to 1) suggest features tend to move together, while strong negative correlations (values close to -1) indicate they move in opposite directions. Weak correlations (values close to 0) suggest little to no linear relationship.

```
83  # A graphical display of a correlation matrix, confidence interval
84  correlation = cor(clusteringDataset.features.norm)
85  corelation
86
87  # Plot the correlation
88  cor.plot(correlation)
89
```



Split Data set

Split the given obesity data set into Training and Test Sets and here we will use (50-50%, 60-40%, 70-30%) split of the data

Splitting data allows us to evaluate the performance of our clustering model on unseen data (the testing set), ensuring it generalizes well to new data and avoids overfitting.

Splitting the data into a 50-50% split between training and testing data

```
> # Split the data set into Training and Test Sets (70-30%, 60-40%, 50-50%)
> set.seed(110)  # Ensures reproducibility
> # For 50-50 Split in the data set
> splitRatio50_50 <- 0.5
> clusteringDataset.norm.rows.50_50 <- nrow(clusteringDataset.norm)
> clusteringDataset.rows.50_50 <- round(splitRatio50_50 * clusteringDataset.norm.rows.50_50)
> clusteringDataset.train.index.50_50 <- sample(clusteringDataset.norm.rows.50_50, clusteringDataset.rows.50_50)
> clusteringDataset.tain.50_50 <- clusteringDataset.norm[clusteringDataset.train.index.50_50, ]
> clusteringDataset.test.50_50 <- clusteringDataset.norm[-clusteringDataset.train.index.50_50, ]
> cat("For the 50-50 Split Ratio: Training data set will have", nrow(clusteringDataset.tain.50_50), "rows. Testing data will have", nrow(clusteringDataset.test.50_50)
"rows.\n")
For the 50-50 Split Ratio: Training data set will have 1056 rows. Testing data will have 1055 rows.
```

- set.seed(110): This line sets a random seed to control the output of random number generators in R.
- splitRatio50_50 <- 0.5: This line specifies that you want to divide the clusteringDataset.norm into a 50-50 split, meaning 50% of the data will be used for training and 50% for testing.

- clusteringDataset.norm.rows.50_50 <- nrow(clusteringDataset.norm): This line gets the total number of rows in the clusteringDataset.norm data frame.

- clusteringDataset.rows.50_50<-round(splitRatio50_50*clusteringDataset.norm.rows.50_50) This line calculates the number of rows that should belong to the training set based on the 50-50 split ratio.

- clusteringDataset.train.index.50_50<-sample(clusteringDataset.norm.rows.50_50, clusteringDataset.rows.50_50): This line randomly selects clusteringDataset.rows.50_50 row indices from the total number of rows in clusteringDataset.norm.

- Splitting Data into Training and Testing Sets:

clusteringDataset.tain.50_50<-clusteringDataset.norm[clusteringDataset.train.index.50_50,]

clusteringDataset.test.50_50<-clusteringDataset.norm[-clusteringDataset.train.index.50_50,]

- Printing Data Set Size Information:

Final split For the 50-50 Split Ratio: Training data set will have 1056 rows. Testing data will have 1055 rows.

Now Similarly we will do a 60-40% split and 70-30% split

- 60-40 % split in training and testing data

```
> # For 60-40 Split in the data set
There were 12 warnings (use warnings() to see them)
> splitRatio60_40 <- 0.6
> clusteringDataset.norm.rows.60_40 <- nrow(clusteringDataset.norm)
> clusteringDataset.rows.60_40 <- round(splitRatio60_40 * clusteringDataset.norm.rows.60_40)
> clusteringDataset.train.index.60_40 <- sample(clusteringDataset.norm.rows.60_40, clusteringDataset.rows.60_40)
> clusteringDataset.tain.60_40 <- clusteringDataset.norm[clusteringDataset.train.index.60_40, ]
> clusteringDataset.test.60_40 <- clusteringDataset.norm[-clusteringDataset.train.index.60_40, ]
> cat("For the 60-40 Split Ratio: Training data set will have", nrow(clusteringDataset.tain.60_40), "rows. Testing data will have", nrow(clusteringDataset.test.60_40),
"rows.\n")
For the 60-40 Split Ratio: Training data set will have 1267 rows. Testing data will have 844 rows.
>
```

- 70-30% split in training and testing data

```
> # For 70-30 Split in the data set
> splitRatio70_30 <- 0.7
> clusteringDataset.norm.rows.70_30 <- nrow(clusteringDataset.norm)
> clusteringDataset.rows.70_30 <- round(splitRatio70_30 * clusteringDataset.norm.rows.70_30)
> clusteringDataset.train.index.70_30 <- sample(clusteringDataset.norm.rows.70_30, clusteringDataset.rows.70_30)
> clusteringDataset.tain.70_30 <- clusteringDataset.norm[clusteringDataset.train.index.70_30, ]
> clusteringDataset.test.70_30 <- clusteringDataset.norm[-clusteringDataset.train.index.70_30, ]
> cat("For the 70_30 Split Ratio: Training data set will have", nrow(clusteringDataset.tain.70_30), "rows. Testing data will have", nrow(clusteringDataset.test.70_30),
"rows.\n")
For the 70_30 Split Ratio: Training data set will have 1478 rows. Testing data will have 633 rows.
>
```

Task 3

K means clustering on the whole data set

To perform K-means clustering on our data, we use kmeans(). We specify centers with different values to specify the number of clusters we expect. The resulting object contains the Clustering Vector, Cluster means and between_SS /total_SS (%) value. The collected findings for different centers are tabularized below:

With 3 Clusters

```
#Create 3 clusters
obesity.k3 = kmeans(clusteringDataset.norm, centers = 3)
str(obesity.k3)
obesity.k3
factoextra::fviz_cluster(obesity.k3, clusteringDataset.norm)
```

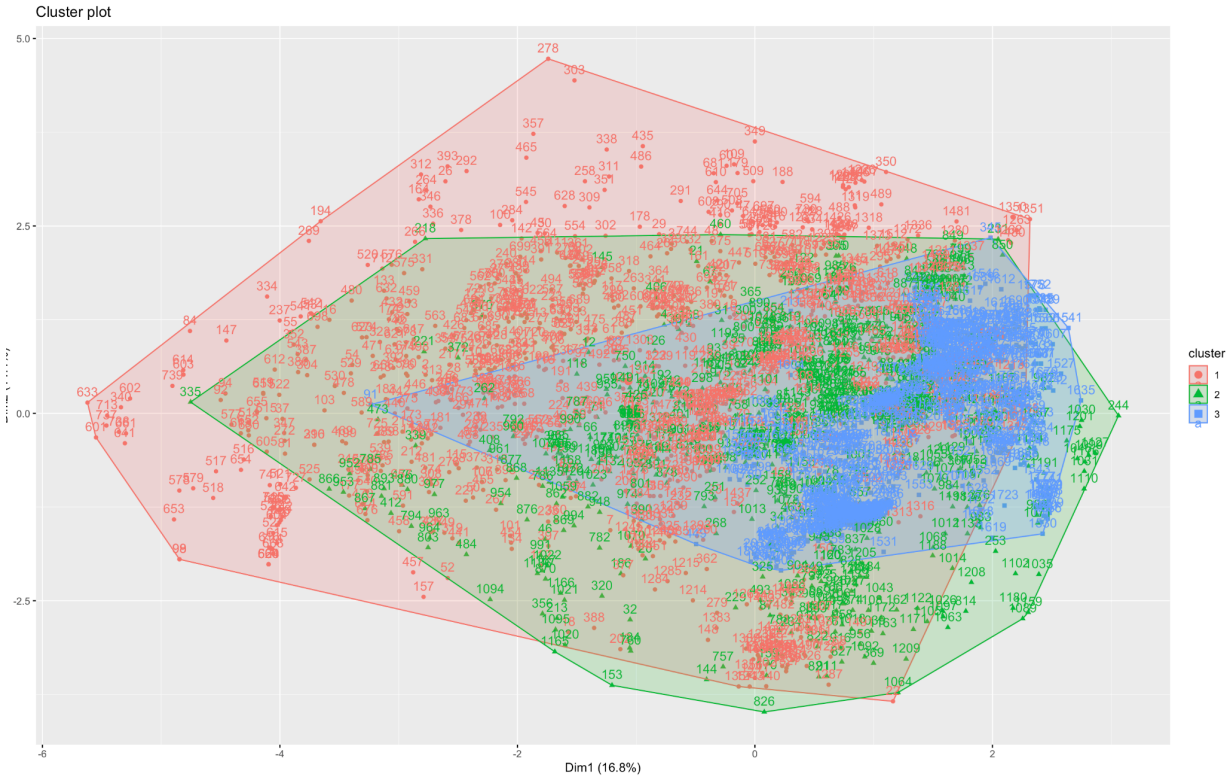
```
> #Create 3 clusters
> obesity.k3 = kmeans(clusteringDataset.norm, centers = 3)
> str(obesity.k3)
List of 9
 $ cluster     : int [1:2111] 1 1 1 2 2 1 1 1 1 1 ...
 $ centers     : num [1:3, 1:17] 0.484 0.572 0.477 0.186 0.238 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "1" "2" "3"
  .. ..$ : chr [1:17] "Gender" "Age" "Height" "Weight" ...
 $ totss       : num 10696
 $ withinss    : num [1:3] 1953 877 581
 $ tot.withinss: num 3410
 $ betweenss   : num 7286
 $ size        : int [1:3] 910 580 621
 $ iter        : int 2
 $ ifault      : int 0
 - attr(*, "class")= chr "kmeans"
```

```
> obesity.k3
K-means clustering with 3 clusters of sizes 910, 580, 621

Cluster means:
     Gender       Age    Height    Weight family_history_with_overweight      FAVC      FCVC       NCP
1 0.4835165 0.1862440 0.4482459 0.2338893                     0.6868132 0.8450549 0.6604824 0.5576195
2 0.5724138 0.2384538 0.4637586 0.2923586                     0.8293103 0.8344828 0.6313023 0.4999580
3 0.4766506 0.2502482 0.5242381 0.5913860                     0.9983897 0.9871176 0.8544374 0.6259435
       CAEC      SMOKE      CH2O         SCC       FAF       TUE      CALC    MTRANS NObeyesdad
1 0.7937729 0.02197802 0.4787786 0.059340659 0.3825584 0.3625657 0.8322344 0.5906593   2.086813
2 0.9310345 0.01379310 0.5209645 0.070689655 0.3358114 0.3275667 0.8574713 0.5469828   6.500000
3 0.9940955 0.02576490 0.5251337 0.001610306 0.2705541 0.2809243 0.9592056 0.6336554   4.521739

Clustering vector:
   [1] 1 1 1 2 2 1 1 1 1 1 1 2 1 1 1 1 2 1 2 2 2 1 1 1 1 1 1 1 1 2 2 1 2 1 2 1 1 1 2 2 1 1 1 1 2 1 1 1 1
  [51] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 3 1 2 1 1 1 2 1 1 2 1 1 1 2 1 1 2 1 1 2 1 2 3 1 2 1 1 1 1 1 1
 [101] 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 2 2 1 1 2 2 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 2 2 2 1 1 1 1
 [151] 1 1 2 2 1 1 1 1 2 2 1 2 1 1 2 3 1 2 2 2 2 1 1 1 2 1 1 1 1 2 1 1 1 2 2 2 1 1 1 1 1 2 2 1 1 2 1 3 1 1
 [201] 1 1 3 1 1 1 1 1 1 1 3 1 2 1 1 1 1 2 1 2 2 2 1 1 1 3 1 1 2 3 2 1 1 2 1 1 1 1 1 1 1 2 1 2 1 1 1 1 1 2
 [251] 2 2 2 1 1 2 2 1 2 1 2 2 1 1 1 1 1 2 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 1 2
 [301] 1 1 1 1 2 3 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 1 1 2 1 1 1 1 1 3 1 1 2 1 1
 [351] 1 1 1 1 1 2 1 1 3 1 1 1 2 1 2 1 2 2 2 2 1 2 1 1 1 2 1 1 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 3 1
 [401] 2 1 2 3 1 2 1 2 1 1 1 2 1 2 1 1 2 2 1 1 1 1 1 2 2 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 2 3 1
 [451] 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1 2 1 1 2 2 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 2 1 1 1 1 1 3 3
 [501] 3 3 3 3 3 3 3 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 1 1 1 1 1 1 1 1
 [551] 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [601] 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [651] 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 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [701] 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 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2
 [751] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [801] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [851] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [901] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [951] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [ reached getOption("max.print") -- omitted 1111 entries ]

Within cluster sum of squares by cluster:
[1] 1952.6656  876.7843  580.8454
 (between_SS / total_SS =  68.1 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> factoextra::fviz_cluster(obesity.k3, clusteringDataset.norm)
```

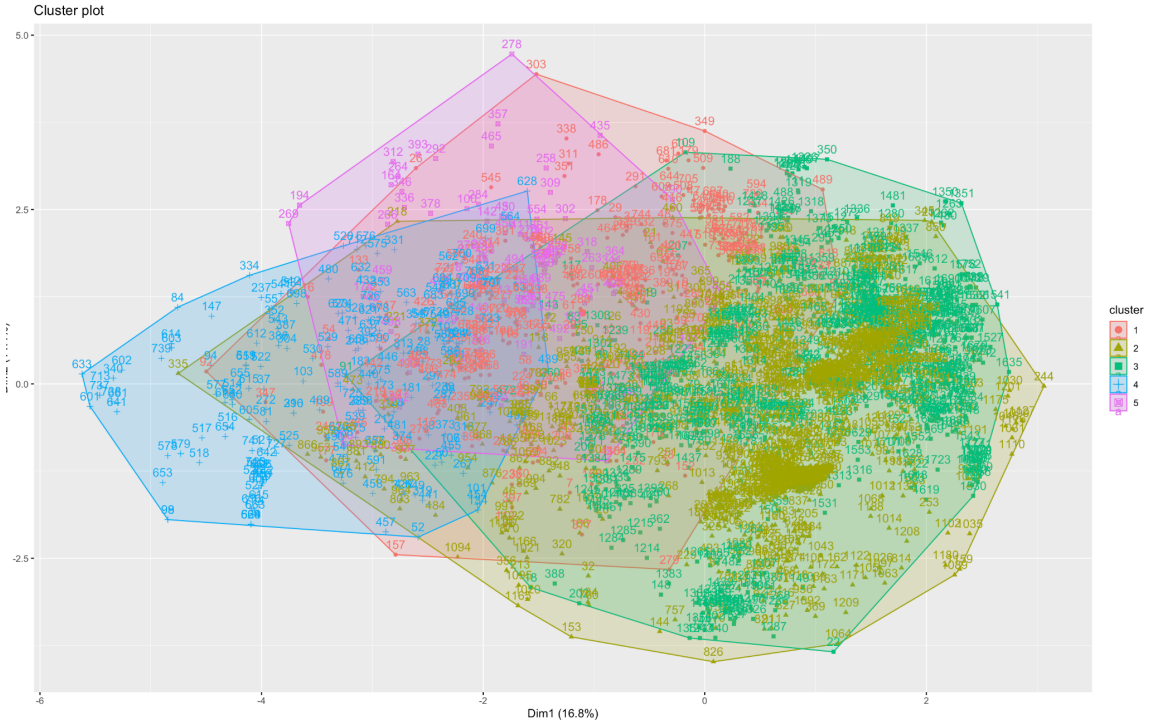
Centers (No. of clusters to expect)	between_SS /total_SS (%)
3	68.1
5	71.4
7	78.0
9	81.4
11	82.3

13	84.1

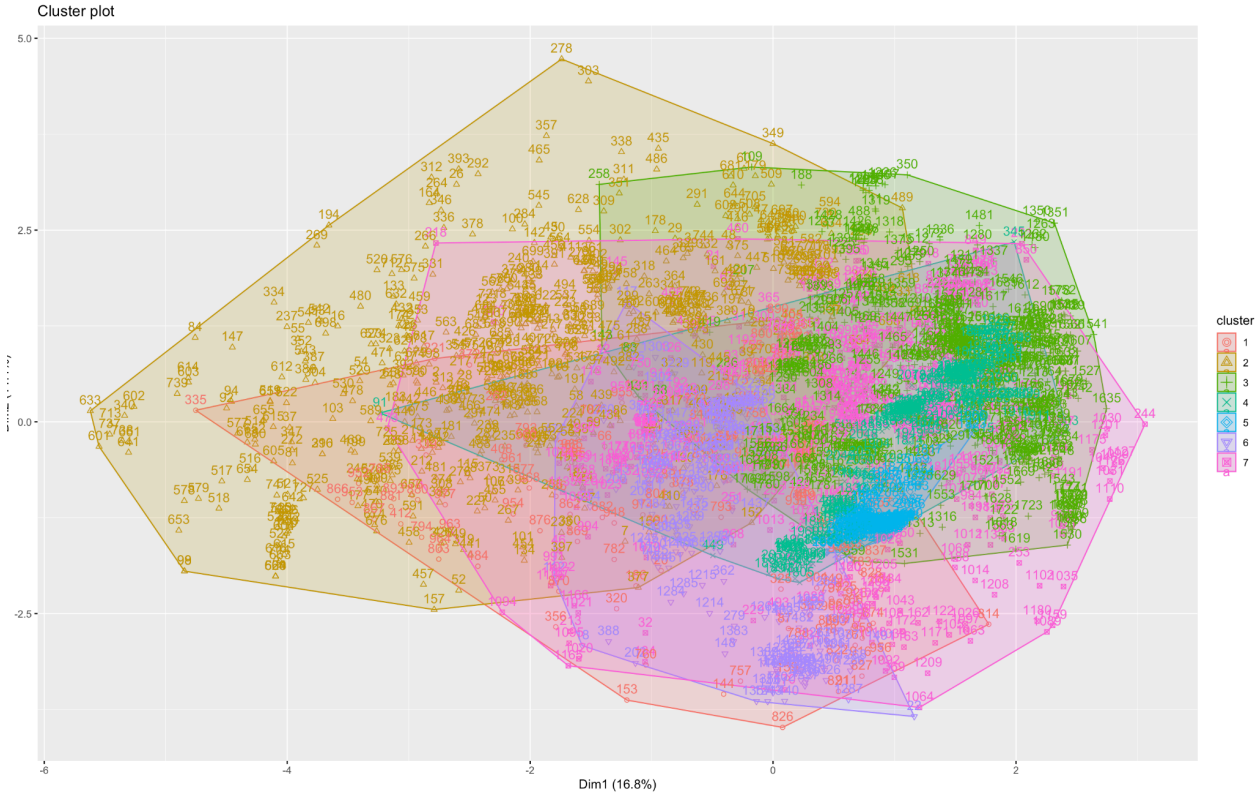
Cluster no 3



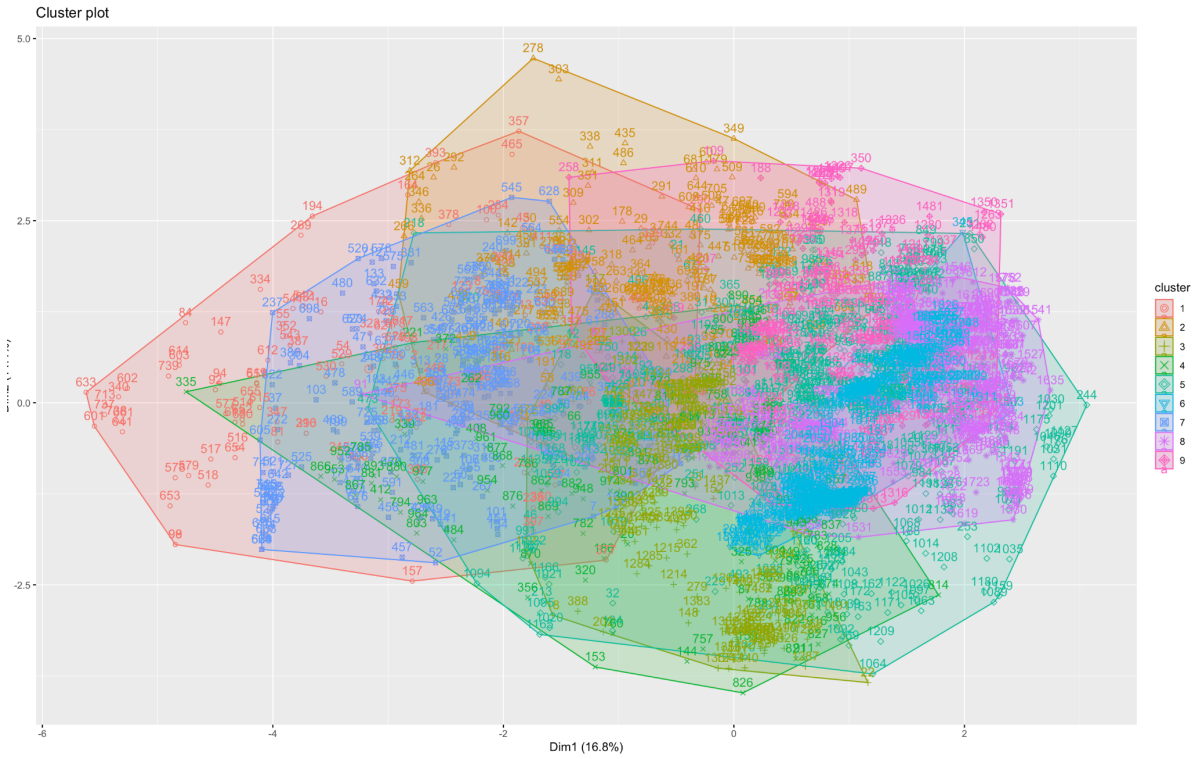
Cluster no 5



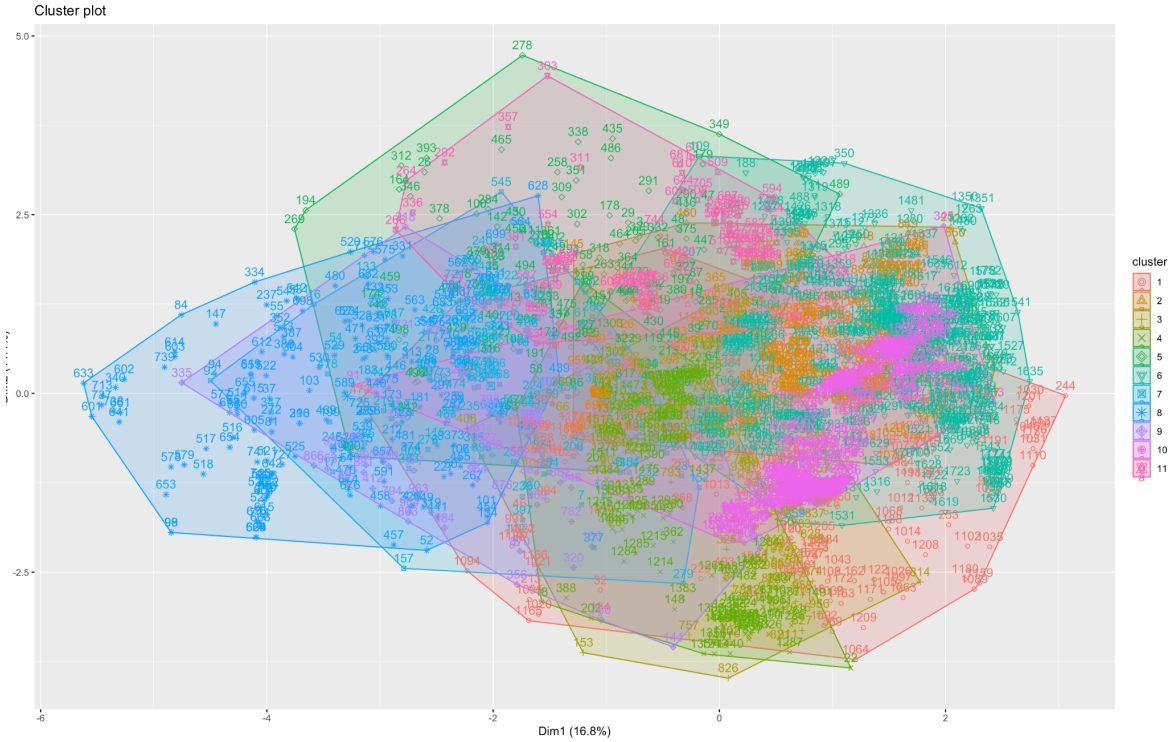
Cluster no 7



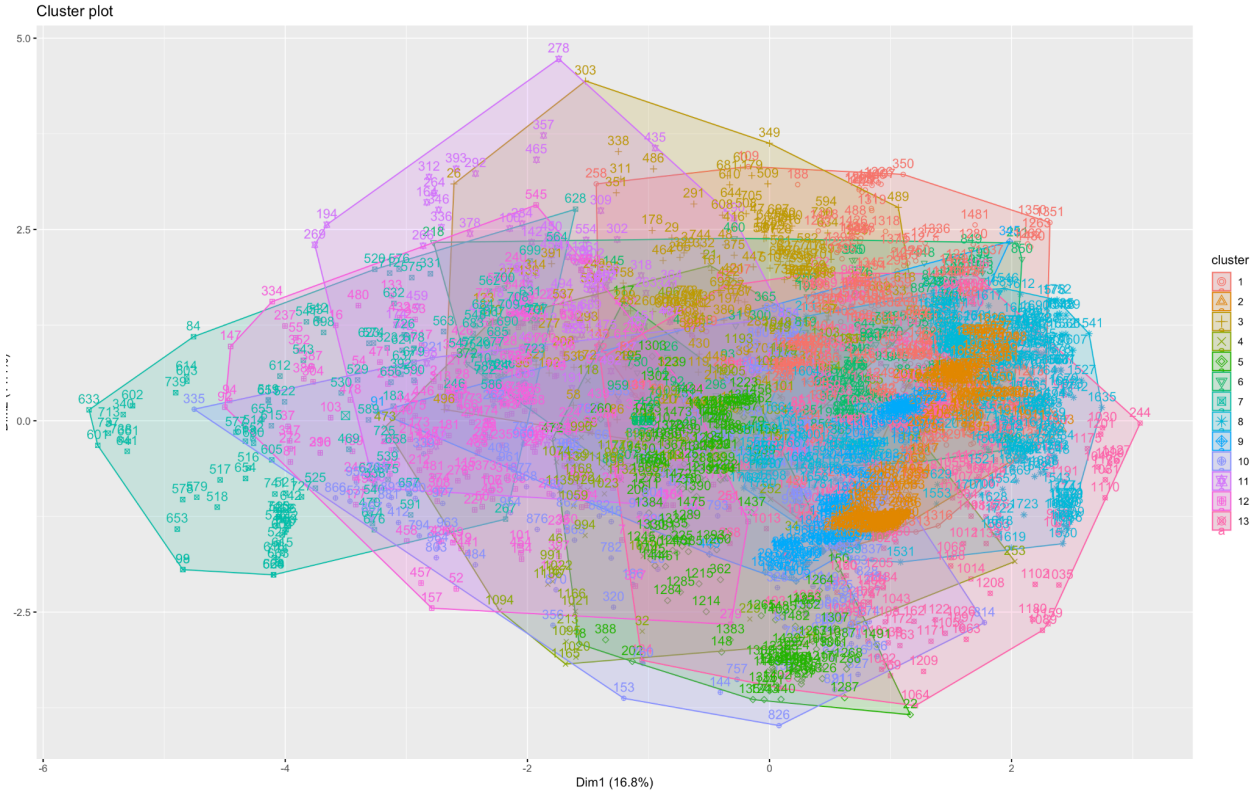
Cluster no 9



Cluster no 11



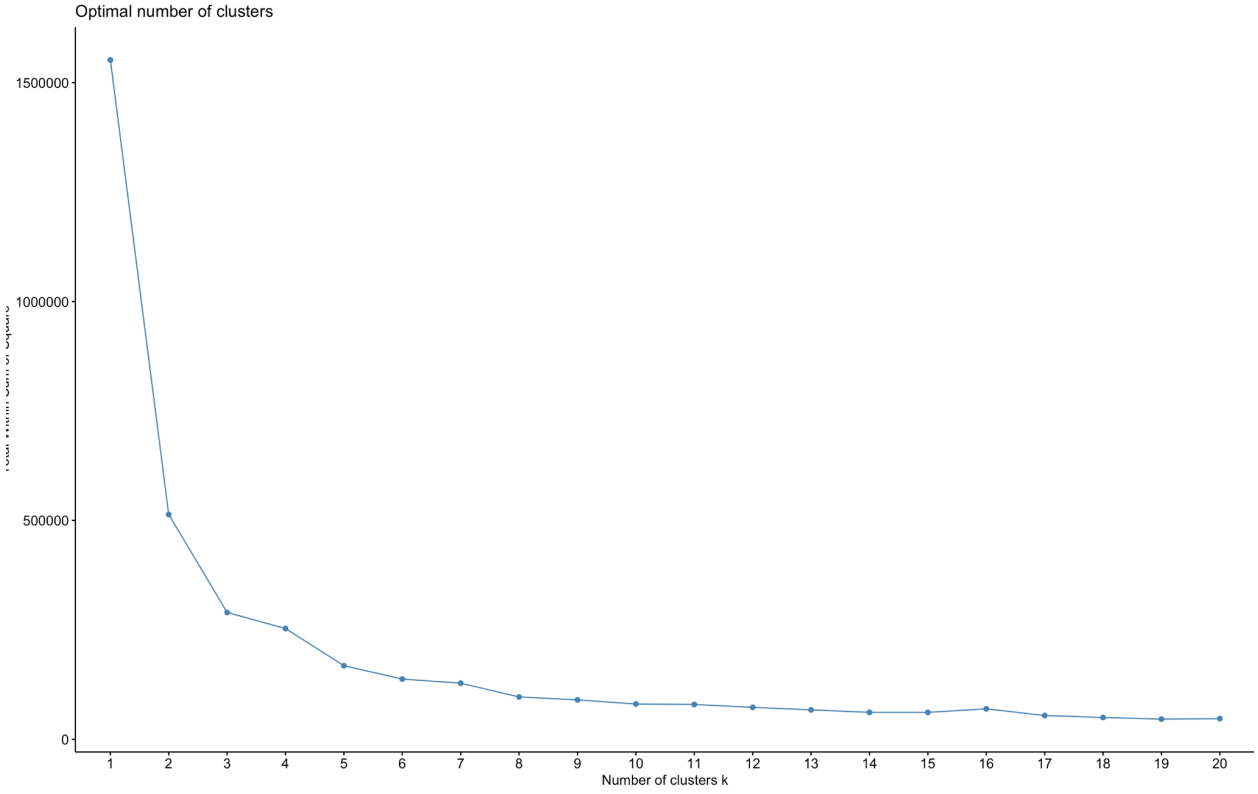
Cluster no 13



From these plots we can observe clear distinction in the data points as the number of clusters increase. However, there is also significant overlap which can be due to external factors that are complex and hard to capture with clustering methods. It can also be due to outliers in the data that could be distorting the clusters.

Optimal number of clusters:
In order to find an optimal or **good number of clusters**, we can use the **wssplot** method outlined in the rubric as well as **factoextra's fviz_nbclust()** method. The plots are shown below, **factoextra::fviz_nbclust()** and **wssplot()** respectively:

```
factoextra::fviz_nbclust(clusteringDataset,
                         FUNcluster = kmeans,
                         method = "wss",
                         k.max = 20,
                         verbose = TRUE)
```



From this we can see that the optimal number of clusters will be between 5-8 clusters.

Task 4

Prediction using the training and test data set

Linear Modeling: To proceed with linear modeling we use the glm() function from the stats package. We also applied the anova() method for analysis of variance on the model result.
glm(): A Generalized Linear Model (GLM) is a statistical method that extends the concept of linear regression for various types of dependent variables.

anova(): Analysis of Variance (ANOVA) is a statistical technique used to compare the means of two or more groups

```
# GLM for 50-50% Split of the training and testing data
clusteringDataset.train.glm.50_50 =  glm(NObeyesdad ~ .,
                                         data = clusteringDataset.tain.50_50,
                                         family = "gaussian")

clusteringDataset.test.pred.50_50 <- predict(clusteringDataset.train.glm.50_50,
                                             newdata = clusteringDataset.test.50_50,
                                             type = "response")
```

clusteringDataset.train.glm.50_50
The formula NObeyesdad ~ . specifies that the target variable is "NObeyesdad"
The family = "gaussian" argument specifies a Gaussian (normal) distribution

clusteringDataset.test.pred.50_50
This line uses the fitted GLM model (clusteringDataset.train.glm.50_50) to make predictions on the unseen testing data.

```
# Adjusting predictions and generating true labels for 50-50 split
clusteringDataset.test.pred.class.50_50 <- ifelse(clusteringDataset.test.pred.50_50 > 0.5, 1, 0)
clusteringDataset.labels.50_50 <- clusteringDataset.test.50_50$NObeyesdad
```

This line converts the continuous GLM predictions into class labels (0 or 1).

It uses an ifelse statement to assign 1 if the predicted probability is greater than 0.5) and 0 otherwise.

```
> # Accuracy Calculation for 50-50 split
> clusteringDataset.accuracy50_50 <- mean(clusteringDataset.labels.50_50 == clusteringDataset.test.pred.class.50_5
0)
> cat("Accuracy for GLM predictions for a 50-50 split in the dataset is =", clusteringDataset.accuracy50_50, "\n")
Accuracy for GLM predictions for a 50-50 split in the dataset is = 0.1222749
>
```

This calculates the accuracy of the GLM model on the testing set.

It compares the predicted class labels (clusteringDataset.test.pred.class.50_50) with the actual labels (clusteringDataset.labels.50_50) from the testing data and calculates the average proportion of correct predictions (accuracy).

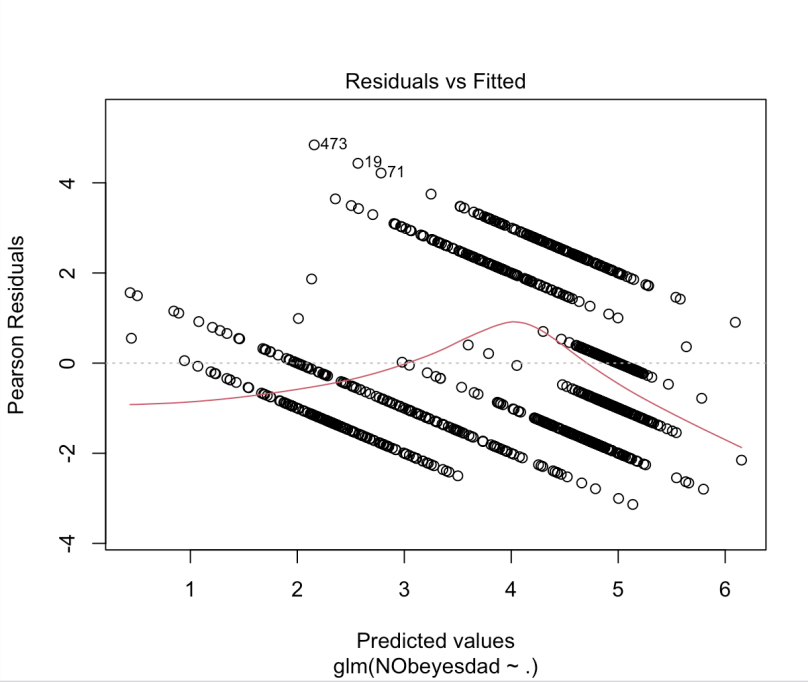
We can observe the difference between the null deviance and the residual deviance. We want this difference to be as high as possible

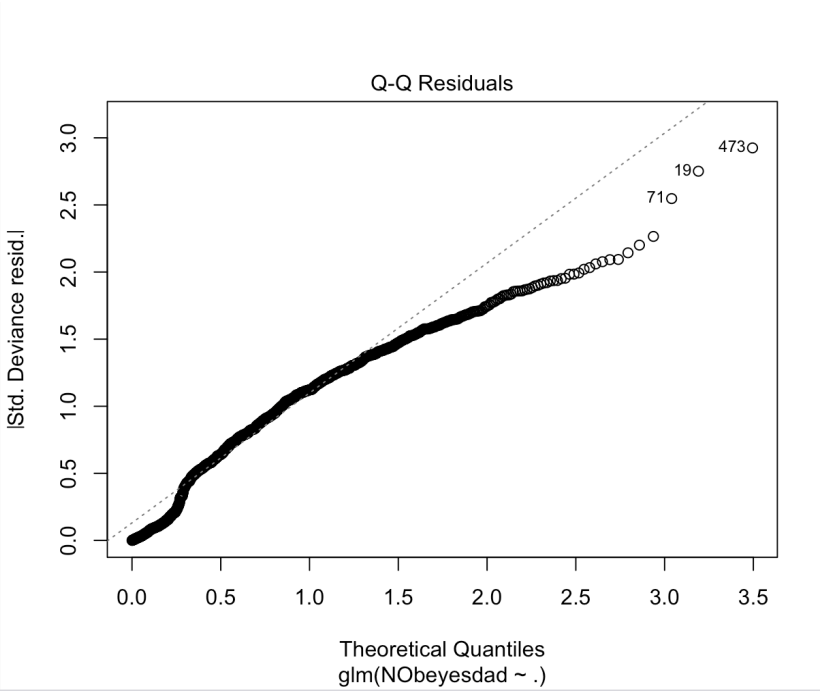
```
> # Anova Test for GLM Model for 50-50 split of training and testing data set
> clusteringDataset.train.glm.anova.50_50 <- anova(clusteringDataset.train.glm.50_50,
+                                                  test = "Chisq")
> clusteringDataset.train.glm.anova.50_50
Analysis of Deviance Table

Model: gaussian, link: identity

Response: NObeyesdad

Terms added sequentially (first to last)


                               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                                            1055     4146.6
Gender                          1     0.91      1054     4145.7  0.570195
Age                             1   246.57      1053     3899.1 < 2.2e-16 ***
Height                          1    20.66      1052     3878.5  0.006688 **
Weight                          1   610.17      1051     3268.3 < 2.2e-16 ***
family_history_with_overweight  1    73.48      1050     3194.8 3.142e-07 ***
FAVC                            1    18.76      1049     3176.0  0.009744 **
FCVC                            1    22.14      1048     3153.9  0.004994 **
NCP                             1    52.97      1047     3100.9 1.408e-05 ***
CAEC                            1   116.00      1046     2984.9 1.305e-10 ***
SMOKE                           1     0.21      1045     2984.7  0.785173
CH2O                            1     8.98      1044     2975.7  0.073706 .
SCC                             1    12.91      1043     2962.8  0.032009 *
FAF                             1    17.24      1042     2945.6  0.013220 *
TUE                             1     0.57      1041     2945.0  0.653724
CALC                            1     0.10      1040     2944.9  0.851904
MTRANS                          1    26.74      1039     2918.2  0.002032 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

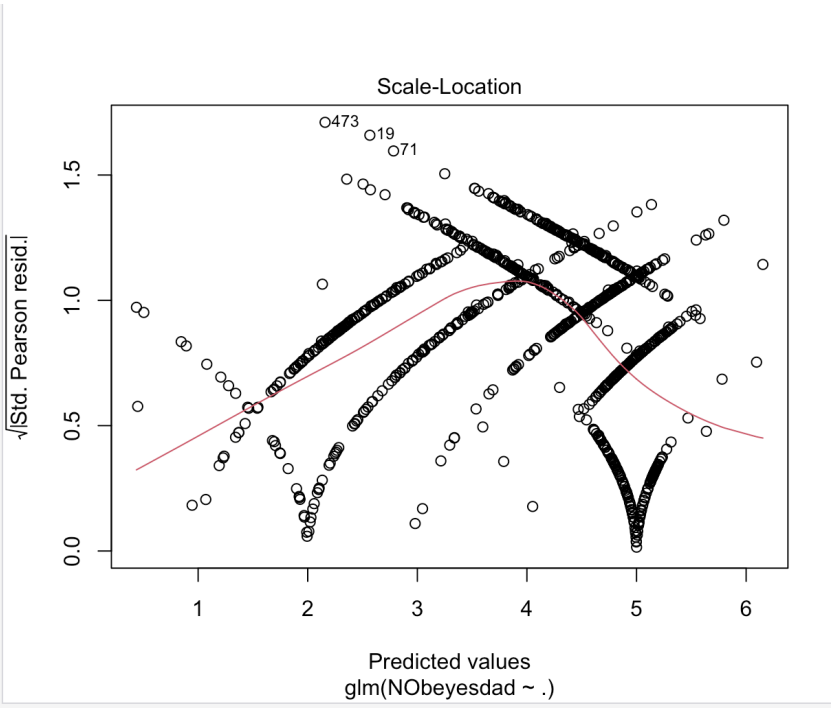
Now, to plot the **residual and fitted values**, **Q-Q plot**, **Scale Location Plot** (**Spread-Location Plot**), and the **Residuals vs Leverage Plot** for the GLM 50-50% split

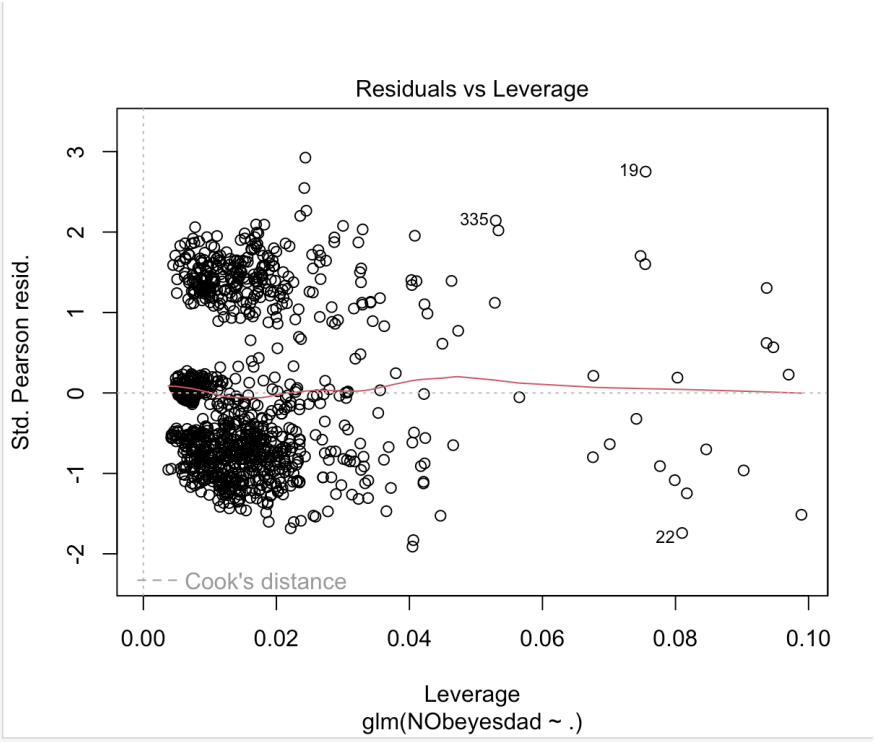
plot(clusteringDataset.train.glm.50_50)

```
>
> plot(clusteringDataset.train.glm.50_50)
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
Hit <Return> to see next plot:
>
```






Scale-Location
473
19
71
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
1
2
3
4
5
6
Predicted values
glm(NObeyesdad ~ .)


Residuals vs Leverage
19
335
22
Cook's distance
Std. Pearson resid.
-2
-1
0
1
2
3
0.00
0.02
0.04
0.06
0.08
0.10
Leverage
glm(NObeyesdad ~ .)

We can understand several things from these plots.
Residual vs fitted plot: From this plot, we can clearly see a few outliers dragging the linear model down. There might be some other dependencies as well.

From the **Q-Q plot**, it is evident that the data is normally distributed until the higher end where certain data points start dragging the curve upward.

From the **Spread Location Plot**, we can see that our line is almost horizontal and has relatively equally spread-out points.

From the **Residuals vs Leverage Plot**, which helps us find influential cases/data points. Here, we can notice that no points are outside Cook's distance and thus not influential in affecting the regression. We can still spot outliers though.

Predictions:

```
> clusteringDataset.test.pred.50_50 <- predict(clusteringDataset.train.glm.50_50,
+                                              newdata = clusteringDataset.test.50_50,
+                                              type = "response")
> clusteringDataset.test.pred.50_50
        3         9        12        14        15        18        20        23        25        26
4.3309270 3.5812328 3.4323403 4.1304503 3.6718827 4.1511727 5.1195262 3.0012451 3.7321194 2.2254116
       28        30        34        38        41        42        43        44        45        49
0.9826665 3.2629770 5.5798303 2.6136337 3.9930171 2.6592992 4.0429213 3.6979005 3.3102220 4.0163022
       51        53        55        57        60        61        63        65        66        67
3.8100489 2.9044327 1.4799147 3.3688560 3.0813922 3.3366387 3.1986561 2.3934923 3.2786639 3.1159439
       68        69        72        73        75        76        77        80        83        85
2.7517635 5.2758748 3.7902794 5.1702751 3.6921843 4.4038546 3.5746528 4.4586354 2.9710208 3.4568546
       90        92        95        97        99       100       103       106       108       111
3.4162449 3.3010725 2.5784144 3.9166257 2.4231975 3.4078736 1.7634744 3.4903419 2.0346885 3.7858713
      112       113       114       116       118       120       124       125       127       128
4.3001129 2.1747553 4.5035747 2.1469070 3.3803584 2.5269337 4.2938065 3.4544177 3.0508213 3.9325360
      130       133       135       138       139       141       148       149       150       151
3.6199690 2.9223503 4.0800925 6.0865563 3.3270590 2.4508295 4.7157890 1.2796727 4.4938678 4.4019507
      153       154       155       156       159       160       162       165       166       167
3.3297000 4.9140608 4.5768922 2.8965873 4.9589122 4.6741368 4.8537894 3.5165594 5.6546684 2.6255576
      169       172       173       175       176       179       180       185       186       189
4.5244319 2.5419960 3.6420406 3.3844725 4.4964282 2.6397940 3.3844725 3.3844725 3.4756487 4.3809910
      191       198       202       204       205       209       212       217       219       220
2.9293576 5.8054248 3.3308235 4.8996390 3.1348028 3.7355782 3.0885471 2.9655146 4.3677162 4.0899946
      222       231       232       233       235       236       237       238       239       242
```

```
> summary(clusteringDataset.test.pred.50_50)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.9556  3.3845  4.3751  4.0499  4.8594  6.0866
>
```

Confidence Interval:

```
> confint(clusteringDataset.train.glm.50_50)
Waiting for profiling to be done...
                                       2.5 %      97.5 %
(Intercept)                         0.73364265  2.56469061
Gender                             -0.25097806  0.31104990
Age                                 1.57414878  3.55978228
Height                             -1.63270028  0.22582183
Weight                              2.22080051  3.78657697
family_history_with_overweight      0.38505505  1.02712496
FAVC                               -0.80410857 -0.13550142
FCVC                               -0.88547143 -0.05805465
NCP                                -1.10645019 -0.29386232
CAEC                                1.09998882  1.98716197
SMOKE                              -0.88354968  0.74394366
CH2O                               -0.02196128  0.66587964
SCC                                 0.12781883  1.17267301
FAF                                -0.89724383 -0.10015880
TUE                                -0.47924740  0.23382973
CALC                               -0.72208014  0.46967469
MTRANS                              0.24359488  1.09191922
>
```

We notice the variability in the Intercept here, which is on the higher end. Other factors affecting the confidence interval include the confidence level and the sample size.

To proceed for comparing actual vs predictions, we start by obtaining the K-means data for the predicted values. Post this, we compare the results from the two clusterings using the CrossTable method as shown in the rubric.

```
# K-Means Clustering for the Test Set (50-50% Split)
kMeans.clusters.50_50 <- 5
predicted.clusters.50_50 <- kmeans(clusteringDataset.test.50_50[, -ncol(clusteringDataset.test.50_50)],
                                   centers = kMeans.clusters.50_50)$cluster

predicted.clusters.50_50
```

```
> # K-Means Clustering for the Test Set (50-50% Split)
> kMeans.clusters.50_50 <- 5
> predicted.clusters.50_50 <- kmeans(clusteringDataset.test.50_50[, -ncol(clusteringDataset.test.50_50)],
+                                    centers = kMeans.clusters.50_50)$cluster
> predicted.clusters.50_50
```

3	9	12	14	15	18	20	23	25	26	28	30	34	38	41	42	43	44	45	49	51
3	3	1	2	3	2	5	4	3	1	2	2	2	2	4	2	4	3	2	5	4
53	55	57	60	61	63	65	66	67	68	69	72	73	75	76	77	80	83	85	90	92
2	5	3	3	4	2	1	1	1	1	3	5	5	1	5	5	3	1	3	4	5
95	97	99	100	103	106	108	111	112	113	114	116	118	120	124	125	127	128	130	133	135
2	5	5	2	2	2	1	3	4	1	5	1	5	5	3	2	2	2	2	1	3
138	139	141	148	149	150	151	153	154	155	156	159	160	162	165	166	167	169	172	173	175
3	5	3	5	2	4	3	1	3	3	1	5	3	2	3	3	5	4	3	5	2
176	179	180	185	186	189	191	198	202	204	205	209	212	217	219	220	222	231	232	233	235
4	1	2	2	5	3	2	3	1	3	3	4	2	5	5	4	3	3	4	5	4
236	237	238	239	242	244	247	252	256	258	261	265	268	269	270	271	272	276	278	279	281
5	2	4	2	2	3	2	3	3	2	2	3	2	2	3	2	2	1	2	4	2
285	286	289	290	292	295	297	298	300	301	302	305	306	307	308	309	311	313	314	316	319
3	1	2	2	2	3	2	2	2	2	2	3	3	1	1	2	1	2	1	2	2
320	321	323	324	327	330	332	334	337	338	341	342	343	344	347	348	349	350	351	353	354
2	1	5	3	3	4	3	5	2	1	2	2	3	2	5	3	3	1	3	2	4
355	357	363	364	367	368	369	370	371	372	373	374	375	376	377	378	379	385	391	392	395
2	2	5	2	3	2	4	2	2	1	2	2	3	3	5	2	2	4	1	5	4
397	400	401	403	405	409	411	414	415	416	420	421	423	425	426	427	428	429	431	433	435

The cross tables for the three splits are shown below:

```
> # Cross-Tabulation to Compare Clusters and True Labels for 50-50 split
> clusteringDataset.test.crossTable.k5.50_50 <- CrossTable(x = as.factor(clusteringDataset.labels.50_50),
+                                                          y = as.factor(predicted.clusters.50_50),
+                                                          prop.chisq = FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  1055


                                    | as.factor(predicted.clusters.50_50)
as.factor(clusteringDataset.labels.50_50) |         1 |         2 |         3 |         4 |         5 | Row Total |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  1 |        23 |        47 |        35 |         0 |        24 |       129 |
                                    |     0.178 |     0.364 |     0.271 |     0.000 |     0.186 |     0.122 |
                                    |     0.418 |     0.278 |     0.080 |     0.000 |     0.338 |           |
                                    |     0.022 |     0.045 |     0.033 |     0.000 |     0.023 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  2 |        18 |        65 |        25 |        15 |        17 |       140 |
                                    |     0.129 |     0.464 |     0.179 |     0.107 |     0.121 |     0.133 |
                                    |     0.327 |     0.385 |     0.057 |     0.047 |     0.239 |           |
                                    |     0.017 |     0.062 |     0.024 |     0.014 |     0.016 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  3 |         6 |         5 |        98 |        76 |         1 |       186 |
                                    |     0.032 |     0.027 |     0.527 |     0.409 |     0.005 |     0.176 |
                                    |     0.109 |     0.030 |     0.223 |     0.237 |     0.014 |           |
                                    |     0.006 |     0.005 |     0.093 |     0.072 |     0.001 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  4 |         0 |         0 |       159 |         0 |         0 |       159 |
                                    |     0.000 |     0.000 |     1.000 |     0.000 |     0.000 |     0.151 |
                                    |     0.000 |     0.000 |     0.362 |     0.000 |     0.000 |           |
                                    |     0.000 |     0.000 |     0.151 |     0.000 |     0.000 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  5 |         0 |         0 |         0 |       154 |         0 |       154 |
                                    |     0.000 |     0.000 |     0.000 |     1.000 |     0.000 |     0.146 |
                                    |     0.000 |     0.000 |     0.000 |     0.480 |     0.000 |           |
                                    |     0.000 |     0.000 |     0.000 |     0.146 |     0.000 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  6 |         4 |        43 |        43 |        54 |         6 |       150 |
                                    |     0.027 |     0.287 |     0.287 |     0.360 |     0.040 |     0.142 |
                                    |     0.073 |     0.254 |     0.098 |     0.168 |     0.085 |           |
                                    |     0.004 |     0.041 |     0.041 |     0.051 |     0.006 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  7 |         4 |         9 |        79 |        22 |        23 |       137 |
                                    |     0.029 |     0.066 |     0.577 |     0.161 |     0.168 |     0.130 |
                                    |     0.073 |     0.053 |     0.180 |     0.069 |     0.324 |           |
                                    |     0.004 |     0.009 |     0.075 |     0.021 |     0.022 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                       Column Total |        55 |       169 |       439 |       321 |        71 |      1055 |
                                    |     0.052 |     0.160 |     0.416 |     0.304 |     0.067 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|


>
```

Similarly we will run the 60-40 split and 70-30 split
60-40 Split

```
# GLM for 60-40% Split of the training and testing data
clusteringDataset.train.glm.60_40 =  glm(NObeyesdad ~ .,
                                         data = clusteringDataset.tain.60_40,
                                         family = "gaussian")

clusteringDataset.test.pred.60_40 <- predict(clusteringDataset.train.glm.60_40,
                                             newdata = clusteringDataset.test.60_40,
                                             type = "response")
```

```
# Adjusting predictions and generating true labels for 60_40 split
clusteringDataset.test.pred.class.60_40 <- ifelse(clusteringDataset.test.pred.60_40 > 0.5, 1, 0)
clusteringDataset.labels.60_40 <- clusteringDataset.test.60_40$NObeyesdad
```

Accuracy

This calculates the accuracy of the GLM model on the testing set.

```
> # Accuracy Calculation for 60_40 split
> clusteringDataset.accuracy.60_40 <- mean(clusteringDataset.labels.60_40 == clusteringDataset.test.pred.class.60_40)
> cat("Accuracy for GLM predictions for a 60_40 split in the dataset is =", clusteringDataset.accuracy.60_40, "\n")
Accuracy for GLM predictions for a 60_40 split in the dataset is = 0.1244076
>
```

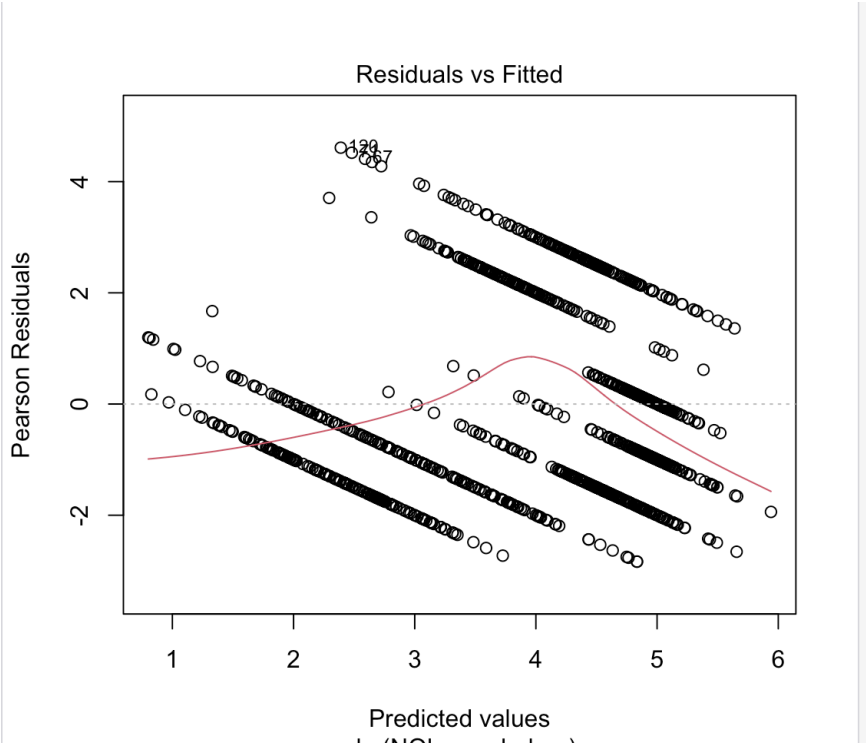
Anova test

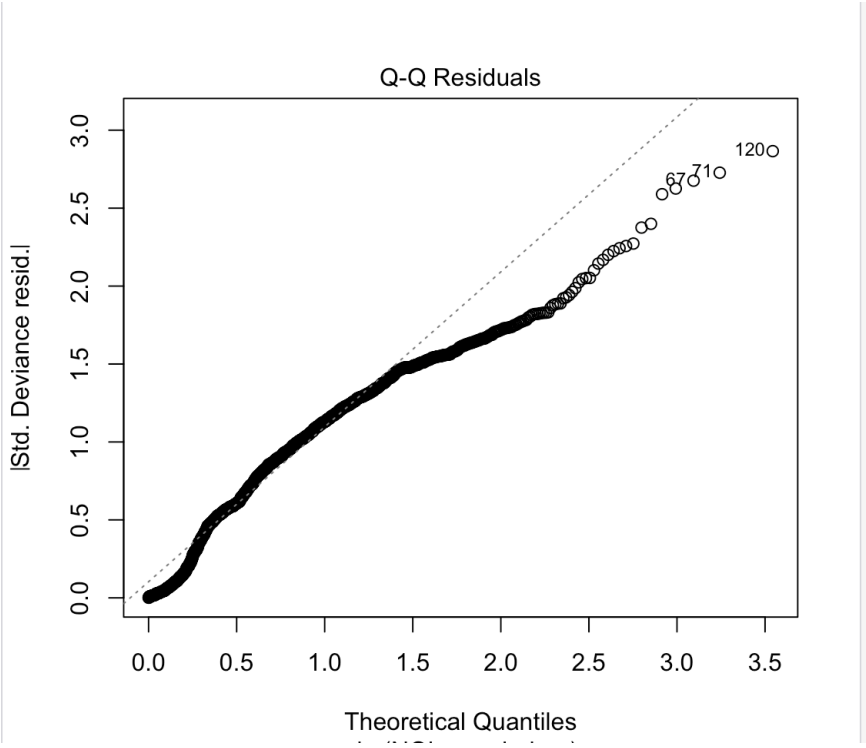
```
271  # Anova Test for GLM Model for 60_40 split of training and testing data set
272  clusteringDataset.train.glm.anova.60_40 <- anova(clusteringDataset.train.glm.60_40,
273                                                   test = "Chisq")
274  clusteringDataset.train.glm.anova.60_40
275
```

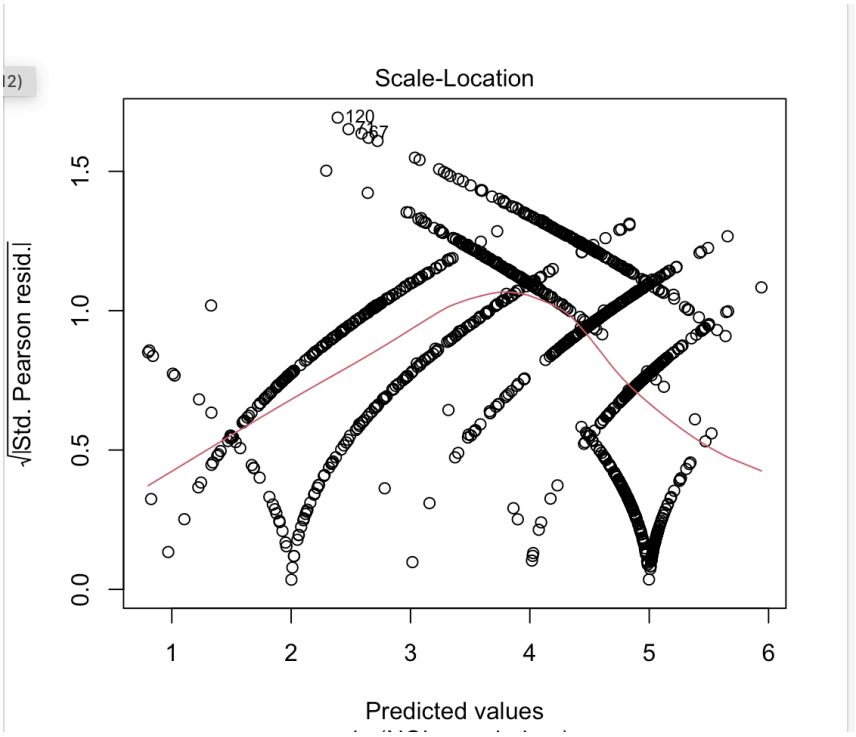
```
> # Anova Test for GLM Model for 60_40 split of training and testing data set
> clusteringDataset.train.glm.anova.60_40 <- anova(clusteringDataset.train.glm.60_40,
+                                                  test = "Chisq")
> clusteringDataset.train.glm.anova.60_40
Analysis of Deviance Table

Model: gaussian, link: identity

Response: NObeyesdad

Terms added sequentially (first to last)


                              Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                                         1266     4887.0
Gender                         1     6.26      1265     4880.7 0.1352502
Age                            1   277.16      1264     4603.6 < 2.2e-16 ***
Height                         1    11.00      1263     4592.6 0.0476765 *
Weight                         1   727.79      1262     3864.8 < 2.2e-16 ***
family_history_with_overweight 1    51.92      1261     3812.9 1.697e-05 ***
FAVC                           1    41.57      1260     3771.3 0.0001186 ***
FCVC                           1    18.97      1259     3752.3 0.0093213 **
NCP                            1    38.13      1258     3714.2 0.0002275 ***
CAEC                           1   163.54      1257     3550.7 2.267e-14 ***
SMOKE                          1    17.84      1256     3532.8 0.0116763 *
CH2O                           1     8.88      1255     3523.9 0.0752448 .
SCC                            1     4.72      1254     3519.2 0.1944152
FAF                            1     4.06      1253     3515.1 0.2291312
TUE                            1     0.30      1252     3514.8 0.7443529
CALC                           1     3.78      1251     3511.1 0.2456367
MTRANS                         1     3.69      1250     3507.4 0.2516248
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

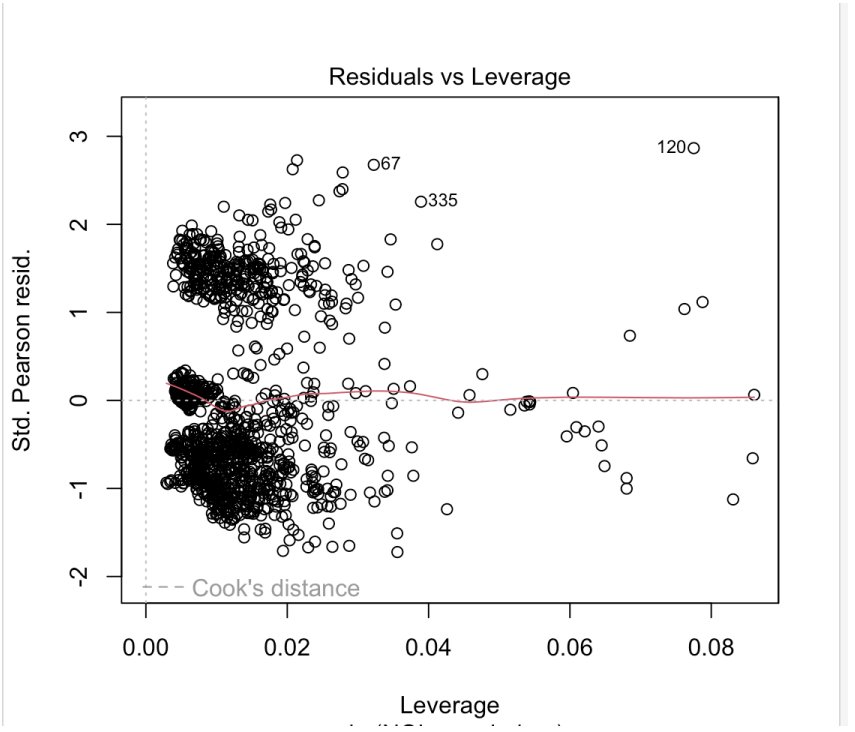
Now, to plot the **residual and fitted values**, **Q-Q plot**, **Scale Location Plot** (**Spread-Location Plot**), and the **Residuals vs Leverage Plot** for the GLM 60-40% split

plot(clusteringDataset.train.glm.60_40)






Scale-Location
120
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
1
2
3
4
5
6
Predicted values


Residuals vs Leverage
120
67
335
Std. Pearson resid.
-2
-1
0
1
2
3
Cook's distance
0.00
0.02
0.04
0.06
0.08
Leverage

Predictions:

```
> clusteringDataset.test.pred.60_40 <- predict(clusteringDataset.train.glm.60_40,
+                                              newdata = clusteringDataset.test.60_40,
+                                              type = "response")
> clusteringDataset.test.pred.60_40
        3         5         8        11        12        14        19        24        26        28        30
4.3014988 4.8513729 3.7351554 3.3139076 3.0405275 4.2229338 1.5782891 4.9109692 1.8970621 1.5261602 3.1905840
       31        32        33        39        40        42        43        45        46        49        53
2.7807379 5.4447225 4.1657239 3.9617516 3.9693676 2.7317705 3.6052692 3.5219092 4.5674058 4.0805955 2.9146871
       59        61        63        64        66        70        73        76        78        80        83
3.6464049 3.5440982 3.1583972 3.3972740 2.9376282 3.8292385 4.8324140 4.1968060 4.1318812 4.6468973 2.7625801
       84        86        89        91        93        94        97        99       100       107       108
3.0926526 4.5054693 2.7107875 2.5615084 5.1089314 1.9733478 4.2521372 2.7588211 3.7592492 3.6283027 2.0058611
      109       112       116       117       119       121       126       128       129       130       132
5.0870325 3.9336424 1.6575510 3.3346213 3.6946906 1.9422970 2.0673675 3.6039352 2.3686669 3.4885593 3.8331170
      133       135       137       138       141       144       145       147       148       149       150
1.3880784 4.4368925 3.7813781 5.3534239 2.1226062 3.8380412 5.0208750 2.3968572 5.1153049 1.5567690 4.7144987
      151       153       155       156       157       158       161       162       175       179       183
4.3098543 2.5603025 4.4127536 2.9580212 3.3089375 2.5283336 2.8621015 5.0225517 3.5462105 1.3201137 1.4361933
      184       186       189       192       193       197       198       199       200       202       203
3.4886158 4.1576058 4.4508704 2.6620594 4.5440538 4.4584931 5.7819475 2.1849351 3.8296889 3.4680469 3.8309051
      207       209       210       219       220       221       223       227       228       231       233
3.1243352 3.5636956 3.5636956 4.6811719 4.0162487 3.5875362 1.5831413 2.4469213 2.9421188 3.4585199 4.8410624
      235       236       241       243       244       245       246       247       251       252       262
3.0426275 4.0710083 3.4178481 2.0250812 3.7320009 5.0289589 2.0854526 3.1559949 3.9340122 5.0798878 4.1145631
      265       266       271       273       278       279       280       281       282       286       287
2.1553564 1.5675664 4.1585844 2.7038048 0.2110374 3.8810724 2.9867060 0.6055587 2.3091363 2.7831693 2.9879409
      289       290       293       294       297       298       304       305       309       319       323
```

```
> summary(clusteringDataset.test.pred.60_40)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.211   3.413   4.323   4.017   4.868   6.299
>
```

Confidence Interval:

```
> confint(clusteringDataset.train.glm.60_40)
Waiting for profiling to be done...
                                      2.5 %      97.5 %
(Intercept)                        1.10033186  2.81526212
Gender                            -0.06635579  0.43629734
Age                                1.27354793  3.21575412
Height                            -2.35824295 -0.68575635
Weight                             2.66702019  4.09446468
family_history_with_overweight     0.24282058  0.82722351
FAVC                              -1.09511607 -0.47634935
FCVC                              -0.79116968 -0.03476906
NCP                               -0.98724038 -0.23897984
CAEC                               1.15442596  1.97608491
SMOKE                             -1.66660893 -0.20138606
CH2O                              -0.01437920  0.61641568
SCC                               -0.11152790  0.78385015
FAF                               -0.54183568  0.17969105
TUE                               -0.36075123  0.29810528
CALC                              -0.23401444  0.85551732
MTRANS                            -0.16323020  0.62325976
>
```

We notice the variability in the Intercept here, which is on the higher end. Other factors affecting the confidence interval include the confidence level and the sample size.

To proceed for comparing actual vs predictions, we start by obtaining the K-means data for the predicted values. Post this, we compare the results from the two clusterings using the CrossTable method as shown in the rubric.

```
MIRAN5                          -0.16525020  0.62525976
> # K-Means Clustering for the Test Set (60_40% Split)
> kMeans.clusters.60_40 <- 5
> predicted.clusters.60_40 <- kmeans(clusteringDataset.test.60_40[, -ncol(clusteringDataset.test.60_40)],
+                                    centers = kMeans.clusters.60_40)$cluster
> predicted.clusters.60_40
   3    5    8   11   12   14   19   24   26   28   30   31   32   33   39   40   42   43   45   46   49   53   59
   2    4    4    2    5    4    5    5    2    1    4    4    5    1    2    5    4    5    4    5    5    1    5
  61   63   64   66   70   73   76   78   80   83   84   86   89   91   93   94   97   99  100  107  108  109  112
   5    4    2    5    2    5    5    5    3    2    1    3    5    1    3    1    5    1    4    1    3    2    5
 116  117  119  121  126  128  129  130  132  133  135  137  138  141  144  145  147  148  149  150  151  153  155
   5    5    2    5    3    1    3    4    4    5    3    2    3    2    1    2    1    5    1    5    2    5    3
 156  157  158  161  162  175  179  183  184  186  189  192  193  197  198  199  200  202  203  207  209  210  219
   5    5    2    2    4    4    2    1    5    1    3    3    3    2    3    5    5    5    5    3    5    5    5
 220  221  223  227  228  231  233  235  236  241  243  244  245  246  247  251  252  262  265  266  271  273  278
   5    5    5    1    1    3    5    5    5    1    5    3    5    1    1    5    3    5    3    4    4    4    4
 279  280  281  282  286  287  289  290  293  294  297  298  304  305  309  319  323  324  325  326  331  336  337
   5    4    4    3    3    1    1    4    2    3    1    4    1    2    4    1    1    3    5    1    1    4    4
 338  339  342  343  345  347  350  353  359  361  364  365  366  372  376  378  381  387  389  397  402  404  407
   2    1    4    3    2    5    2    1    3    4    4    2    4    2    3    4    4    1    3    5    4    5    4
 408  409  413  415  418  423  425  426  428  429  430  435  437  438  440  441  446  449  450  454  455  456  457
   5    3    4    2    2    4    4    5    1    4    3    4    1    5    1    1    1    5    5    1    1    4    1
 458  463  467  470  475  476  479  484  492  493  501  503  504  507  510  512  513  514  515  517  519  520  523
   1    3    3    1    4    1    4    1    4    4    5    5    5    5    2    5    5    1    1    1    1    1    1
 526  529  532  535  538  545  550  552  554  555  556  557  559  562  563  565  566  571  572  580  581  583  585
   1    1    3    2    1    5    5    5    4    4    4    4    3    1    1    3    3    1    1    3    3    5    5
 588  589  590  592  594  597  598  602  610  612  613  615  616  618  619  621  625  627  628  633  636  637  642
   1    1    1    3    3    3    2    1    2    1    1    1    1    3    5    1    4    3    1    1    1    1    1
 645  648  652  653  654  657  660  661  665  674  677  680  681  682  684  685  686  691  693  694  699  705  708
   2    5    1    1    1    1    1    1    5    1    1    5    2    5    1    1    5    4    2    2    1    2    1
 709  715  716  717  718  720  721  722  723  725  729  731  733  736  745  746  750  753  756  757  759  761  763
   1    1    3    3    3    5    5    1    1    1    3    3    3    2    2    4    4    2    2    5    4    5    4
 769  770  771  772  773  774  775  776  777  778  779  781  785  787  788  791  794  795  798  800  802  803  804
   2    3    2    5    3    5    5    3    3    5    2    3    1    1    5    4    1    5    3    5    5    1    4
 805  810  815  817  819  829  830  831  839  842  851  852  855  856  859  862  863  864  869  872  874  875  876
   4    5    2    2    2    5    5    4    3    2    3    5    3    3    3    1    5    5    1    3    5    5    5
 877  879  881  883  889  891  892  893  896  898  901  904  906  907  909  910  912  913  914  917  920  921  922
   5    5    1    4    5    5    5    1    4    3    5    1    2    3    2    2    5    5    4    5    5    4    4
 924  925  928  932  934  937  938  943  944  945  948  950  956  957  959  969  972  976  980  982  984  991  992
   4    5    3    5    2    5    5    5    2    2    1    2    5    3    4    5    3    5    2    3    3    5    2
 996  998 1000 1001 1002 1003 1004 1006 1007 1009 1011 1012 1014 1016 1019 1020 1021 1023 1024 1026 1031 1037 1046
   2    5    2    5    3    3    5    5    2    2    5    3    4    2    2    5    5    5    2    5    3    3    3
1049 1050 1051 1052 1055 1057 1060 1061 1062 1063 1065 1066 1067 1069 1070 1074 1075 1078 1081 1085 1088 1090 1095
   2    2    3    2    3    2    5    5    3    5    3    2    3    2    5    5    2    5    5    2    5    3    5
1099 1100 1102 1103 1105 1106 1107 1110 1111 1112 1118 1119 1120 1121 1122 1125 1126 1130 1131 1132 1133 1135 1138
   3    5    3    2    5    2    3    3    2    2    2    5    5    2    5    2    2    2    5    5    3    5    5
```

The cross tables for the three splits are shown below:

```
> # Cross-Tabulation to Compare Clusters and True Labels for 60_40 split
> clusteringDataset.test.crossTable.k5.60_40 <- CrossTable(x = as.factor(clusteringDataset.labels.60_40),
+                                                          y = as.factor(predicted.clusters.60_40),
+                                                          prop.chisq = FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  844


                                    | as.factor(predicted.clusters.60_40)
as.factor(clusteringDataset.labels.60_40) |         1 |         2 |         3 |         4 |         5 | Row Total |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  1 |        51 |        12 |        17 |         9 |        17 |       106 |
                                    |     0.481 |     0.113 |     0.160 |     0.085 |     0.160 |     0.126 |
                                    |     0.531 |     0.076 |     0.084 |     0.141 |     0.052 |           |
                                    |     0.060 |     0.014 |     0.020 |     0.011 |     0.020 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  2 |        29 |        14 |        10 |        28 |        28 |       109 |
                                    |     0.266 |     0.128 |     0.092 |     0.257 |     0.257 |     0.129 |
                                    |     0.302 |     0.089 |     0.049 |     0.438 |     0.086 |           |
                                    |     0.034 |     0.017 |     0.012 |     0.033 |     0.033 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  3 |         0 |        41 |        37 |         2 |        58 |       138 |
                                    |     0.000 |     0.297 |     0.268 |     0.014 |     0.420 |     0.164 |
                                    |     0.000 |     0.261 |     0.182 |     0.031 |     0.179 |           |
                                    |     0.000 |     0.049 |     0.044 |     0.002 |     0.069 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  4 |         1 |        35 |        90 |         0 |         1 |       127 |
                                    |     0.008 |     0.276 |     0.709 |     0.000 |     0.008 |     0.150 |
                                    |     0.010 |     0.223 |     0.443 |     0.000 |     0.003 |           |
                                    |     0.001 |     0.041 |     0.107 |     0.000 |     0.001 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  5 |         0 |         1 |         0 |         0 |       129 |       130 |
                                    |     0.000 |     0.008 |     0.000 |     0.000 |     0.992 |     0.154 |
                                    |     0.000 |     0.006 |     0.000 |     0.000 |     0.398 |           |
                                    |     0.000 |     0.001 |     0.000 |     0.000 |     0.153 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  6 |        15 |        20 |        23 |        19 |        45 |       122 |
                                    |     0.123 |     0.164 |     0.189 |     0.156 |     0.369 |     0.145 |
                                    |     0.156 |     0.127 |     0.113 |     0.297 |     0.139 |           |
                                    |     0.018 |     0.024 |     0.027 |     0.023 |     0.053 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  7 |         0 |        34 |        26 |         6 |        46 |       112 |
                                    |     0.000 |     0.304 |     0.232 |     0.054 |     0.411 |     0.133 |
                                    |     0.000 |     0.217 |     0.128 |     0.094 |     0.142 |           |
                                    |     0.000 |     0.040 |     0.031 |     0.007 |     0.055 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                       Column Total |        96 |       157 |       203 |        64 |       324 |       844 |
                                    |     0.114 |     0.186 |     0.241 |     0.076 |     0.384 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
```

70-30 Split:

```
288
289 ##########################################################################################
290
291 # GLM for 70-30% Split of the training and testing data
292 clusteringDataset.train.glm.70_30 =  glm(NObeyesdad ~ .,
293                                         data = clusteringDataset.tain.70_30,
294                                         family = "gaussian")
295
296 clusteringDataset.test.pred.70_30 <- predict(clusteringDataset.train.glm.70_30,
297                                              newdata = clusteringDataset.test.70_30,
298                                              type = "response")
```

```
299
300 # Adjusting predictions and generating true labels for 70-30 split
301 clusteringDataset.test.pred.class.70_30 <- ifelse(clusteringDataset.test.pred.70_30 > 0.5, 1, 0)
302 clusteringDataset.labels.70_30 <- clusteringDataset.test.70_30$NObeyesdad
303
```

Accuracy

This calculates the accuracy of the GLM model on the testing set.

```
> # Accuracy Calculation for 70-30 split
> clusteringDataset.accuracy.70_30 <- mean(clusteringDataset.labels.70_30 == clusteringDataset.test.pred.class.70_30)
> cat("Accuracy for GLM predictions for a 70-30 split in the dataset is =", clusteringDataset.accuracy.70_30, "\n")
Accuracy for GLM predictions for a 70-30 split in the dataset is = 0.1169036
>
```

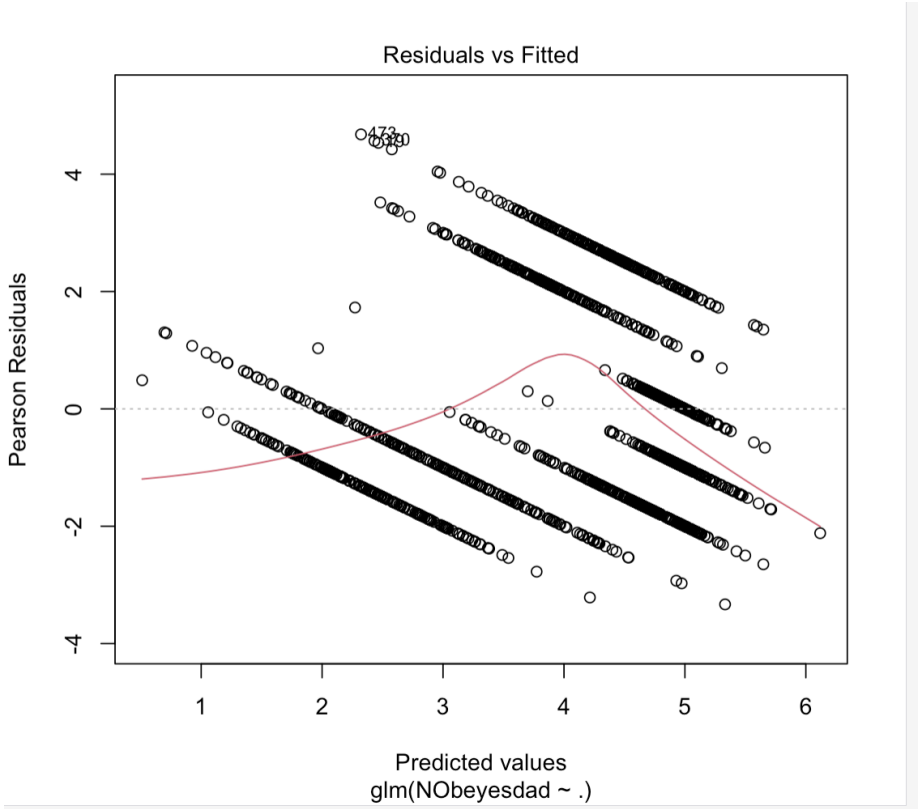
Anova test

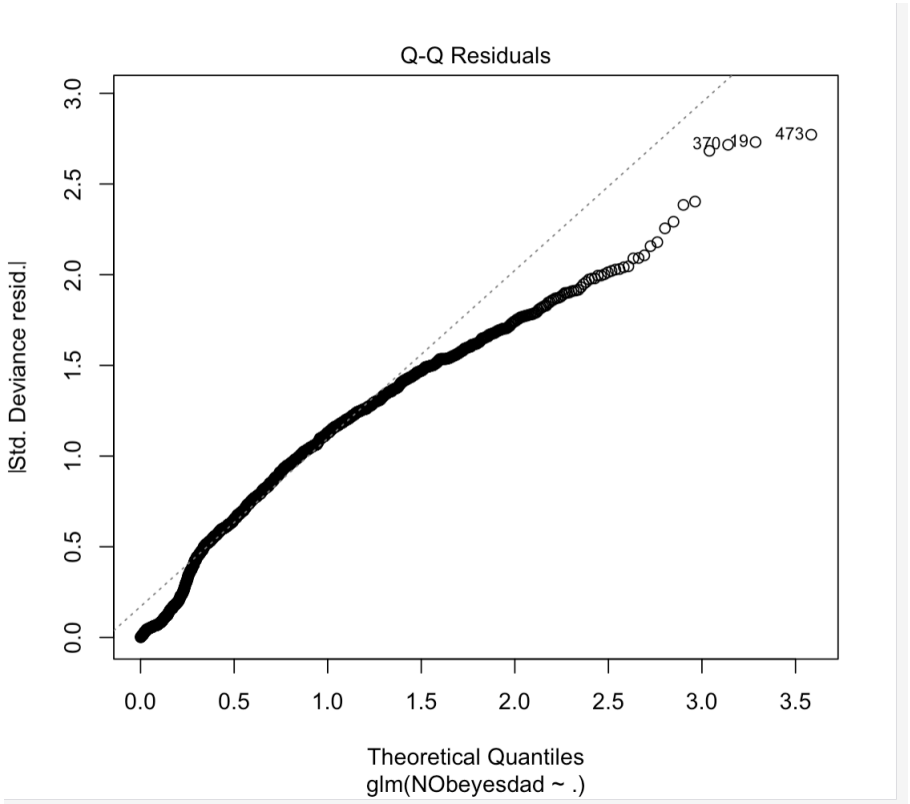
```
318 # Anova Test for GLM Model for 70-30 split of training and testing data set
319 clusteringDataset.train.glm.anova.70_30 <- anova(clusteringDataset.train.glm.70_30,
320                                                  test = "Chisq")
321
322 clusteringDataset.train.glm.anova.70_30
323
324
```

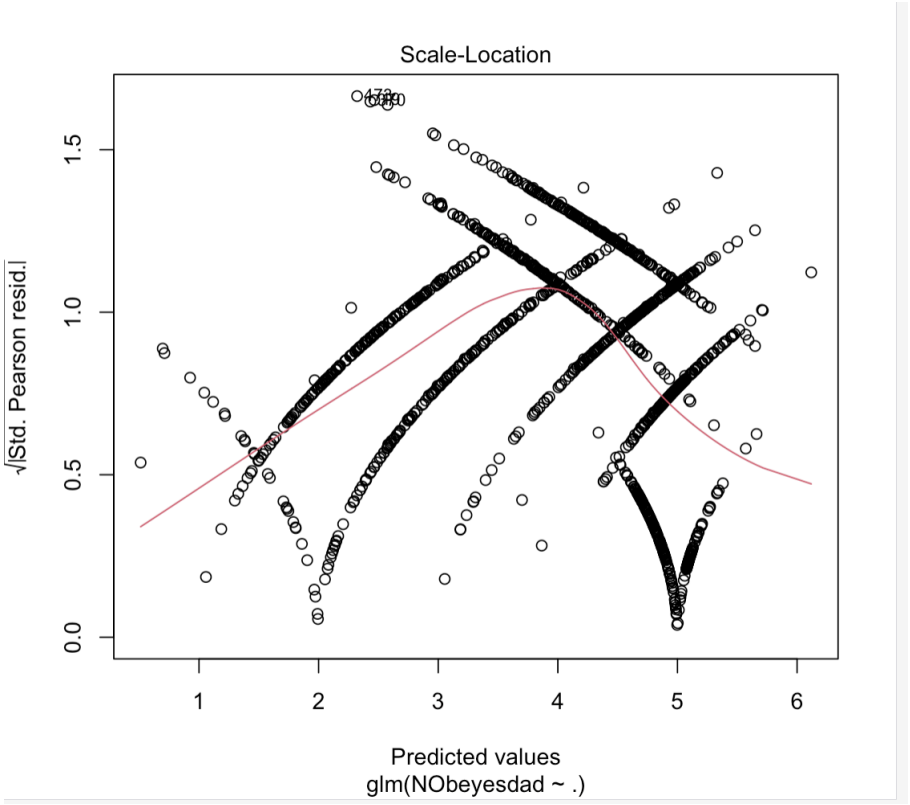
```
> # Anova Test for GLM Model for 70-30 split of training and testing data set
> clusteringDataset.train.glm.anova.70_30 <- anova(clusteringDataset.train.glm.70_30,
+                                                  test = "Chisq")
> clusteringDataset.train.glm.anova.70_30
Analysis of Deviance Table

Model: gaussian, link: identity

Response: NObeyesdad

Terms added sequentially (first to last)


                               Df Deviance Resid. Df Resid. Dev  Pr(>Chi)
NULL                                            1477     5831.9
Gender                          1     6.76      1476     5825.2  0.126604
Age                             1   365.80      1475     5459.4 < 2.2e-16 ***
Height                          1    24.85      1474     5434.5  0.003407 **
Weight                          1   791.00      1473     4643.5 < 2.2e-16 ***
family_history_with_overweight  1   125.47      1472     4518.0 4.696e-11 ***
FAVC                            1    12.32      1471     4505.7  0.039198 *
FCVC                            1    29.50      1470     4476.2  0.001418 **
NCP                             1    54.10      1469     4422.1 1.553e-05 ***
CAEC                            1   125.65      1468     4296.5 4.553e-11 ***
SMOKE                           1     4.33      1467     4292.1  0.221480
CH2O                            1     8.15      1466     4284.0  0.093506 .
SCC                             1    22.36      1465     4261.6  0.005471 **
FAF                             1    22.22      1464     4239.4  0.005619 **
TUE                             1     0.05      1463     4239.4  0.900595
CALC                            1     0.37      1462     4239.0  0.720446
MTRANS                          1     5.43      1461     4233.6  0.170923
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

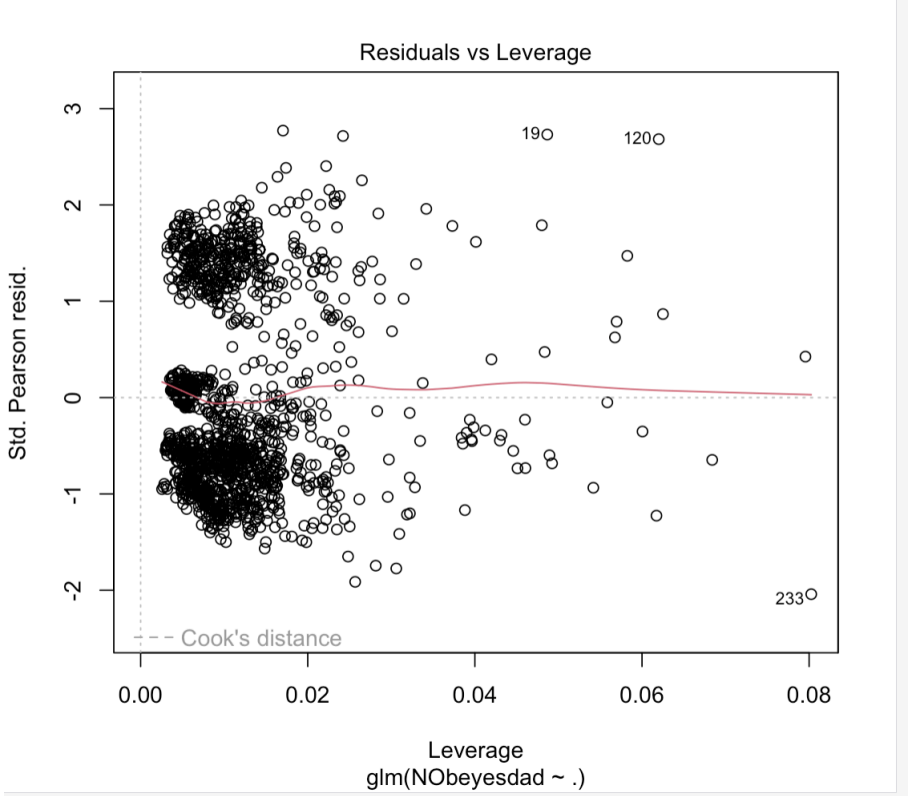
Now, to plot the **residual and fitted values**, **Q-Q plot**, **Scale Location Plot** (**Spread-Location Plot**), and the **Residuals vs Leverage Plot** for the GLM 60-40% split

plot(clusteringDataset.train.glm.70_30)


Residuals vs Fitted
Pearson Residuals
Predicted values
glm(NObeyesdad ~ .)


Q-Q Residuals
|Std. Deviance resid.|
Theoretical Quantiles
glm(NObeyesdad ~ .)


Scale-Location
√|Std. Pearson resid.|
0.0
0.5
1.0
1.5
1
2
3
4
5
6
Predicted values
glm(NObeyesdad ~ .)


Residuals vs Leverage
Std. Pearson resid.
-2
-1
0
1
2
3
19
120
233
Cook's distance
0.00
0.02
0.04
0.06
0.08
Leverage
glm(NObeyesdad ~ .)

Predictions:

```
Hit <Return> to see next plot:
> clusteringDataset.test.pred.70_30 <- predict(clusteringDataset.train.glm.70_30,
+                                              newdata = clusteringDataset.test.70_30,
+                                              type = "response")
> clusteringDataset.test.pred.70_30
       1        4       12       15       17       22       23       29       32       33       37       38       41       48
4.314253 3.662549 3.549730 3.574042 4.661718 5.324495 3.222862 3.883554 5.132083 3.772700 3.272152 2.564216 3.911461 3.131447
      58       62       66       73       77       78       89       90       97      101      102      109      117      118
4.266297 2.572871 3.363493 4.962879 3.479536 4.384003 3.346683 3.528979 3.801845 3.052759 3.503447 4.652992 3.252249 3.417744
     119      121      123      127      128      134      138      144      145      147      150      152      153      154
3.698299 2.855119 2.291835 3.038745 3.782059 3.387348 5.602561 3.801051 4.867601 2.115092 4.629284 3.500836 3.365621 5.032676
     155      156      162      163      164      166      168      170      172      177      178      179      180      182
4.524426 3.197631 4.912860 1.494179 2.687016 5.058334 4.586609 4.151594 2.757967 2.051347 3.169556 2.178090 3.405085 2.946374
     183      186      189      193      194      195      196      198      199      204      205      213      217      221
1.693522 3.630106 4.497642 4.366152 3.438315 3.053204 3.699535 5.870961 2.618961 5.107196 3.328349 5.115326 2.963834 4.124558
     228      229      232      237      243      244      246      250      259      267      275      276      283      285
2.798012 4.497189 3.795623 1.903106 2.535898 4.278193 1.974455 3.827971 3.652651 2.375641 3.312423 2.581884 2.357257 4.270810
     287      291      300      303      304      311      316      319      321      322      326      332      333      341
2.858466 2.326042 3.217339 1.039827 1.184571 1.882818 2.264632 2.930739 2.670873 3.338074 3.314855 3.038603 4.379037 2.682334
     347      354      361      368      374      384      385      395      401      405      412      413      414      415
2.111197 3.827475 1.802375 2.841823 3.560466 4.688843 4.430343 3.524944 4.367768 3.139857 4.074735 2.677273 5.051261 4.004547
     417      425      429      430      432      436      438      439      442      447      452      453      456      460
3.797304 4.252762 2.191968 3.353274 1.483263 3.940018 2.637062 2.238458 2.143901 2.488791 3.168279 2.492471 1.535291 3.027601
     461      462      466      471      478      482      486      489      494      497      501      506      509      511
2.357257 5.041592 2.329145 1.645522 2.767009 3.396519 2.306116 3.587072 1.509790 3.381280 5.118527 4.953242 2.911252 2.212168
     516      517      519      520      526      534      536      541      549      554      557      560      563      564
2.463126 2.546669 2.066126 2.615573 2.061401 2.857338 2.225700 1.564912 2.254254 1.934780 2.396270 2.863227 2.115461 1.469500
     565      567      569      582      587      588      590      591      597      598      602      605      606      607
2.533436 2.703898 1.965018 2.543342 2.749798 2.318510 2.495987 2.554920 2.555482 3.357073 2.333376 1.707376 2.085496 3.432351
     609      615      619      621      624      625      635      636      640      642      645      646      647      652
3.167117 1.994270 2.237658 2.700231 2.820692 2.927620 2.614181 2.532089 3.283429 2.069026 2.642172 1.956650 2.046607 2.294561
     656      657      662      665      671      683      686      694      695      701      705      713      716      717
2.961355 2.657293 2.040812 2.184481 2.860839 2.295125 2.626866 3.113887 2.440342 2.312339 2.973211 2.315702 2.488691 2.360211
     718      723      729      730      732      739      745      748      750      751      754      760      766      768
2.398243 2.486400 2.388206 2.499273 2.588942 2.602951 3.105907 3.316969 3.080931 3.873464 4.034132 3.954820 4.242549 3.338000
     769      770      773      783      790      793      796      807      810      812      815      818      821      823
3.589138 3.918105 3.764119 4.120742 4.014636 4.111859 2.983960 3.342789 3.453693 4.045006 3.975923 4.160153 4.454642 3.662335
     825      827      832      833      839      840      846      849      850      859      863      870      873      881
3.405085 4.568605 3.405085 3.405085 4.386823 3.624060 3.384861 3.454866 3.682522 4.208204 4.191662 3.550495 4.077310 3.490468
     889      896      898      899      902      912      913      914      915      921      922      928      930      931
```

```
> summary(clusteringDataset.test.pred.70_30)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.040   3.438   4.497   4.118   4.879   5.871
>
```

Confidence Interval:

```
> confint(clusteringDataset.train.glm.70_30)
Waiting for profiling to be done...
                                    2.5 %     97.5 %
(Intercept)                      1.21508063  2.7809452
Gender                          -0.12370989  0.3470521
Age                              1.38750675  3.2203427
Height                          -1.93176725 -0.3472901
Weight                           2.49295549  3.8213155
family_history_with_overweight   0.45476513  0.9880802
FAVC                            -0.72061090 -0.1483946
FCVC                            -0.82893410 -0.1071320
NCP                             -0.95380883 -0.2687778
CAEC                             0.92805321  1.6761797
SMOKE                           -1.09477688  0.1775436
CH2O                            -0.02032378  0.5810704
SCC                              0.22239956  1.1040190
FAF                             -0.80413892 -0.1266060
TUE                             -0.32764273  0.2794849
CALC                            -0.42391634  0.5626970
MTRANS                          -0.11112993  0.6263110
> 
```

We notice the variability in the Intercept here, which is on the higher end. Other factors affecting the confidence interval include the confidence level and the sample size.

To proceed for comparing actual vs predictions, we start by obtaining the K-means data for the predicted values. Post this, we compare the results from the two clusterings using the CrossTable method as shown in the rubric.

```
# K-Means Clustering for the Test Set (70-30% Split)
kMeans.clusters.70_30 <- 5
predicted.clusters.70_30 <- kmeans(clusteringDataset.test.70_30[, -ncol(clusteringDataset.test.70_30)],
                                   centers = kMeans.clusters.70_30)$cluster

predicted.clusters.70_30
```

```
> # K-Means Clustering for the Test Set (70-30% Split)
> kMeans.clusters.70_30 <- 5
> predicted.clusters.70_30 <- kmeans(clusteringDataset.test.70_30[, -ncol(clusteringDataset.test.70_30)],
+                                    centers = kMeans.clusters.70_30)$cluster
> predicted.clusters.70_30
   1    4   12   15   17   22   23   29   32   33   37   38   41   48   58   62   66   73   77   78   89   90   97  101  102
   4    1    5    3    3    5    5    2    4    4    4    4    5    2    3    1    5    4    4    5    5    5    4    4    3
 109  117  118  119  121  123  127  128  134  138  144  145  147  150  152  153  154  155  156  162  163  164  166  168  170
   2    5    4    2    5    3    1    4    4    3    4    2    4    5    5    5    2    3    5    1    5    1    3    2    5
 172  177  178  179  180  182  183  186  189  193  194  195  196  198  199  204  205  213  217  221  228  229  232  237  243
   3    5    2    2    1    5    4    4    2    2    1    5    2    3    5    3    2    4    4    4    4    4    5    4    5
 244  246  250  259  267  275  276  283  285  287  291  300  303  304  311  316  319  321  322  326  332  333  341  347  354
   3    4    3    1    4    4    5    5    3    4    2    1    2    4    2    1    4    2    2    4    2    3    1    4    5
 361  368  374  384  385  395  401  405  412  413  414  415  417  425  429  430  432  436  438  439  442  447  452  453  456
   1    4    4    3    5    5    3    1    4    1    1    2    1    1    1    3    4    5    5    4    5    3    3    1    1
 460  461  462  466  471  478  482  486  489  494  497  501  506  509  511  516  517  519  520  526  534  536  541  549  554
   2    5    2    5    4    5    3    2    2    1    2    5    5    2    5    4    4    4    4    4    2    3    4    4    1
 557  560  563  564  565  567  569  582  587  588  590  591  597  598  602  605  606  607  609  615  619  621  624  625  635
   1    2    4    4    2    2    5    2    4    4    4    4    2    2    4    4    4    3    3    4    5    4    5    1    5
 636  640  642  645  646  647  652  656  657  662  665  671  683  686  694  695  701  705  713  716  717  718  723  729  730
   4    2    4    2    5    5    4    4    4    4    5    2    4    5    2    2    2    2    4    2    2    2    4    2    2
 732  739  745  748  750  751  754  760  766  768  769  770  773  783  790  793  796  807  810  812  815  818  821  823  825
   2    4    3    1    1    3    3    4    5    2    2    2    2    5    5    5    1    1    5    3    3    2    5    5    1
 827  832  833  839  840  846  849  850  859  863  870  873  881  889  896  898  899  902  912  913  914  915  921  922  928
   5    1    1    3    2    5    2    2    3    5    4    3    4    5    1    3    3    3    5    5    1    1    1    1    2
 930  931  932  933  947  948  952  955  956  959  965  966  969  975  979  985 1002 1007 1009 1010 1015 1017 1028 1030 1032
   3    5    5    2    2    4    4    4    5    1    5    5    5    5    3    2    3    2    2    3    2    2    5    2    4
1037 1043 1044 1045 1046 1047 1057 1062 1064 1066 1069 1071 1075 1077 1079 1082 1085 1087 1094 1097 1098 1102 1104 1109 1113
   2    5    4    4    3    2    3    3    5    2    2    3    3    3    3    3    3    2    4    5    5    3    3    4    2
1114 1115 1119 1120 1123 1129 1136 1139 1147 1148 1158 1161 1162 1166 1167 1169 1173 1174 1175 1184 1185 1192 1193 1197 1201
   2    3    4    5    3    3    4    4    3    5    5    2    2    4    4    2    5    5    2    5    2    2    2    2    2
1207 1214 1216 1217 1218 1219 1222 1228 1229 1235 1242 1243 1246 1266 1267 1268 1273 1275 1278 1286 1291 1294 1303 1305 1309
   3    5    5    5    5    5    5    5    3    2    5    5    3    3    5    5    3    5    5    5    2    5    5    3    3
1310 1312 1313 1315 1317 1321 1333 1339 1340 1343 1346 1347 1349 1350 1351 1373 1374 1378 1379 1380 1383 1386 1387 1388 1390
   5    3    3    3    2    2    2    5    3    3    5    5    3    2    2    2    2    2    2    2    5    5    5    5    5
1393 1396 1400 1401 1402 1404 1405 1408 1409 1410 1411 1413 1414 1416 1426 1427 1432 1436 1437 1438 1442 1446 1447 1448 1451
   5    5    5    5    5    2    3    5    5    5    5    3    5    3    2    2    5    5    5    5    2    3    3    2    5
1452 1460 1464 1466 1467 1477 1480 1481 1486 1487 1490 1492 1493 1494 1498 1499 1501 1503 1511 1515 1516 1521 1522 1524 1525
   3    3    3    3    3    3    2    2    3    3    5    2    3    2    5    5    3    3    5    3    3    3    2    2    3
1526 1528 1529 1532 1533 1534 1535 1536 1540 1541 1542 1546 1549 1550 1556 1559 1570 1572 1576 1578 1580 1582 1584 1589 1590
   2    3    3    3    3    3    3    3    3    3    3    2    2    3    3    3    3    3    3    2    3    3    3    3    3
1596 1598 1603 1604 1605 1608 1612 1613 1615 1616 1618 1619 1620 1621 1622 1626 1628 1630 1633 1635 1636 1638 1639 1640 1646
   3    3    3    3    3    3    2    2    2    3    3    3    3    3    3    3    3    3    3    3    3    3    2    2    3
1649 1651 1652 1656 1657 1660 1661 1663 1667 1668 1670 1673 1676 1688 1690 1691 1694 1695 1699 1703 1704 1708 1711 1717 1721
   2    3    3    3    3    3    3    3    3    3    3    3    3    3    2    2    2    2    3    3    3    3    2    3    3
1722 1724 1729 1735 1737 1738 1739 1743 1745 1746 1747 1748 1750 1758 1759 1761 1762 1763 1769 1774 1778 1785 1787 1794 1795
   3    3    3    2    3    3    3    2    3    3    2    3    2    3    3    3    3    2    3    3    3    3    3    3    3
1797 1802 1803 1804 1807 1815 1816 1818 1820 1828 1831 1836 1840 1841 1845 1846 1850 1852 1857 1859 1861 1862 1865 1867 1868
```

Cross Table

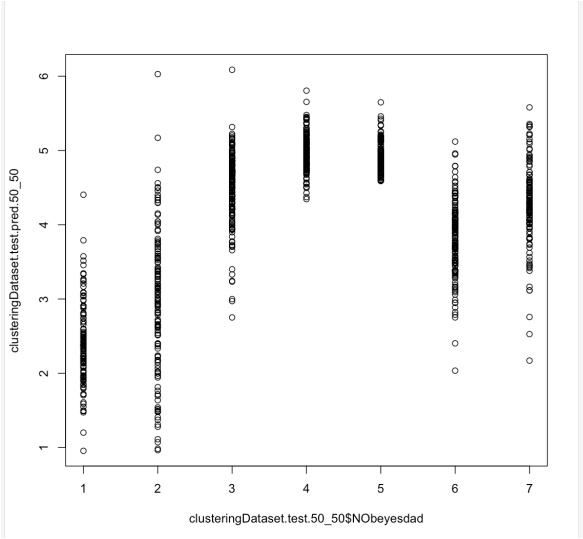
```
> # Cross-Tabulation to Compare Clusters and True Labels for 70-30 split
> clusteringDataset.test.crossTable.k5.70_30 <- CrossTable(x = as.factor(clusteringDataset.labels.70_30),
+                                                          y = as.factor(predicted.clusters.70_30),
+                                                          prop.chisq = FALSE)


   Cell Contents
|-------------------------|
|                       N |
|           N / Row Total |
|           N / Col Total |
|         N / Table Total |
|-------------------------|


Total Observations in Table:  633


                                    | as.factor(predicted.clusters.70_30)
as.factor(clusteringDataset.labels.70_30) |         1 |         2 |         3 |         4 |         5 | Row Total |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  1 |         3 |        24 |         5 |        32 |        10 |        74 |
                                    |     0.041 |     0.324 |     0.068 |     0.432 |     0.135 |     0.117 |
                                    |     0.081 |     0.197 |     0.030 |     0.400 |     0.044 |           |
                                    |     0.005 |     0.038 |     0.008 |     0.051 |     0.016 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  2 |        13 |        12 |        10 |        23 |        19 |        77 |
                                    |     0.169 |     0.156 |     0.130 |     0.299 |     0.247 |     0.122 |
                                    |     0.351 |     0.098 |     0.060 |     0.287 |     0.084 |           |
                                    |     0.021 |     0.019 |     0.016 |     0.036 |     0.030 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  3 |         0 |        25 |        32 |         0 |        48 |       105 |
                                    |     0.000 |     0.238 |     0.305 |     0.000 |     0.457 |     0.166 |
                                    |     0.000 |     0.205 |     0.190 |     0.000 |     0.212 |           |
                                    |     0.000 |     0.039 |     0.051 |     0.000 |     0.076 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  4 |         0 |        22 |        87 |         0 |         0 |       109 |
                                    |     0.000 |     0.202 |     0.798 |     0.000 |     0.000 |     0.172 |
                                    |     0.000 |     0.180 |     0.518 |     0.000 |     0.000 |           |
                                    |     0.000 |     0.035 |     0.137 |     0.000 |     0.000 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  5 |         0 |         0 |         0 |         0 |       109 |       109 |
                                    |     0.000 |     0.000 |     0.000 |     0.000 |     1.000 |     0.172 |
                                    |     0.000 |     0.000 |     0.000 |     0.000 |     0.482 |           |
                                    |     0.000 |     0.000 |     0.000 |     0.000 |     0.172 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  6 |        18 |        15 |        13 |        11 |        25 |        82 |
                                    |     0.220 |     0.183 |     0.159 |     0.134 |     0.305 |     0.130 |
                                    |     0.486 |     0.123 |     0.077 |     0.138 |     0.111 |           |
                                    |     0.028 |     0.024 |     0.021 |     0.017 |     0.039 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                                  7 |         3 |        24 |        21 |        14 |        15 |        77 |
                                    |     0.039 |     0.312 |     0.273 |     0.182 |     0.195 |     0.122 |
                                    |     0.081 |     0.197 |     0.125 |     0.175 |     0.066 |           |
                                    |     0.005 |     0.038 |     0.033 |     0.022 |     0.024 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|
                       Column Total |        37 |       122 |       168 |        80 |       226 |       633 |
                                    |     0.058 |     0.193 |     0.265 |     0.126 |     0.357 |           |
------------------------------------|-----------|-----------|-----------|-----------|-----------|-----------|


>
```

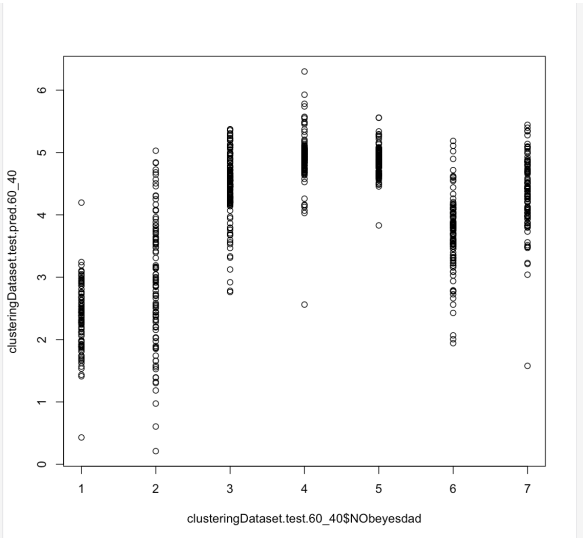
We plot clusteringDataset.test.50_50$NObeyesdad versus clusteringDataset.test.pred.50_50 values to understand the linearity. Looking at the plots, we can understand that there is some non-linearity present and it is not perfectly linear

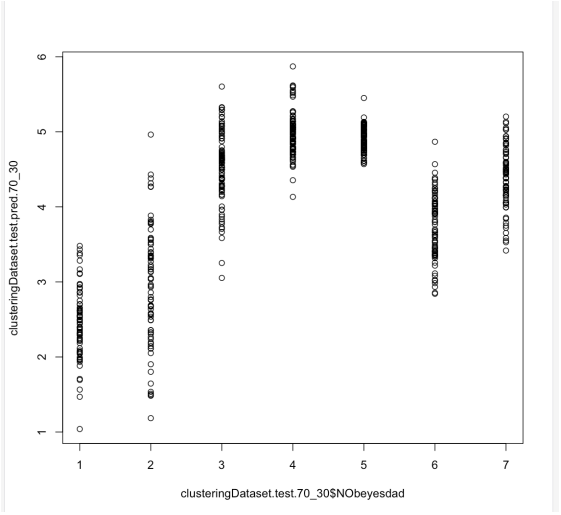
plot(clusteringDataset.test.50_50$NObeyesdad, clusteringDataset.test.pred.50_50)

plot(clusteringDataset.test.60_40$NObeyesdad, clusteringDataset.test.pred.60_40)

plot(clusteringDataset.test.70_30$NObeyesdad, clusteringDataset.test.pred.70_30)







Performance Measure

We can now look at some performance measures to further understand our results. To do this, we first compute the confusion matrix and then use it for computing our metrics as shown below.

```
> confusion_Matrix = clusteringDataset.test.crossTable.k5.50_50$t
> confusion_Matrix
   y
x     1   2   3   4   5   6   7
  1  18  21   0  24   5  42  19
  2  23  23   7  17  33  28   9
  3  71  53   2   1   4   1  54
  4  99   0  12   0   0   0  48
  5   0 154   0   0   0   0   0
  6  41  34   0   5  25  19  26
  7  57   9   1  23   9   0  38
> accuracy = sum(diag(confusion_Matrix)) / sum(confusion_Matrix)
> accuracy
[1] 0.09478673
> precision = diag(confusion_Matrix) / colSums(confusion_Matrix)
> precision
         1          2          3          4          5          6          7
0.05825243 0.07823129 0.09090909 0.00000000 0.00000000 0.21111111 0.19587629
> recall = diag(confusion_Matrix) / rowSums(confusion_Matrix)
> recall
         1          2          3          4          5          6          7
0.13953488 0.16428571 0.01075269 0.00000000 0.00000000 0.12666667 0.27737226
> specificity = sapply(1:nrow(confusion_Matrix), function(i) {
+   TN = sum(confusion_Matrix[-i,-i])
+   FP = sum(confusion_Matrix[-i, i])
+   TN / (TN + FP)
+ })
> specificity
[1] 0.6857451 0.7038251 0.9769850 0.9218750 0.9156493 0.9215470 0.8300654
> error = 1 - accuracy
> error
[1] 0.9052133
> sum(confusion_Matrix)
[1] 1055
>
```

The Data is collected from the results of the three splits 50-50, 60-40 and 70-30 and the performance measure has been displayed in the table below

Since we only have True Positives and False Positives, we calculate the Precision and Error performance metrics.

Split	Precision	Error	TP	FP
50-50	19.58%	90.52%	205	850
60-40	29.68	91.7%	244	600

70-30	2.04	96.05%	13	620

As we can see from our results above, our precision rates are quite low specially for the 70-30 split (which is only 2.04 %) and it improves to 29.68% for the 60-40 split and it's at 19.58% for the 50-50 split in the dataset.

This indicates that additional analysis is required on the attributes, and potentially, some of the existing attributes may need to be eliminated to achieve greater precision.

Understanding From the Project

this project on obesity classification using clustering and linear modeling reinforces several key data science concepts, highlighting the importance of data exploration for understanding the analysis process

The project likely involved initial exploration of the obesity dataset. This step is essential for comprehending the data's characteristics (distribution of obesity rates, correlations between variables), identifying potential issues (missing values, outliers), and informing feature selection for both clustering and modeling.

The project demonstrates the importance of evaluating model performance using metrics like accuracy, precision, and the confusion matrix. Calculating these metrics for the 50-50, 60-40, and 70-30 splits provided insights into the model's strengths and weaknesses. For example, a high false positive rate for a particular split might indicate overfitting and a need for further data exploration or model tuning.

Splitting the data into training and testing sets allows for unbiased evaluation of the model's generalizability. This project showcases the importance of this practice in data science. By using 50-50, 60-40, and 70-30 splits, I could assess the model's performance on unseen data, giving a more realistic picture of its effectiveness in real-world applications.

In conclusion, this project serves as a microcosm of the data science workflow. It emphasizes the significance of data exploration to guide analysis choices, understanding the assumptions of chosen algorithms, evaluating model performance with various metrics, and employing techniques like data splitting to ensure robust and generalizable results.