

# Problem

Diabetes mellitus is a chronic disease that occurs when the pancreas does not produce enough insulin, or when the body cannot effectively use the insulin, it produces. Our goal is to anticipate the occurrence of diabetes before it occurs in order to avoid its occurrence. by studying and analyzing the database we can extract useful results in diagnosing people.

# Data Mining Task

We will use classification and clustering to solve the problem. Initially we shall use the classification to create a model to predict whether a patient has diabetes or not using the Class Attribute. As for the Clustering, we will use it to divide patients into groups based on the similarity of their vital signs (measurements of basic body functions)

# Data

* Source: [**https://datahub.io/machine-learning/diabetes#resource-diabetes\_zip**](https://datahub.io/machine-learning/diabetes#resource-diabetes_zip)
* Number of objects:768
* Number of attributes:9

-Attributes characteristics:

Preg: Pregnancies

Plas: Plasma

Pres: Blood Pressure

Skin: Skin Thickness

Insu: Insulin

Mass: Mass

Pedi: Diabetes Pedigree Function

Age: Age

Class: Tested value (0 or 1)

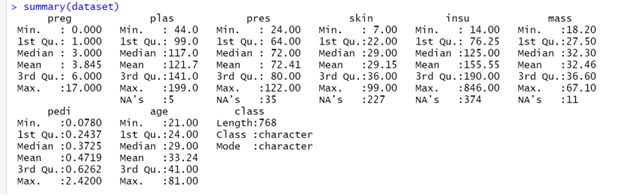
**Attributes Type possible values missing values**

|  |  |  |  |
| --- | --- | --- | --- |
| **Pregnancies** | Numeric (Ratio) | 0-17 | No |
| **Plasma** | Numeric(interval) | 44-199 | Yes |
| **Blood Pressure** | Numeric(interval) | 24-122 | Yes |
| **Skin Thickness** | Numeric(interval) | 7-99 | Yes |
| **Insulin** | Numeric(interval) | 14-846 | Yes |
| **Mass** | Numeric(interval) | 18-67 | Yes |
| **Diabetes Pedigree**  **Function** | Numeric (interval) | 0.078-2.42 | No |
| **Age** | Numeric (Ratio) | 21-81 | No |
| **Class** | Binary (Asymmetric) | 0, 1 | No |

missing value:-

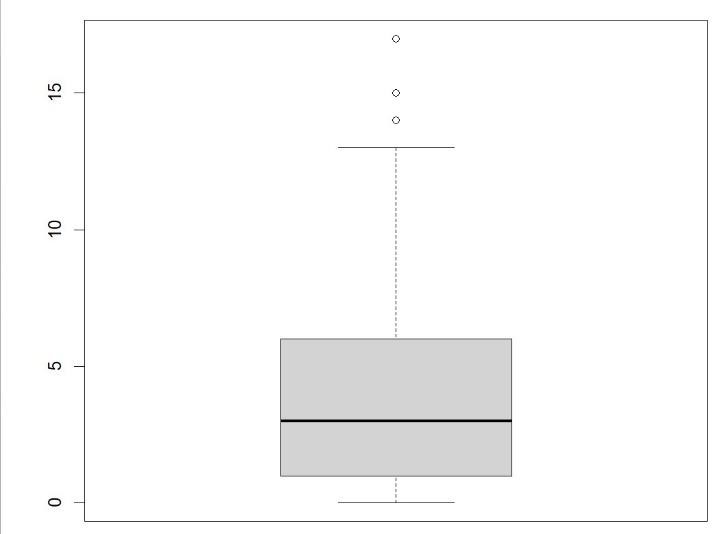


Summary:-



1-BOX PLOT:

1-preg Boxplot:



The box plot of pregnant attribute displays how the values are distributed, and show us that there are some outliers from 14 to 17.

Minimum= 0

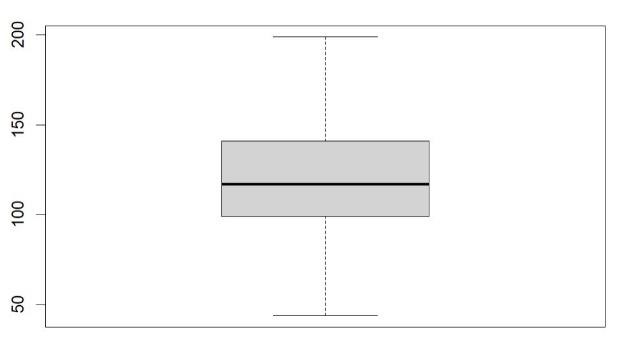
Q1= 1

Median= 3

Q3= 6

Maximum= 17

2- plas BoxPlot:



The box plot of Plasma attribute displays how the values are distributed, and show us that we do not have any outliers.

Minimum= 44

Q1= 99

Median= 117

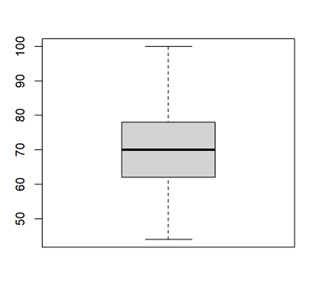
Q3= 141

Maximum= 199

3

-

pres Boxplot:



The box plot of blood pressure attribute displays how the values are distributed, and show us that we have outliers from 24 -38 and 106 -122

Minimum= 24

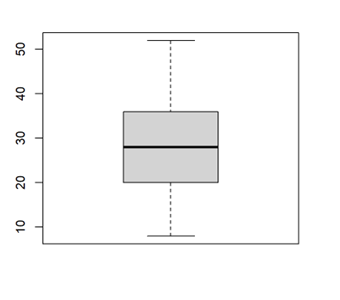
Q1= 64

Median= 72

Q3= 80

Maximum= 122

4- skin



The box plot of skin attribute displays how the values are distributed, and show us that we have outliers from 60-99

Minimum= 7

Q1= 22

Median= 29

Q3= 36

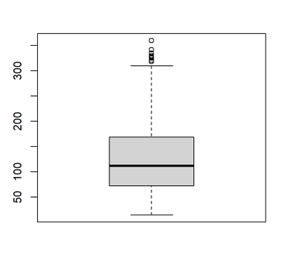
Maximum= 99

5

-

insulin

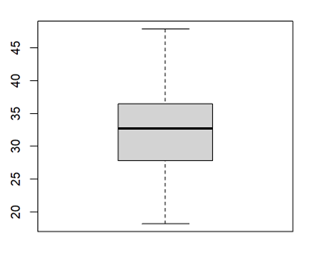
Boxplot:



The box plot of insulin attribute displays how the values are distributed, and show us that we have outliers from 465-846

Minimum= 14 , Q1= 76.25 , Median= 125 , Q3= 190 , Maximum= 846

6- mass



The box plot of mass attribute displays how the values are distributed, and show us that we have outliers from 52-67.10

Minimum= 18

Q1= 27.50

Median= 32.30

Q3= 36.60

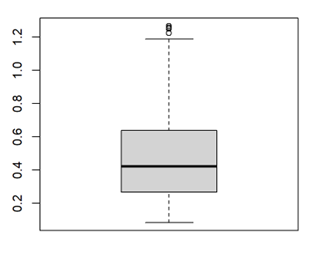
Maximum= 67.10

7

-

pedi

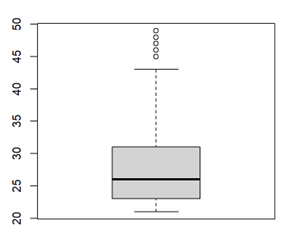
Boxplot:



The box plot of pedi attribute displays how the values are distributed, and show us that we have outliers from 1.213-2.4200

Minimum= 0.0780 , Q1= 0.2437 , Median= 0.3725 , Q3= 0.6262 , Maximum= 2.4200

8- age



The box plot of age attribute displays how the values are distributed, and show us that we have outliers from 67-81

Minimum= 21

Q1= 24

Median= 29

Q3= 41

Maximum= 81

2-HISTOGRAM:

1-preg

Chart, histogram

Description automatically generated

This histogram represents the frequency of pregnancy, so we observed that the highest frequency was at 0-5 and the lowest in the range 10-15

2-pals

Chart, histogram

Description automatically generated

This histogram represents the frequency of Plasma, so we observed that the highest frequency was approximately from 70-140.

1. pres

Chart, histogram

Description automatically generated

This histogram represents the frequency of blood pressure, so we observed that the highest frequency was within range 60-90

1. skin

Chart, histogram

Description automatically generated

This histogram represents the frequency of skin thickness, so we observed that the highest frequency was within range 20-40.

1. Insu

Chart, histogram

Description automatically generated

This histogram represents the frequency of Insulin, so we observed that the highest frequency was within range 0-200 and have a lowest frequency from 600-800.

6-mass

Chart, histogram

Description automatically generated

This histogram represents the frequency of mass, so we observed that the highest frequency was 25-40.

1. pedi

Chart, histogram

Description automatically generated

This histogram represents the frequency of Diabetes Pedigree Function, so we observed that highest frequency was within range 0.3-0.5

1. age

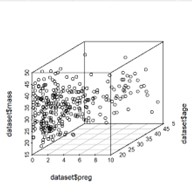
Chart, histogram

Description automatically generated

This histogram represents the frequency of Age, so we observed that the frequency was highest within range 21-45

3-scatter plot:

1. preg, mass and age



From the scatter plot of mass, age and Pregnancies we can notice the direct proportion between them.

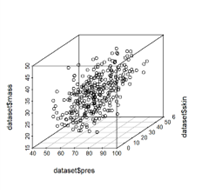
1. plas, insu and pedi

Diagram

Description automatically generated with medium confidence

From the scatter plot of plasma, insulin and Diabetes Pedigree Function we observed the correlation between them and we also noticed some values that are far away which could be outliers.

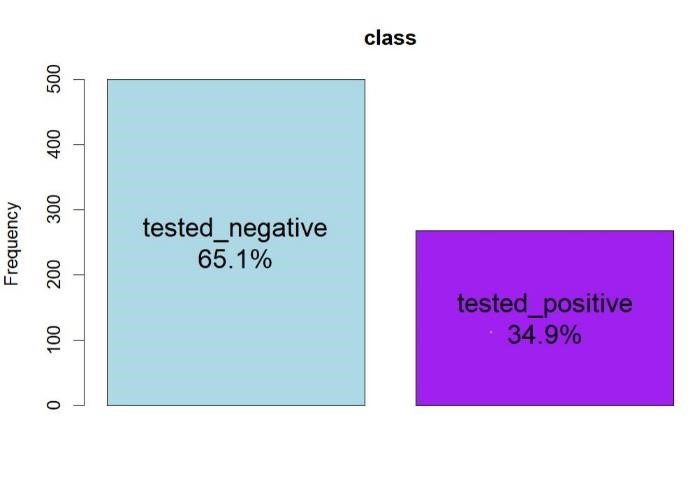
3-mass, skin and blood pressure:



From this graph of mass, skin and blood pressure we observed that when one of them increase the other two are also increased.

4-bar plot:

This bar plot represents the total values tested, and we found that 65.1% tested negative while 34.9% tested positive.



# Data preprocessing

For this dataset we used four types of preprocessing methods: filling missing data – omitting null insulin values – removing outliers – encoding.

Filling missing data: due to having null values we decided to fill the missing data with the average value We make it for the attributes:

* Plasma
* Blood Pressure
* Skin Thickness
* Mass

Removing null values: we noticed that the insulin column had too many null values and that filling them with the average value would not improve the accuracy of the classification

We make it for the attributes:

* Insulin

Removing outliers: because they will affect other calculations like mean and median

We make it for the attributes:

* Pregnancies
* Plasma
* Blood Pressure
* Skin Thickness
* Insulin
* Mass
* Diabetes Pedigree Function
* Age

Encoding: to simply the dataset and reduce its complixity

We make it for the attributes:

* Class

only 3 methods used (filling missing values-removing outliers-encoding) other preprocessing methods such as normalization and discretization will not add any value because our dataset values are already numeric and is in the interval of possible values in the human body

Data information and summary:

A screenshot of a computer

Description automatically generated with medium confidence

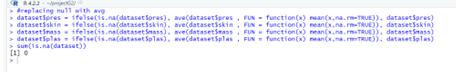
Filling missing Data:



R

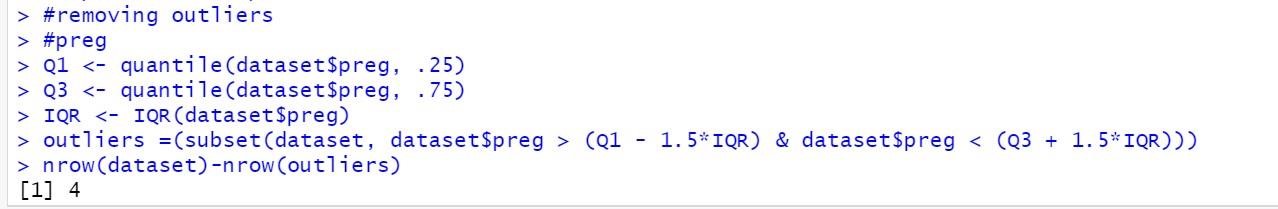
emoving

outliers:



E

ncoding:





# Data Mining Technique

We applied classification technique (which is a supervised method), because we have class label attribute outcome, and we well do the classification by predicting if a patient has diabetes or not by using the outcome as a class label (#Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function and age) for the patients. To do the classification we use decision tree technique. By splitting the data into three different sizes (70%, 30%), (75%, 25%), (80%, 20%) we test If the accuracy is acceptable, use the model to classify data tuples whose class labels are not known. we applied these packages: (party, caret) using the following methods: ConfusionMatrix: to measure how often the algorithm correctly or incorrectly predicted the outcome ctree(): to create conditional inference trees. Predict: to make prediction for new data. And for clustering technique (the unsupervised method) we used kmeans to represent each cluster by the center of the cluster. kmeans algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster. Its’ partition data into k clusters in a way that data points in the same cluster are similar and data points in the different clusters are farther apart. we applied this package and library: (factoextra , cluster) using the following methods:

1. set.seed() :initialize a pseudorandom number generator
2. scale(): centers and scales the columns of a numeric matrix
3. kmeans():cluster data based on similarity or similar groups
4. fviz\_cluster(): visualize clusters.
5. Silhouette(): computes the average silhouette for each cluster.
6. fviz\_silhouette(): Visualize silhouette information.
7. fviz\_nbclust(): Dertemines and visualize the optimal number of clusters

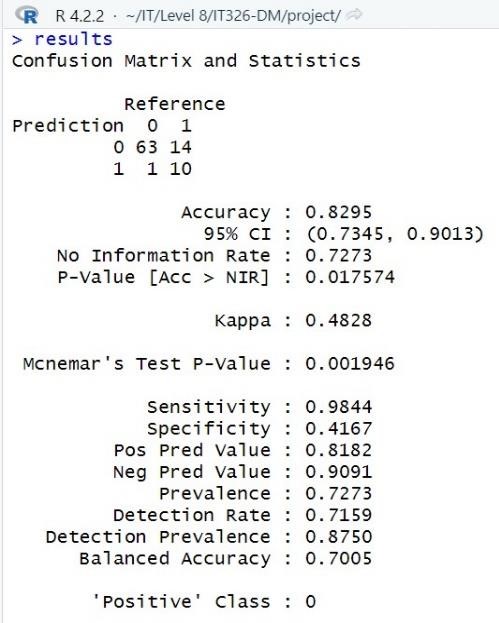
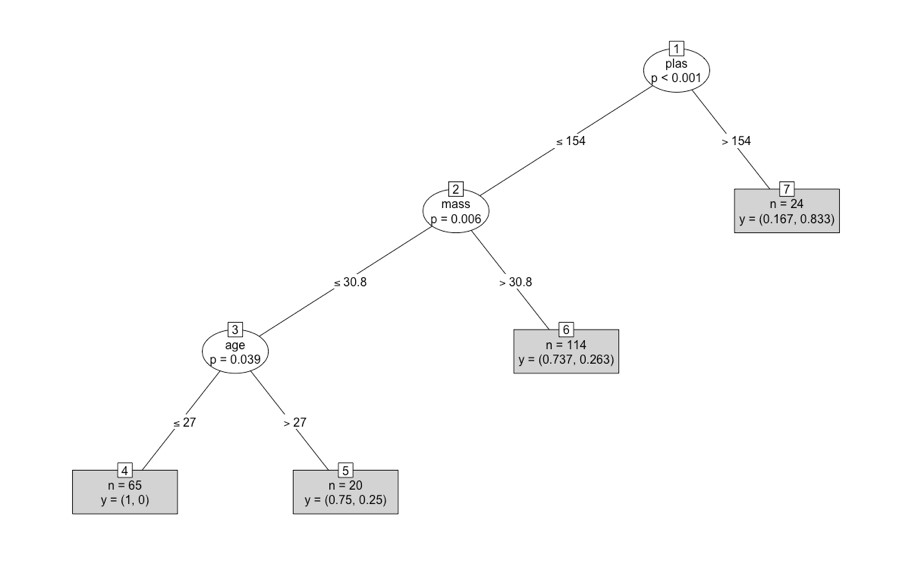
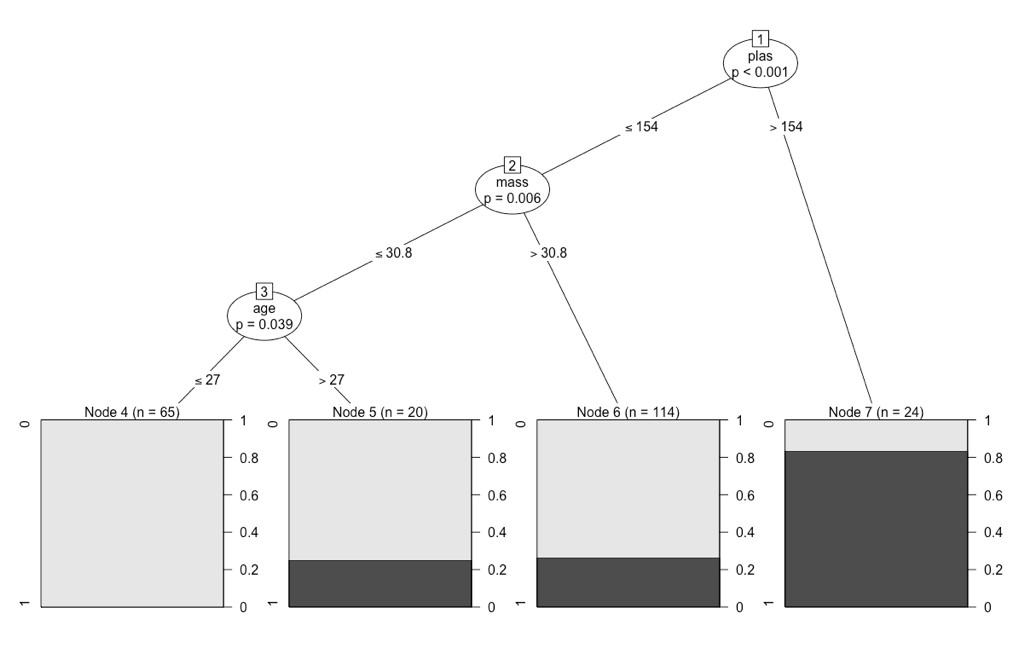
# Evaluation and Comparison

In the classification we use Confusion Matrix It’s a table that compares predicted values with the actual values, and it helps us to select the better classification model.

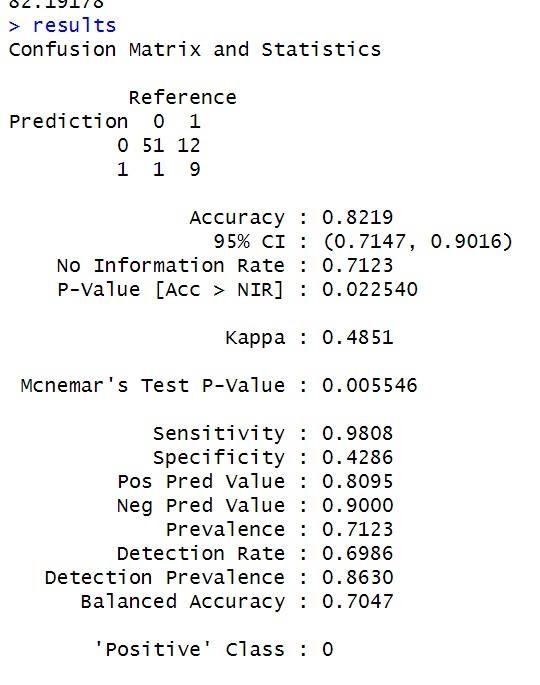
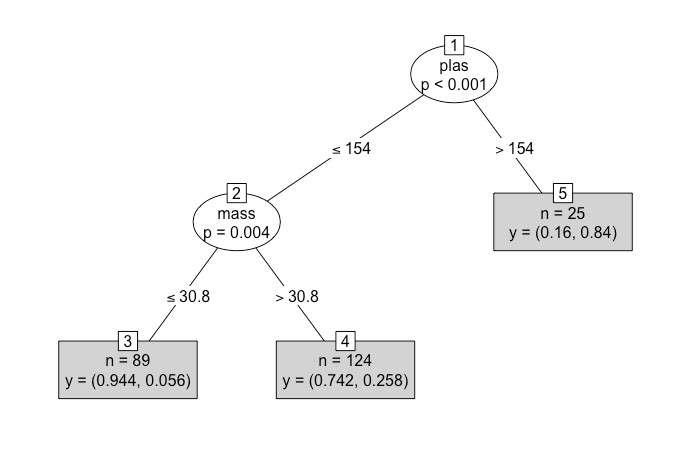
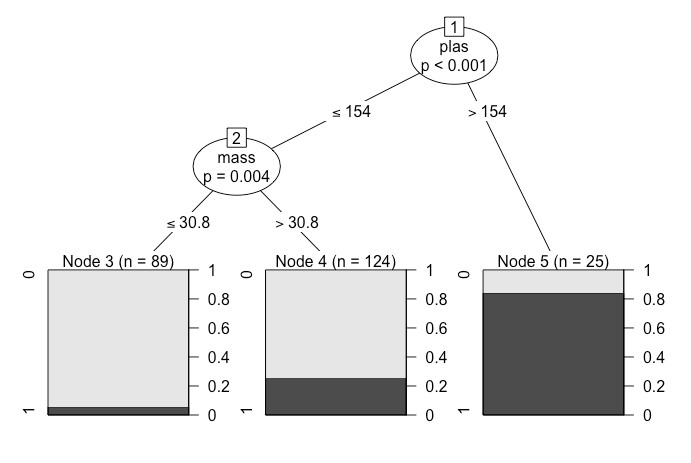
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Mining task | Comparison Criteria | | | |
| Classification | |  |  | | --- | --- | | **Train/Test** | **70,30** | | Accuracy | 0.8295 | | precision | TP/TP+FP 63/64=0.984 | | sensitivity | 0.9844 | | specificity | 0.4167 |      |  |  | | --- | --- | | **Train/Test** | **75, 25** | | Accuracy | 0.8219 | | precision | TP/TP+FP 51/52=0.981 | | sensitivity | 0.9808 | | specificity | 0.4286 | | | | |
|  | **Train/Test** | **80, 20** |  |
| Accuracy | 0.8393 |
| precision | TP/TP+FP 39/40=0.975 |
| sensitivity | 0.9750 |
| specificity | 0.5000 |
| Clustering | |  |  |  |  | | --- | --- | --- | --- | | Number of clusters | K=4 | K=3 | K=2 | | Silhouette width for each cluster | Silhouette width  Cluster1 = 0.14  Cluster2 = 0.20  Cluster3=0.09  Cluster4=0.21 | Silhouette width  Cluster1 = 0.16  Cluster2 = 0.29  Cluster3=0.12 | Silhouette width  Cluster1 = 0.09  Cluster2 = 0.3 | | Silhouette width for all clusters | Silhouette width for all clusters = 0.17 | Silhouette width for all clusters = 0.19 | Silhouette width for all clusters =0.195 | | Visualization |  | Next page |  | | | | |

Visualization of classification:

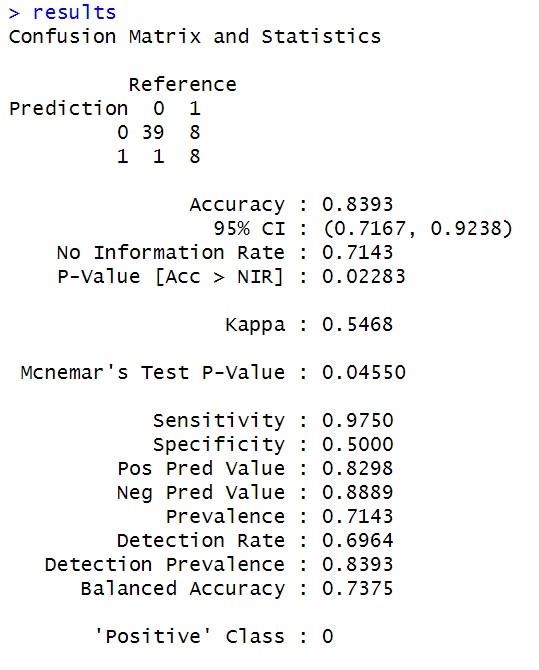
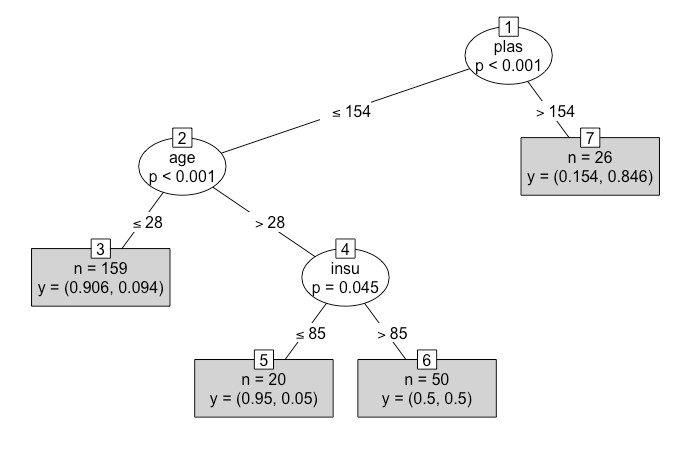
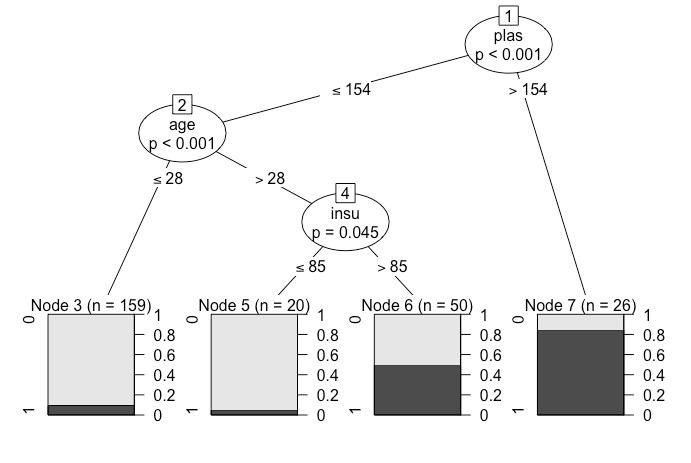
**(0.7,0.3):**



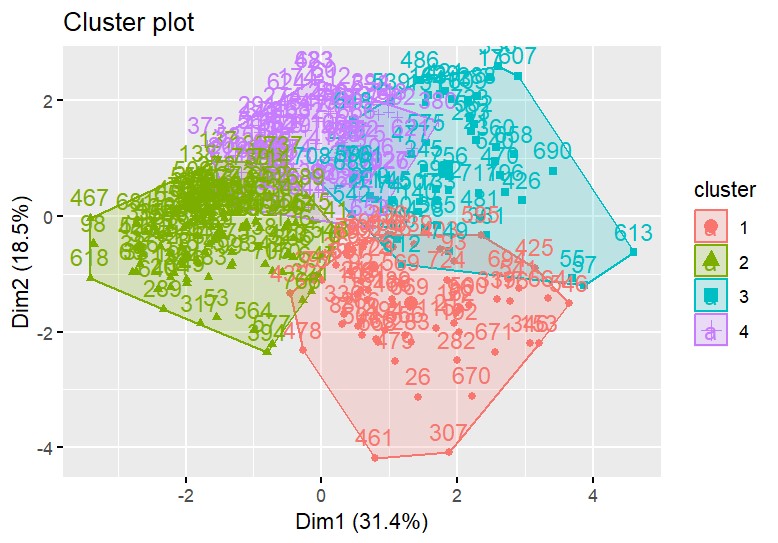
**(0.75, 0.25):**



**(0.8,0.2):**



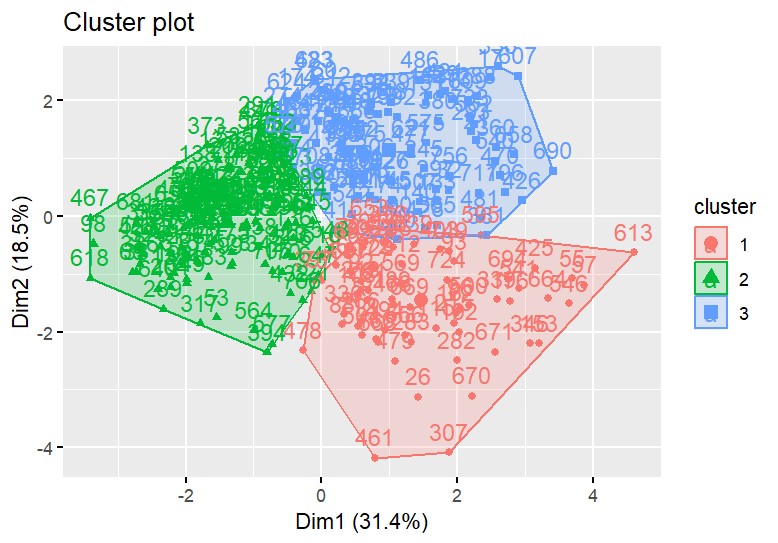
Visualization of clusters: k=4



Chart, histogram

Description automatically generated

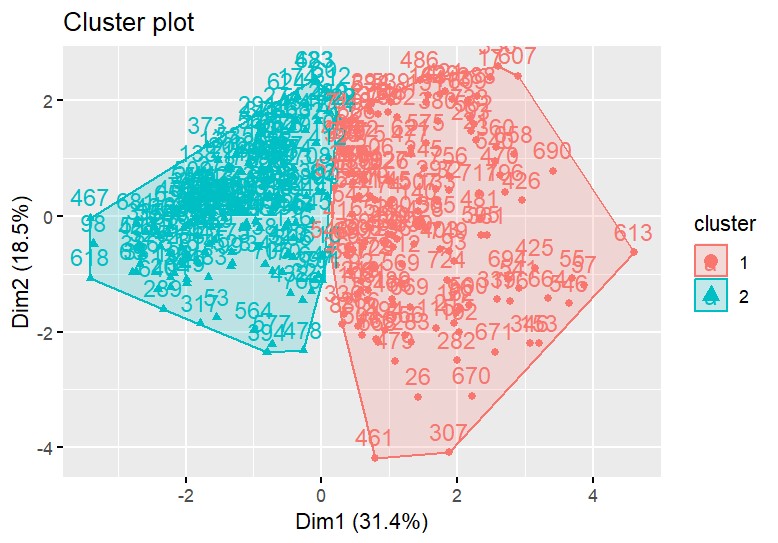
K=3

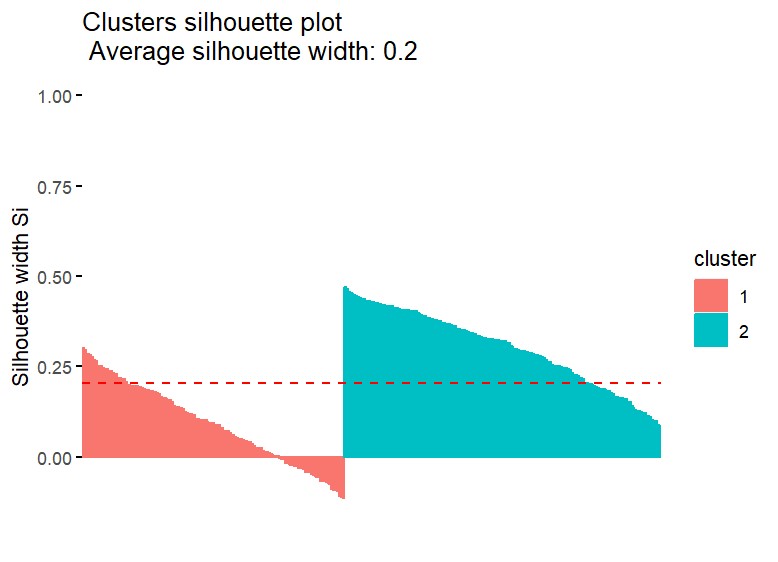


Chart, histogram

Description automatically generated

K=2





• Optimal Number of clusters

Chart, line chart

Description automatically generated

# Findings:

**In classification:**

At first, we Diabetes studied the dataset, what is each attribute does, and how it will affect each other. We analyzed our cases, to know how the attribute will help us to predict the best result. After that we did some preprocessing methods to our data, for example: cleaning data and transformation. Then we started our data mining techniques which are classification and clustering. And we came up with these results. The evaluation models result of the accuracy in 3 different Comparison Criteria:

-Training set 70% and Testing set 30% accuracy = 0.8295

-Training set 80% and Testing set 20% accuracy = 0.8392

-Training set 75% and Testing set 25% accuracy = 0.8219

The best evaluation model that has best accuracy in the binary trees was the first model which is

[80%,20%]. and the result if a patient has diabetes or not was affected by (Pregnancies, Plasma, Blood-Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function, and age) as a result it learned better than other evaluation models.

We find that only the (Plasma, Insulin ,Age) affect the classification.

**In clustering:**

We select 3 random numbers for dividing the clusters (2,3,4).and represented in the figures above. and we find out that k=2 is the optimal number of clusters.

When k=2 average for cluster#1= 0.09 and for cluster#2= 0.3, while the average for all clusters = 0.195(approximate value in graph=0.2). The objects in cluster#1= 141 and in cluster#2 = 170, cluster#2 have more objects than cluster#1 .

After using Silhouette width for clusters and find out that the average for all clusters is 0.195(approximate value in graph=0.2) which is the closest result compared to the average of other clusters in our dataset (k=3 average is 0.19(approximate value in graph=0.2) and k=4 average is

0.17 ) , we can figure out that k=2 is the optimal number of clusters. Also, to validate the result we used fviz\_nbclust() method which also shows that k=2 is the optimal number.

When k=3 average for cluster 1= 0.16 and for cluster 2= 0.29 and for cluster 3=0.12. while the average for all clusters = 0.19(approximate value in graph=0.2) and this average is acceptable but k=2 still have slightly better results, also the distance between objects from difference clusters short, so we do not analyze it.

When k=4 average for cluster 1= 0.14 and for cluster 2= 0.2 and for cluster 3=0.09 and for cluster

4= 0.21. while the average for all clusters = 0.17 and it’s the lowest value we have, and we can see from the figure clusters overlap in different points which makes the objects from difference clusters closer to each another. And for those reasons, we did not choose it for our data.

After studying the results of the clustering we can study the data in each cluster to identify the similarities between the nodes within the cluster which may improve with early diagnoses.

# Code

All attributes were used in both classification and clustering. Except that the class attribute in clustering was removed (all attributes were numeric, so no transformation was needed).

**- Preprocessing:**

only 3 methods used (filling missing values-removing outliers-encoding) other preprocessing methods such as normalization and discretization will not add any value because our dataset values are already numeric and is in the interval of possible values in the human body

#Replacing null with avg dataset$pres = ifelse(is.na(dataset$pres), ave(dataset$pres , FUN = function(x) mean(x,na.rm=TRUE)), dataset$pres) dataset$skin = ifelse(is.na(dataset$skin), ave(dataset$skin , FUN = function(x) mean(x,na.rm=TRUE)), dataset$skin) dataset$mass = ifelse(is.na(dataset$mass), ave(dataset$mass , FUN = function(x) mean(x,na.rm=TRUE)), dataset$mass) dataset$plas = ifelse(is.na(dataset$plas), ave(dataset$plas , FUN = function(x) mean(x,na.rm=TRUE)), dataset$plas) sum(is.na(dataset))

#Removing null insulin cells dataset = na.omit(dataset)

#Removing outliers

#preg

Q1 <- quantile(dataset$preg, .25)

Q3 <- quantile(dataset$preg, .75) IQR <- IQR(dataset$preg) outliers =(subset(dataset, dataset$preg > (Q1 - 1.5\*IQR) & dataset$preg < (Q3 +

1.5\*IQR))) nrow(dataset)-nrow(outliers) dataset <- subset(dataset, dataset$preg > (Q1 - 1.5\*IQR) & dataset$preg < (Q3 + 1.5\*IQR))

#plasma

Q1 <- quantile(dataset$plas, .25)

Q3 <- quantile(dataset$plas, .75) IQR <- IQR(dataset$plas) outliers =(subset(dataset, dataset$plas > (Q1 - 1.5\*IQR) & dataset$plas < (Q3 + 1.5\*IQR))) nrow(dataset)-nrow(outliers) dataset <- subset(dataset, dataset$plas > (Q1 - 1.5\*IQR) & dataset$plas < (Q3 + 1.5\*IQR))

#pressure

Q1 <- quantile(dataset$pres, .25)

Q3 <- quantile(dataset$pres, .75) IQR <- IQR(dataset$pres) outliers =(subset(dataset, dataset$pres > (Q1 - 1.5\*IQR) & dataset$pres < (Q3 +

1.5\*IQR))) nrow(dataset)-nrow(outliers) dataset <- subset(dataset, dataset$pres > (Q1 - 1.5\*IQR) & dataset$pres < (Q3 + 1.5\*IQR))

#skin Thickness

Q1 <- quantile(dataset$skin, .25)

Q3 <- quantile(dataset$skin, .75) IQR <- IQR(dataset$skin) outliers =(subset(dataset, dataset$skin > (Q1 - 1.5\*IQR) & dataset$skin < (Q3 +

1.5\*IQR))) nrow(dataset)-nrow(outliers) dataset <- subset(dataset, dataset$skin > (Q1 - 1.5\*IQR) & dataset$skin < (Q3 + 1.5\*IQR))

#insulin

Q1 <- quantile(dataset$insu, .25)

Q3 <- quantile(dataset$insu, .75)

IQR <- IQR(dataset$insu) outliers =(subset(dataset, dataset$insu > (Q1 - 1.5\*IQR) & dataset$insu < (Q3 +

1.5\*IQR))) nrow(dataset)-nrow(outliers) dataset <- subset(dataset, dataset$insu > (Q1 - 1.5\*IQR) & dataset$insu < (Q3 + 1.5\*IQR))

#mass

Q1 <- quantile(dataset$mass, .25)

Q3 <- quantile(dataset$mass, .75) IQR <- IQR(dataset$mass) outliers =(subset(dataset, dataset$mass > (Q1 - 1.5\*IQR) & dataset$mass < (Q3 +

1.5\*IQR))) nrow(dataset)-nrow(outliers) dataset <- subset(dataset, dataset$mass > (Q1 - 1.5\*IQR) & dataset$mass < (Q3 + 1.5\*IQR))

#pedi

Q1 <- quantile(dataset$pedi, .25)

Q3 <- quantile(dataset$pedi, .75) IQR <- IQR(dataset$pedi) outliers =(subset(dataset, dataset$pedi > (Q1 - 1.5\*IQR) & dataset$pedi < (Q3 +

1.5\*IQR))) nrow(dataset)-nrow(outliers) dataset <- subset(dataset, dataset$pedi > (Q1 - 1.5\*IQR) & dataset$pedi < (Q3 + 1.5\*IQR))

#age

Q1 <- quantile(dataset$age, .25)

Q3 <- quantile(dataset$age, .75) IQR <- IQR(dataset$age)

outliers =(subset(dataset, dataset$age > (Q1 - 1.5\*IQR) & dataset$age < (Q3 + 1.5\*IQR))) nrow(dataset)-nrow(outliers)

dataset <- subset(dataset, dataset$age > (Q1 - 1.5\*IQR) & dataset$age < (Q3 + 1.5\*IQR))

#Encoding dataset$class = factor(dataset$class, levels = c("tested\_negative", "tested\_positive"), labels = c(0,1))

**2- Data Mining Task:**

**Classification:**

#--------------------------------------------------------------------------------------classification1 set.seed(1234)

#Dividing the data into training and testing sets ind <- sample (2, nrow(dataset), replace=TRUE, prob=c(0.7, 0.3)) trainData <- dataset[ind==1,] testData <- dataset[ind==2,]

#Building decision tree install.packages('party') library(party) myFormula <- class ~ preg + mass + plas + pres + skin + insu + pedi + age dataset\_ctree <- ctree(myFormula, data=trainData)

# Check the prediction table(predict(dataset\_ctree), trainData$class) print(dataset\_ctree) plot(dataset\_ctree,type="simple") plot(dataset\_ctree)

# Predict on test data testPred <- predict (dataset\_ctree, newdata = testData) table(testPred, testData$class)

#--------------------------------------------------------------------------------------classification2 set.seed(1234)

#Dividing the data into training and testing sets ind <- sample (2, nrow(dataset), replace=TRUE, prob=c(0.75, 0.25)) trainData <- dataset[ind==1,] testData <- dataset[ind==2,]

#Building decision tree install.packages('party') library(party) myFormula <- class ~ preg + mass + plas + pres + skin + insu + pedi + age dataset\_ctree <- ctree(myFormula, data=trainData)

# Check the prediction table(predict(dataset\_ctree), trainData$class) print(dataset\_ctree) plot(dataset\_ctree,type="simple") plot(dataset\_ctree)

# Predict on test data testPred <- predict (dataset\_ctree, newdata = testData) table(testPred, testData$class)

#--------------------------------------------------------------------------------------classification3 set.seed(1234)

#Dividing the data into training and testing sets ind <- sample(2, nrow(dataset), replace=TRUE, prob=c(0.8, 0.2)) trainData <- dataset[ind==1,] testData <- dataset[ind==2,]

#Building decision tree install.packages('party') library(party) myFormula <- class ~ preg + mass + plas + pres + skin + insu + pedi + age dataset\_ctree <- ctree(myFormula, data=trainData)

# Check the prediction table(predict(dataset\_ctree), trainData$class) print(dataset\_ctree) plot(dataset\_ctree,type="simple") plot(dataset\_ctree)

# Predict on test data testPred <- predict(dataset\_ctree, newdata = testData) table(testPred, testData$class

**Clustring:**

#--------------------------------------------------------------------------------------------------Clustring1

# k-means clustering set a seed for random number generation to make the results reproducible set.seed(8953)

#Remove class

dataset3 <- dataset[, unlist(lapply(dataset, is.numeric))] dataset3 <- scale(dataset3)

# Run kmeans clustering to find 4 clusters kmeans.result <- kmeans(dataset3, 4)

# print the clustering result kmeans.result # Visualize clustering install.packages("factoextra") library(factoextra) fviz\_cluster(kmeans.result, data = dataset3)

#--------------------------------------------------------------------------------------------------Clustring2

# k-means clustering set a seed for random number generation to make the results reproducible

set.seed(8953)

#Remove class attribute

dataset3 <- dataset[, unlist(lapply(dataset, is.numeric))] dataset3 <- scale(dataset3)

# Run kmeans clustering to find 4 clusters

kmeans.result <- kmeans(dataset3, 3)

# Print the clusterng result

kmeans.result # Visualize clustering install.packages("factoextra") library(factoextra) fviz\_cluster(kmeans.result, data = dataset3)

#--------------------------------------------------------------------------------------------------Clustring3

# k-means clustering set a seed for random number generation to make the results reproducible set.seed(8953) #Remove class dataset3 <- dataset[, unlist(lapply(dataset, is.numeric))] dataset3 <- scale(dataset3)

# Run kmeans clustering to find 4 clusters kmeans.result <- kmeans(dataset3, 2)

# Print the clusterng result kmeans.result # Visualize clustering install.packages("factoextra") library(factoextra) fviz\_cluster(kmeans.result, data = dataset3)

**3- Evaluation:**

**Same code for all 3 classification sets:** install.packages('caret') library(caret)

#Show classification results in confusion matrix results <- confusionMatrix(testPred, testData$class) results

**Same code for all 3 clustring sets:** #Silhouette for each cluster sil <- silhouette(kmeans.result$cluster,dist(dataset3)) fviz\_silhouette(sil)

#--------------------------------------------------------------------------------------------

#Silhouette for all clusters fviz\_nbclust(dataset3, kmeans, method = "silhouette")+ labs(subtitle = "Silhouette method")