MysecondProject

JOSE IGNACIO SORDO SIERPE

7/1/2022

0.1 1. Introduction

As the second project of the Edx Data Science Capstone course called “Choose Your Own”, I am going to choose the data set called PimaIndiansDiabetes

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases, and is available at https://www.kaggle.com/uciml/pima-indians-diabetes-database

This dataset tries to predict the probability that a patient has diabetes or not based on certain diagnostic measures included in the dataset.

There are several restrictions in the database being taken in this case, that all patients have to be female, over 21 years of age and of Pima Indian descent.

In the first place we will make a statistical and graphic description of the data, I will analyze if there are statistically significant differences between the two groups (positive and negative) in each of the predictor variables.

Supervised learning machine learning techniques will be applied using 7 different binary classification algorithms and in the results phase we will stick with those that achieve better accuracy.

0.2 2. Methodology and General Description

First we will load the data set and the necessary libraries for the project.

```
# load libraries
library(mlbench)
```

```
## Warning: package 'mlbench' was built under R version 4.0.5
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.0.5
```

```
## Loading required package: lattice
```

```
library(Hmisc)
```

```
## Warning: package 'Hmisc' was built under R version 4.0.5
```

```
## Loading required package: survival
```

```
## Warning: package 'survival' was built under R version 4.0.5
```

```
##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster
```

```
## Loading required package: Formula
```

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

```r
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(Matrix)
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

```
## Loaded glmnet 4.1-3
```

```r
library (pROC)
```

```
## Warning: package 'pROC' was built under R version 4.0.5
```

```
## Type 'citation("pROC")' for a citation.
```

```
##
## Attaching package: 'pROC'
```

```
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
library(ROCR)
```

```
## Warning: package 'ROCR' was built under R version 4.0.5
```

```r
library(funModeling)
```

```
## Warning: package 'funModeling' was built under R version 4.0.5
```

```
## funModeling v.1.9.4 :)
## Examples and tutorials at livebook.datascienceheroes.com
##  / Now in Spanish: librovivodecienciadedatos.ai
```

```r
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.0.5
```

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

```
## The following object is masked from 'package:randomForest':
##
##     combine
## The following objects are masked from 'package:Hmisc':
##
##     src, summarize
```

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

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

```
library(smotefamily)
```

```
## Warning: package 'smotefamily' was built under R version 4.0.5
```

```
library(mlbench)
library(ggpubr)
```

```
## Warning: package 'ggpubr' was built under R version 4.0.5
```

```
library(onewaytests)
```

```
## Warning: package 'onewaytests' was built under R version 4.0.5
```

```
##
## Attaching package: 'onewaytests'
```

```
## The following object is masked from 'package:Hmisc':
##
##     describe
```

```
# load data
data(PimaIndiansDiabetes)
# rename dataset to keep code below generic
dataset <- PimaIndiansDiabetes
```

The methodology will follow the following successive steps: 1.- Description of the data set. 2.- Obtaining the statistical descriptions. 3.- Numerical and graphic analysis of the variables to check if they follow a normal distribution. 4.- Analysis of differences in means and variances to check if there are statistically significant differences between the variables. 5.- Generate a partition of 70% of the sample for training, reserving 30% for validation. 6.- A cross validation will be applied to the 70% partition. 7.- Training of the different algorithms.

0.2.1 2.1. Dataset Description

```
#dataset description.

head(dataset)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72      35       0 33.6    0.627  50      pos
## 2        1      85       66      29       0 26.6    0.351  31      neg
## 3        8     183       64       0       0 23.3    0.672  32      pos
## 4        1      89       66      23      94 28.1    0.167  21      neg
## 5        0     137       40      35     168 43.1    2.288  33      pos
## 6        5     116       74       0       0 25.6    0.201  30      neg
```

The data set is composed of 768 observations with 9 different variables. The variables are: Pregnant — Number of times pregnant Glucose — Plasma glucose concentration a 2 hours in an oral glucose tolerance test Pressure — Diastolic blood pressure (mm Hg) Triceps — Triceps skin fold thickness (mm) Insulin — 2-Hour serum insulin (mu U/ml) Mass — Body mass index (weight in kg/(height in m)^2) Pedigree — Diabetes pedigree function Age — Age (years) Diabetes — Class variable (positive or negative)

0.2.2 2.2. Descriptive Statistical

```
# cuantitative variables
df = dataset[,c(1:8)]

# statitical descriptives
estadisticaDescriptiva <- t(do.call(data.frame,
                       list(mean = apply(df, 2, mean),
                          Desv.Estandar = apply(df, 2, sd),
                          Mediana = apply(df, 2, median),
                          IQR = apply(df,2,IQR),
                          Min = apply(df, 2, min),
                          Max = apply(df, 2, max),
                          Rango = apply(df, 2,max)-apply(df, 2,min),
                          Cuartil1 = apply(df,2,quantile,probs = c(0.25)),
                          Cuartil3 = apply(df,2,quantile,probs = c(0.75)),
                          N = apply(df,2,length),
                          ErrorEstandar = apply(df,2,sd)/sqrt(apply(df,2,length)),
                          IC95MediaLower = apply(df,2,mean)-1.96*apply(df,2,sd)/sqrt(apply(df,2,length)),
                          IC95MediaUpper = apply(df,2,mean)+1.96*apply(df,2,sd)/sqrt(apply(df,2,length)),
                          Varianza = apply(df, 2, var)
                                      )))
estadisticaDescriptiva
```

```
##                  pregnant    glucose    pressure     triceps     insulin
## mean            3.8450521 120.894531  69.1054688  20.5364583    79.79948
## Desv.Estandar   3.3695781  31.972618  19.3558072  15.9522176   115.24400
## Mediana         3.0000000 117.000000  72.0000000  23.0000000    30.50000
## IQR             5.0000000  41.250000  18.0000000  32.0000000   127.25000
## Min             0.0000000   0.000000   0.0000000   0.0000000     0.00000
## Max            17.0000000 199.000000 122.0000000  99.0000000   846.00000
## Rango          17.0000000 199.000000 122.0000000  99.0000000   846.00000
## Cuartil1        1.0000000  99.000000  62.0000000   0.0000000     0.00000
## Cuartil3        6.0000000 140.250000  80.0000000  32.0000000   127.25000
## N             768.0000000 768.000000 768.0000000 768.0000000   768.00000
## ErrorEstandar   0.1215892   1.153712   0.6984425   0.5756261     4.15851
## IC95MediaLower  3.6067373 118.633255  67.7365214  19.4082312    71.64880
## IC95MediaUpper  4.0833669 123.155808  70.4744161  21.6646854    87.95016
## Varianza       11.3540563 1022.248314 374.6472712 254.4732453 13281.18008
##                     mass    pedigree         age
## mean           31.9925781  0.47187630  33.2408854
## Desv.Estandar   7.8841603  0.33132860  11.7602315
## Mediana        32.0000000  0.37250000  29.0000000
## IQR             9.3000000  0.38250000  17.0000000
## Min             0.0000000  0.07800000  21.0000000
## Max            67.1000000  2.42000000  81.0000000
## Rango          67.1000000  2.34200000  60.0000000
## Cuartil1       27.3000000  0.24375000  24.0000000
## Cuartil3       36.6000000  0.62625000  41.0000000
## N             768.0000000 768.00000000 768.0000000
## ErrorEstandar   0.2844951  0.01195579   0.4243608
## IC95MediaLower 31.4349677  0.44844295  32.4091382
## IC95MediaUpper 32.5501886  0.49530965  34.0726326
## Varianza       62.1599840  0.10977864 138.3030459
```

0.2.3 2.3. Analysys Normal Distribution

a. Pregnant

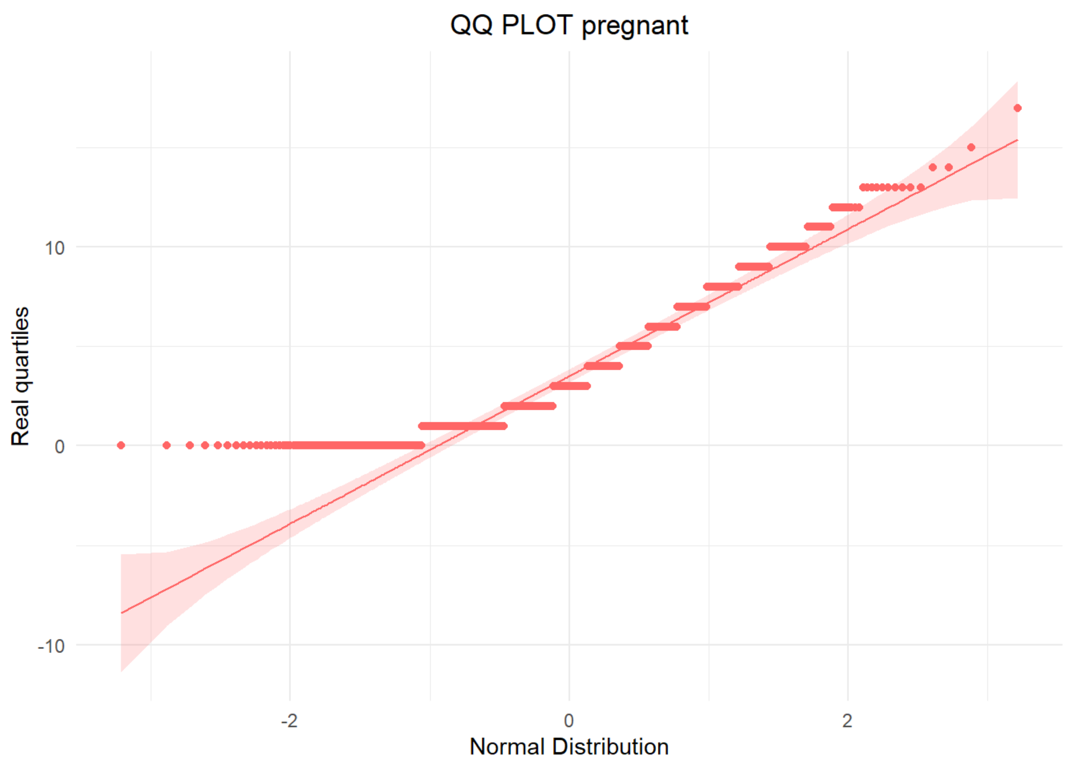
```
# Normality Test "Pregnant Vairable"

df <- dataset [1]
dvector= df$pregnant
shapiro.test(dvector)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dvector
## W = 0.90428, p-value < 2.2e-16
```

For a 95% confidence interval the variable does not follow a normal distribution

Graphycally:

```
#...hacemos el qqplot
name1 <- names(df)
ggqqplot(df, x = names(df),color = "#FF6666",add.params = list(color = "black"))+
  xlab("Normal Distribution") + ylab("Real quartiles") +
  theme_minimal() +
  ggtitle("QQ PLOT pregnant") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
df <- dataset [1]
dvector= df$pregnant
```

b. Glucose

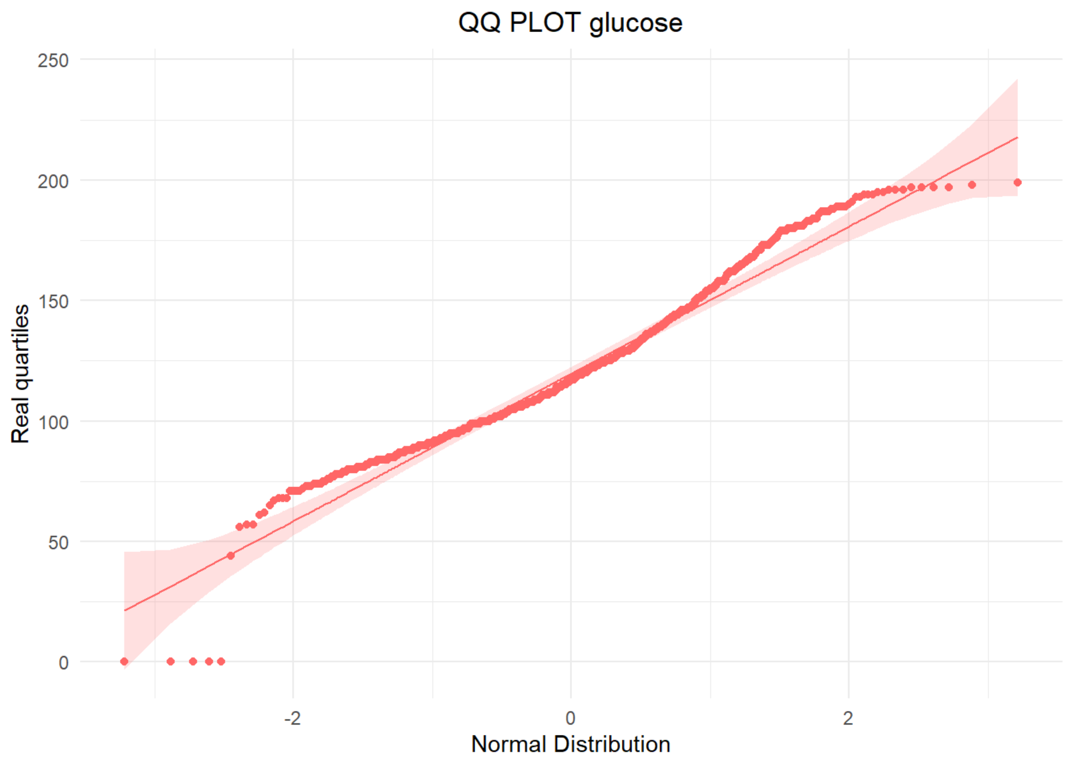
```
# Normality Test "Glucose Vairable"

df <- dataset [2]
dvector= df$glucose
shapiro.test(dvector)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dvector
## W = 0.9701, p-value = 1.986e-11
```

For a 95% confidence interval the variable does not follow a normal distribution

Graphycally:

```
#...hacemos el qqplot
name1 <- names(df)
ggqqplot(df, x = names(df),color = "#FF6666",add.params = list(color = "black"))+
  xlab("Normal Distribution") + ylab("Real quartiles") +
  theme_minimal() +
  ggtitle("QQ PLOT glucose") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
df <- dataset [2]
dvector= df$glucose
```

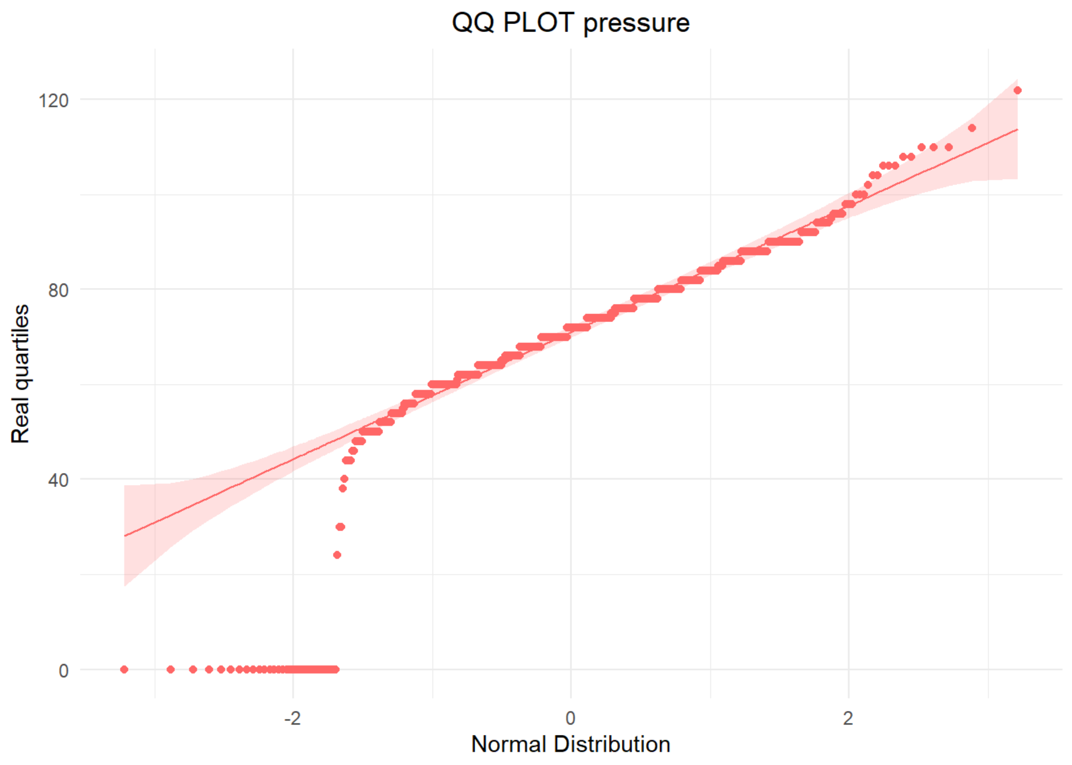
c. Pressure

```
# Normality Test "Pressure Variable"

df <- dataset [3]
dvector= df$pressure
shapiro.test(dvector)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dvector
## W = 0.81892, p-value < 2.2e-16
```

For a 95% confidence interval the variable does not follow a normal distribution Graphycally:

```
#...hacemos el qqplot
name1 <- names(df)
ggqqplot(df, x = names(df),color = "#FF6666",add.params = list(color = "black"))+
  xlab("Normal Distribution") + ylab("Real quartiles") +
  theme_minimal() +
  ggtitle("QQ PLOT pressure") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
df <- dataset [3]
dvector= df$pressure
```

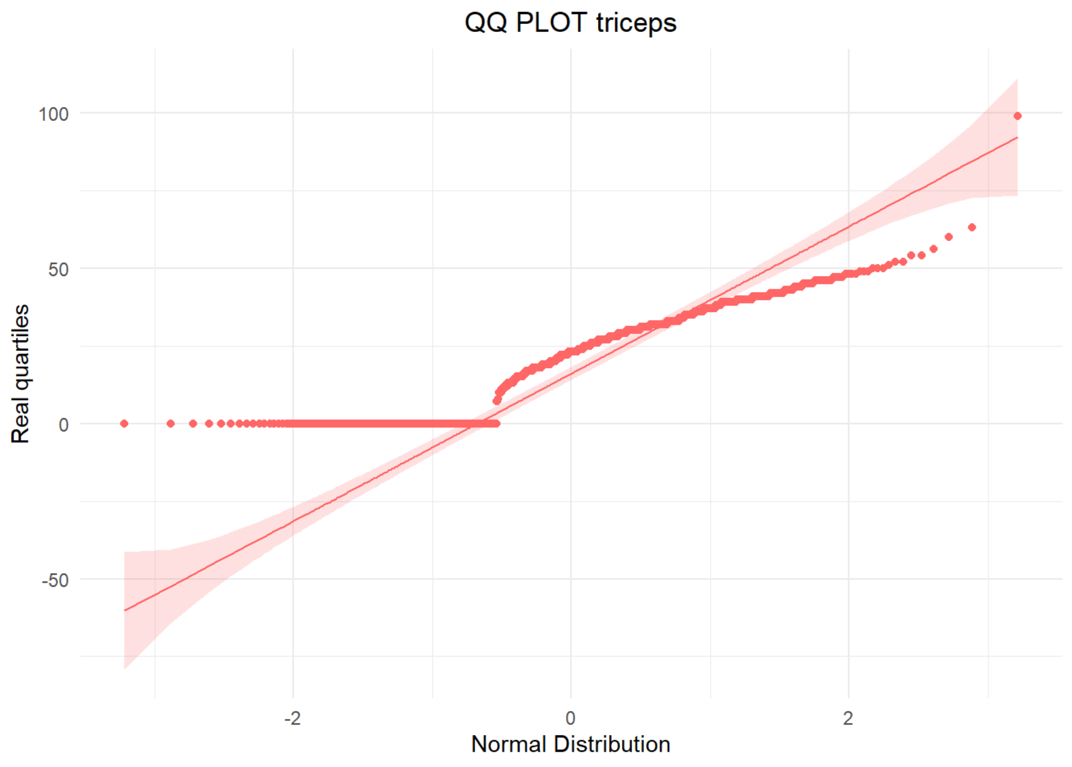
d. Triceps

```
# Normality Test "Triceps Variable"

df <- dataset [4]
dvector= df$triceps
shapiro.test(dvector)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dvector
## W = 0.90463, p-value < 2.2e-16
```

For a 95% confidence interval the variable does not follow a normal distribution Graphycally:

```
#...hacemos el qqplot
name1 <- names(df)
ggqqplot(df, x = names(df),color = "#FF6666",add.params = list(color = "black"))+
  xlab("Normal Distribution") + ylab("Real quartiles") +
  theme_minimal() +
  ggtitle("QQ PLOT triceps") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
df <- dataset [4]
dvector= df$triceps
```

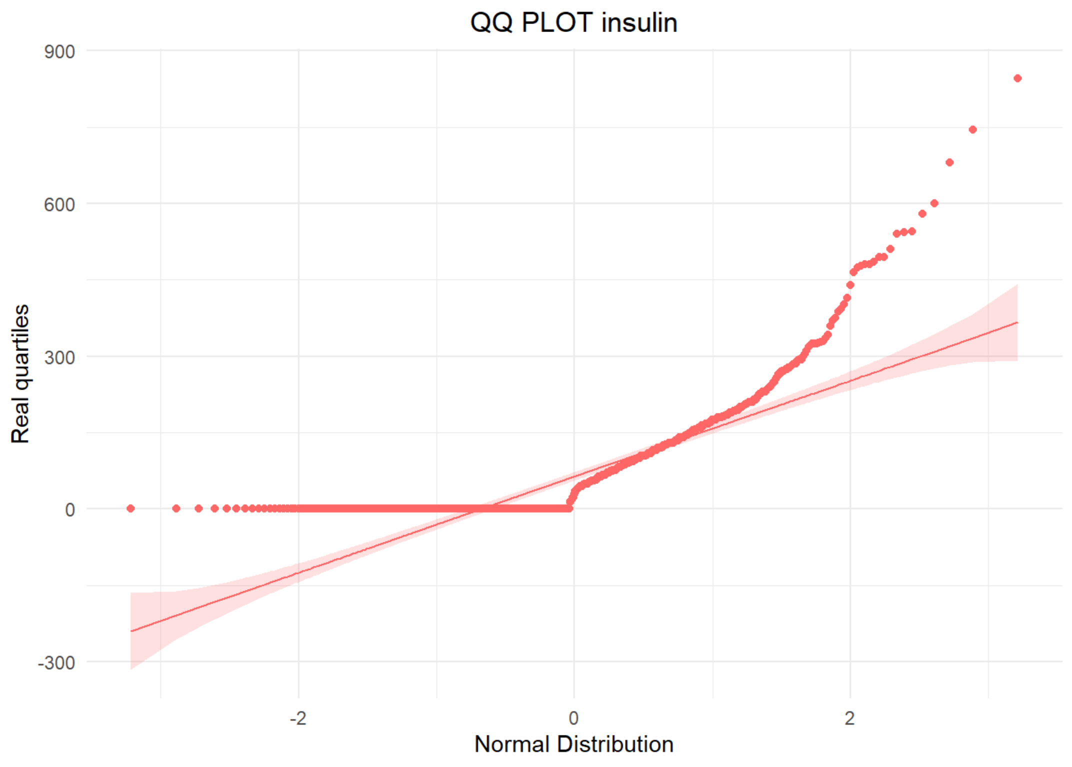
e. Insulin

```
# Normality Test "Insulin Variable"

df <- dataset [5]
dvector= df$insulin
shapiro.test(dvector)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dvector
## W = 0.72202, p-value < 2.2e-16
```

For a 95% confidence interval the variable does not follow a normal distribution Graphycally:

```
#...hacemos el qqplot
name1 <- names(df)
ggqqplot(df, x = names(df),color = "#FF6666",add.params = list(color = "black"))+
  xlab("Normal Distribution") + ylab("Real quartiles") +
  theme_minimal() +
  ggtitle("QQ PLOT insulin") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
df <- dataset [5]
dvector= df$insulin
```

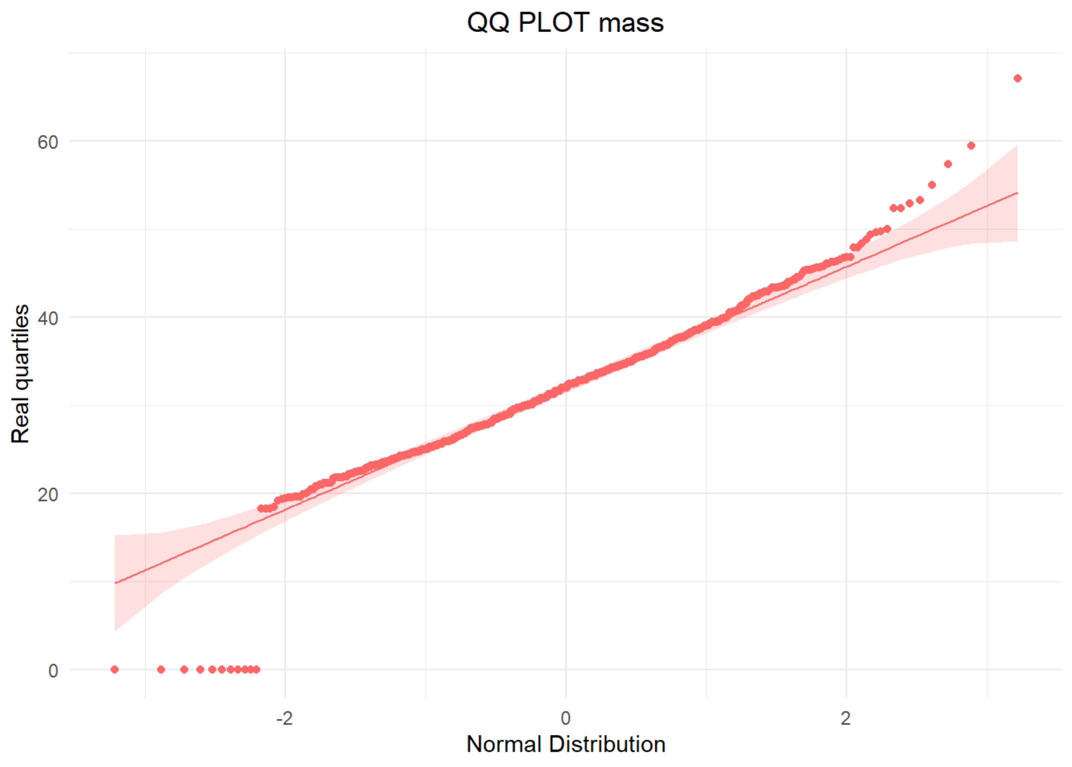
f. Mass

```
# Normality Test "Mass Variable"

df <- dataset [6]
dvector= df$mass
shapiro.test(dvector)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dvector
## W = 0.94999, p-value = 1.842e-15
```

For a 95% confidence interval the variable does not follow a normal distribution Graphycally:

```
#...hacemos el qqplot
name1 <- names(df)
ggqqplot(df, x = names(df),color = "#FF6666",add.params = list(color = "black"))+
  xlab("Normal Distribution") + ylab("Real quartiles") +
  theme_minimal() +
  ggtitle("QQ PLOT mass") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
df <- dataset [6]
dvector= df$mass
```

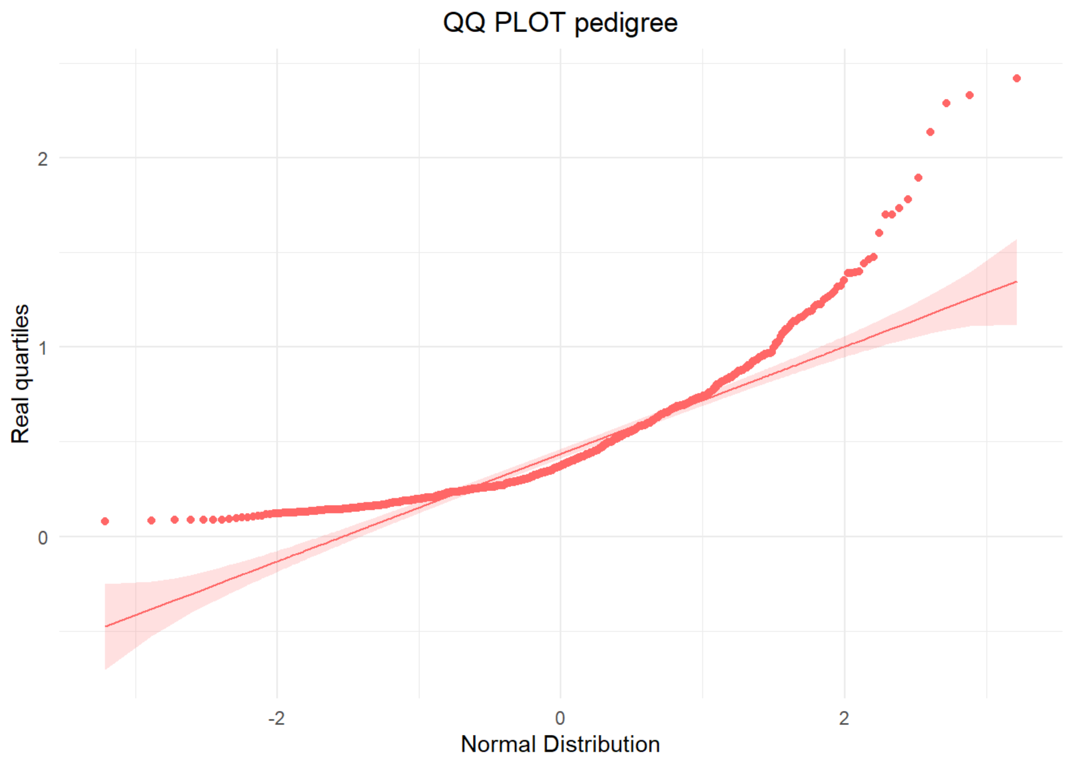
g. Pedigree

```
# Normality Test "Pedigree Variable"

df <- dataset [7]
dvector= df$pedigree
shapiro.test(dvector)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dvector
## W = 0.83652, p-value < 2.2e-16
```

For a 95% confidence interval the variable does not follow a normal distribution Graphycally:

```
#...hacemos el qqplot
name1 <- names(df)
ggqqplot(df, x = names(df),color = "#FF6666",add.params = list(color = "black"))+
  xlab("Normal Distribution") + ylab("Real quartiles") +
  theme_minimal() +
  ggtitle("QQ PLOT pedigree") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
df <- dataset [7]
dvector= df$pedigree
```

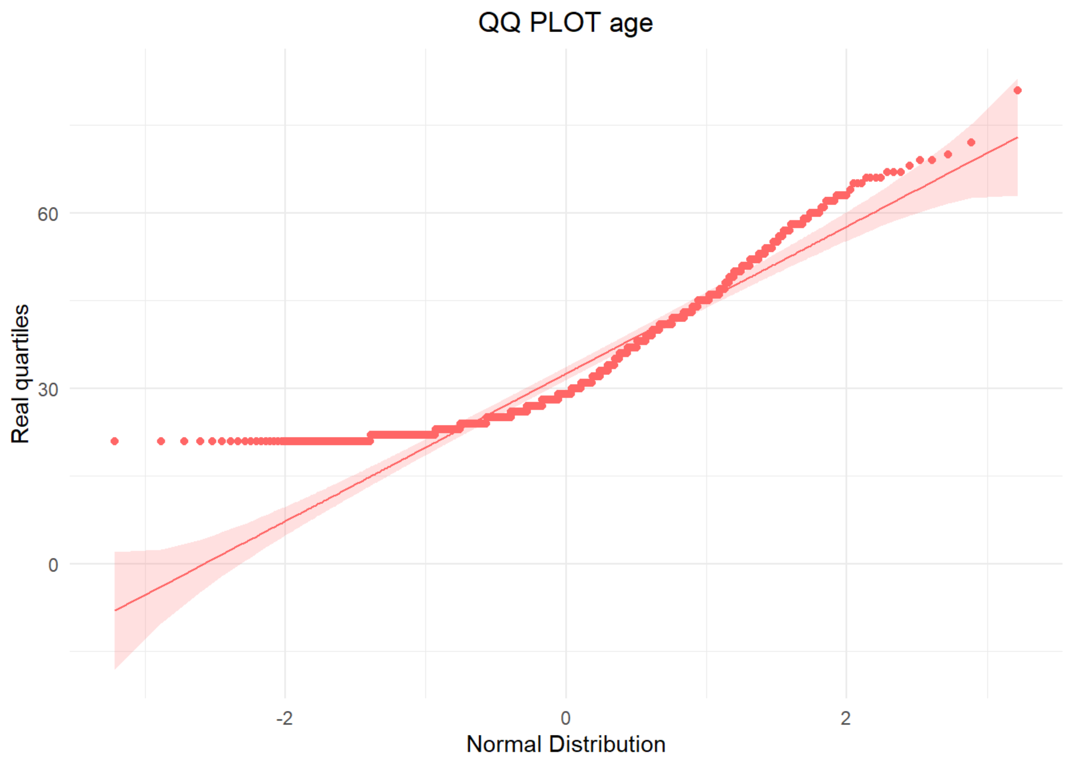
h. Age

```
# Normality Test "Pedigree Variable"

df <- dataset [8]
dvector= df$age
shapiro.test(dvector)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  dvector
## W = 0.87477, p-value < 2.2e-16
```

For a 95% confidence interval the variable does not follow a normal distribution Graphycally:

```
#...hacemos el qqplot
name1 <- names(df)
ggqqplot(df, x = names(df),color = "#FF6666",add.params = list(color = "black"))+
  xlab("Normal Distribution") + ylab("Real quartiles") +
  theme_minimal() +
  ggtitle("QQ PLOT age") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
df <- dataset [8]
dvector= df$age
```

0.2.4 2.4. Analysis of statistically significant differences.

As the variables do not follow a normal distribution, we will use non-parametric techniques.

a. Brown-Forsythe Test, to chek if there is equility of variances

```
bf.test(pregnant ~ diabetes, data = dataset)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : pregnant and diabetes
##
##   statistic  : 34.89219
##   num df     : 1
##   denom df   : 455.9558
##   p.value    : 6.821926e-09
##
##   Result     : Difference is statistically significant.
## -------------------------------------------------------------
```

```
bf.test(glucose ~ diabetes, data = dataset)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : glucose and diabetes
##
##   statistic  : 189.1048
##   num df     : 1
##   denom df   : 461.3317
##   p.value    : 2.644161e-36
##
##   Result     : Difference is statistically significant.
## -------------------------------------------------------------
```

```
bf.test(pressure~ diabetes, data = dataset)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : pressure and diabetes
##
##   statistic  : 2.934666
##   num df     : 1
##   denom df   : 471.3066
##   p.value    : 0.08735425
##
##   Result     : Difference is not statistically significant.
## -------------------------------------------------------------
```

```
bf.test(triceps ~ diabetes, data = dataset)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : triceps and diabetes
##
##   statistic  : 3.883182
##   num df     : 1
##   denom df   : 472.1018
##   p.value    : 0.04935586
##
##   Result     : Difference is statistically significant.
## -------------------------------------------------------------
```

```
bf.test(insulin ~ diabetes, data = dataset)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : insulin and diabetes
##
##   statistic  : 10.89591
##   num df     : 1
##   denom df   : 415.753
##   p.value    : 0.001046929
##
##   Result     : Difference is statistically significant.
## -------------------------------------------------------------
```

```
bf.test(mass ~ diabetes, data = dataset)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : mass and diabetes
##
##   statistic  : 74.29262
##   num df     : 1
##   denom df   : 573.4725
##   p.value    : 6.566238e-17
##
##   Result     : Difference is statistically significant.
## -------------------------------------------------------------
```

```
bf.test(pedigree ~ diabetes, data = dataset)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : pedigree and diabetes
##
##   statistic  : 20.94721
##   num df     : 1
##   denom df   : 454.5111
##   p.value    : 6.100481e-06
##
##   Result     : Difference is statistically significant.
## -------------------------------------------------------------
```

```
bf.test(age~ diabetes, data = dataset)
```

```
##
##   Brown-Forsythe Test (alpha = 0.05)
## -------------------------------------------------------------
##   data : age and diabetes
##
##   statistic  : 47.89662
##   num df     : 1
##   denom df   : 575.7773
##   p.value    : 1.201513e-11
##
##   Result     : Difference is statistically significant.
## -------------------------------------------------------------
```

There are difference statistically significant excepto in pression variable

b. Mann-Whitney Test to check if there are equility of means.

```
wilcox.test(pregnant~ diabetes, data = dataset, paired=FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  pregnant by diabetes
## W = 50985, p-value = 3.745e-08
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(glucose~ diabetes, data = dataset, paired=FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  glucose by diabetes
## W = 28391, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(pressure~ diabetes, data = dataset, paired=FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  pressure by diabetes
## W = 55415, p-value = 7.559e-05
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(triceps~ diabetes, data = dataset, paired=FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  triceps by diabetes
## W = 59814, p-value = 0.01296
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(insulin~ diabetes, data = dataset, paired=FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  insulin by diabetes
## W = 61927, p-value = 0.06566
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(mass~ diabetes, data = dataset, paired=FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  mass by diabetes
## W = 41866, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(pedigree~ diabetes, data = dataset, paired=FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  pedigree by diabetes
## W = 52769, p-value = 1.197e-06
## alternative hypothesis: true location shift is not equal to 0
```

```
wilcox.test(age~ diabetes, data = dataset, paired=FALSE)
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  age by diabetes
## W = 41950, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Only the insulin variable does not present statistically significant difference in equality of means for a 95% confidence interval.

As there are no variables that do not present statistically significant differences in both tests, we will use all the variables in the models.

0.2.5 2.5. Generate a partition of 70% of the sample for training, reserving 30% for validation.

```
# define an 70%/30train/test split of the dataset
split=0.70

trainIndex <- createDataPartition(dataset$diabetes, p=split, list=FALSE)
data_train <- dataset[ trainIndex,]
data_test  <- dataset[-trainIndex,]
```

The training sample is made up of 538 observations and the test sample of 230 observations.

0.2.6 2.6. Cross validation will be applied to the 70% partition

```
# set-up test options
control <- trainControl(method="cv", number=5)
seed <- (338)
metric <- "Accuracy"
```

0.2.7 2.7. Training algorithms

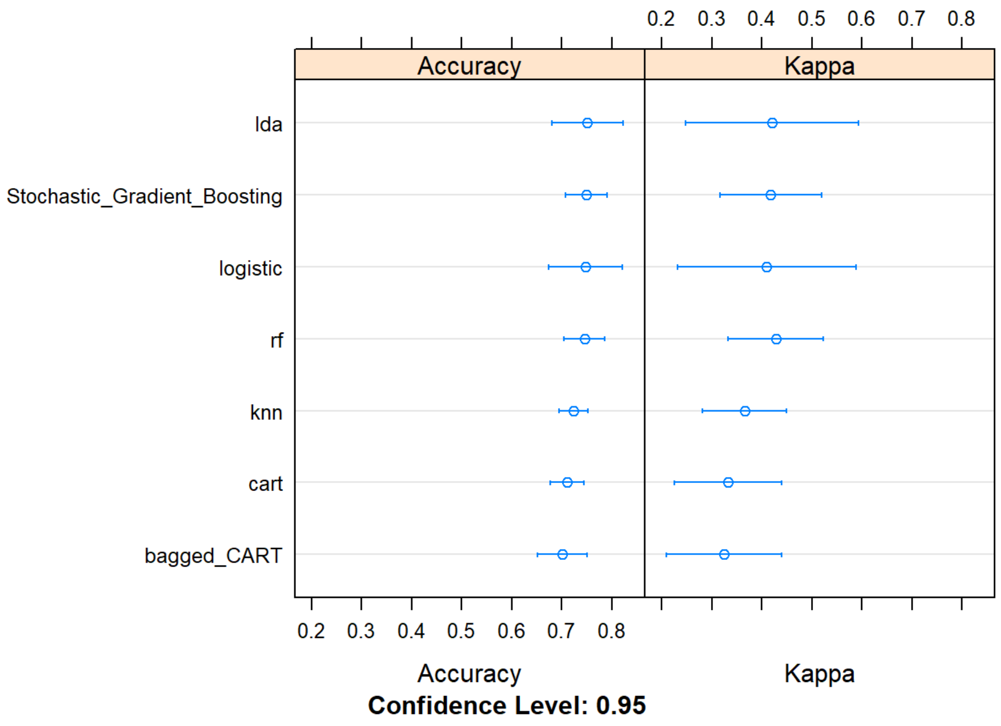
I am going to train the following algorithms: a) Linear Discriminant Analysis b) Logistic Regression c) Knn d) CART e) Bagged CART f) Random Forest g) Stochastic Gradient Boosting

```r
#  train algorithms

# Linear Discriminant Analysis
set.seed(seed)
fit.lda <- train(diabetes~., data=data_train, method="lda", metric=metric, preProc=c("center", "scale"), trControl=control)
# Logistic Regression
set.seed(seed)
fit.glm <- train(diabetes~., data=data_train, method="glm", metric=metric, trControl=control)
# kNN
set.seed(seed)
fit.knn <- train(diabetes~., data=data_train, method="knn", metric=metric, preProc=c("center", "scale"), trControl=control)
# CART
set.seed(seed)
fit.cart <- train(diabetes~., data=data_train, method="rpart", metric=metric, trControl=control)
# Bagged CART
set.seed(seed)
fit.treebag <- train(diabetes~., data=data_train, method="treebag", metric=metric, trControl=control)
# Random Forest
set.seed(seed)
fit.rf <- train(diabetes~., data=data_train, method="rf", metric=metric, trControl=control)
# Stochastic Gradient Boosting (Generalized Boosted Modeling)
set.seed(seed)
fit.gbm <- train(diabetes~., data=data_train, method="gbm", metric=metric, trControl=control, verbose=FALSE)

# Compare algorithms
results <- resamples(list(lda=fit.lda, logistic=fit.glm,
     knn=fit.knn, cart=fit.cart,
    bagged_CART=fit.treebag, rf=fit.rf, Stochastic_Gradient_Boosting=fit.gbm))
# Table comparison
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: lda, logistic, knn, cart, bagged_CART, rf, Stochastic_Gradient_Boosting
## Number of resamples: 5
##
## Accuracy
##                                   Min.   1st Qu.    Median      Mean   3rd Qu.
## lda                          0.6666667 0.7383178 0.7407407 0.7510211 0.7962963
## logistic                     0.6666667 0.7222222 0.7383178 0.7473174 0.7962963
## knn                          0.6851852 0.7196262 0.7314815 0.7230876 0.7314815
## cart                         0.6759259 0.6944444 0.7102804 0.7101073 0.7222222
## bagged_CART                  0.6481481 0.6759259 0.7009346 0.7008307 0.7314815
## rf                           0.7037037 0.7222222 0.7500000 0.7454656 0.7663551
## Stochastic_Gradient_Boosting 0.7037037 0.7383178 0.7500000 0.7490654 0.7570093
##                                   Max. NA's
## lda                          0.8130841    0
## logistic                     0.8130841    0
## knn                          0.7476636    0
## cart                         0.7476636    0
## bagged_CART                  0.7476636    0
## rf                           0.7850467    0
## Stochastic_Gradient_Boosting 0.7962963    0
##
## Kappa
##                                   Min.   1st Qu.    Median      Mean   3rd Qu.
## lda                          0.2123177 0.3952000 0.3986351 0.4203290 0.5308057
## logistic                     0.2123177 0.3435981 0.3986351 0.4100086 0.5308057
## knn                          0.2560778 0.3470301 0.3930233 0.3655550 0.4077156
## cart                         0.2190083 0.2732463 0.3601896 0.3325615 0.3716613
## bagged_CART                  0.2095532 0.2674419 0.3127258 0.3242892 0.3930233
## rf                           0.3260530 0.3759630 0.4485628 0.4279508 0.4735288
## Stochastic_Gradient_Boosting 0.3175355 0.3737458 0.4205087 0.4180083 0.4415897
##                                   Max. NA's
## lda                          0.5646867    0
## logistic                     0.5646867    0
## knn                          0.4239282    0
## cart                         0.4387022    0
## bagged_CART                  0.4387022    0
## rf                           0.5156465    0
## Stochastic_Gradient_Boosting 0.5366615    0
```

I will do a doplot to see the results of the algorithm training graphically

```
# Dot-plot comparison
dotplot(results)
```



1 3. Resultados

We make the confusion matrix for each of the algorithms on the test sample

```
# Matriz de confusion con datos de validacion
print("Accuracy Test Data Linear Discriminant Analysis")
```

```
## [1] "Accuracy Test Data Linear Discriminant Analysis"
```

```
pred <- predict(newdata=data_test,fit.lda)
real <- as.factor(data_test$diabetes)
cm_train_tot <- caret::confusionMatrix(data=pred,reference=real)
print(cm_train_tot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 132  30
##        pos  18  50
##
##                Accuracy : 0.7913
##                  95% CI : (0.733, 0.8419)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 2.878e-06
##
##                   Kappa : 0.5233
##
##  Mcnemar's Test P-Value : 0.1124
##
##             Sensitivity : 0.8800
##             Specificity : 0.6250
##          Pos Pred Value : 0.8148
##          Neg Pred Value : 0.7353
##              Prevalence : 0.6522
##          Detection Rate : 0.5739
##    Detection Prevalence : 0.7043
##       Balanced Accuracy : 0.7525
##
##        'Positive' Class : neg
##
```

```r
# Matriz de confusion con datos de validacion
print("Accuracy Test Data Linear Logistic Regression")
```

```
## [1] "Accuracy Test Data Linear Logistic Regression"
```

```r
pred <- predict(newdata=data_test,fit.glm)
real <- as.factor(data_test$diabetes)
cm_train_tot <- caret::confusionMatrix(data=pred,reference=real)
print(cm_train_tot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 132  29
##        pos  18  51
##
##                Accuracy : 0.7957
##                  95% CI : (0.7377, 0.8458)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 1.383e-06
##
##                   Kappa : 0.5347
##
##  Mcnemar's Test P-Value : 0.1447
##
##             Sensitivity : 0.8800
##             Specificity : 0.6375
##          Pos Pred Value : 0.8199
##          Neg Pred Value : 0.7391
##              Prevalence : 0.6522
##          Detection Rate : 0.5739
##    Detection Prevalence : 0.7000
##       Balanced Accuracy : 0.7588
##
##        'Positive' Class : neg
##
```

```r
# Matriz de confusion con datos de validacion
print("Accuracy Test Data Knn")
```

```
## [1] "Accuracy Test Data Knn"
```

```r
pred <- predict(newdata=data_test,fit.knn)
real <- as.factor(data_test$diabetes)
cm_train_tot <- caret::confusionMatrix(data=pred,reference=real)
print(cm_train_tot)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction neg pos
##        neg 127  33
##        pos  23  47
## 
##                Accuracy : 0.7565
##                  95% CI : (0.6958, 0.8105)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 0.000421
## 
##                   Kappa : 0.4472
## 
##  Mcnemar's Test P-Value : 0.229102
## 
##             Sensitivity : 0.8467
##             Specificity : 0.5875
##          Pos Pred Value : 0.7938
##          Neg Pred Value : 0.6714
##              Prevalence : 0.6522
##          Detection Rate : 0.5522
##    Detection Prevalence : 0.6957
##       Balanced Accuracy : 0.7171
## 
##        'Positive' Class : neg
## 
```

```
# Matriz de confusion con datos de validacion
print("Accuracy Test Data Cart")
```

```
## [1] "Accuracy Test Data Cart"
```

```
pred <- predict(newdata=data_test,fit.cart)
real <- as.factor(data_test$diabetes)
cm_train_tot <- caret::confusionMatrix(data=pred,reference=real)
print(cm_train_tot)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction neg pos
##        neg 134  39
##        pos  16  41
## 
##                Accuracy : 0.7609
##                  95% CI : (0.7004, 0.8145)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 0.000245
## 
##                   Kappa : 0.435
## 
##  Mcnemar's Test P-Value : 0.003012
## 
##             Sensitivity : 0.8933
##             Specificity : 0.5125
##          Pos Pred Value : 0.7746
##          Neg Pred Value : 0.7193
##              Prevalence : 0.6522
##          Detection Rate : 0.5826
##    Detection Prevalence : 0.7522
##       Balanced Accuracy : 0.7029
## 
##        'Positive' Class : neg
## 
```

```
# Matriz de confusion con datos de validacion
print("Accuracy Test Data Bagged Cart")
```

```
## [1] "Accuracy Test Data Bagged Cart"
```

```r
pred <- predict(newdata=data_test,fit.treebag)
real <- as.factor(data_test$diabetes)
cm_train_tot <- caret::confusionMatrix(data=pred,reference=real)
print(cm_train_tot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 126  26
##        pos  24  54
##
##                Accuracy : 0.7826
##                  95% CI : (0.7236, 0.8341)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 1.156e-05
##
##                   Kappa : 0.518
##
##  Mcnemar's Test P-Value : 0.8875
##
##             Sensitivity : 0.8400
##             Specificity : 0.6750
##          Pos Pred Value : 0.8289
##          Neg Pred Value : 0.6923
##              Prevalence : 0.6522
##          Detection Rate : 0.5478
##    Detection Prevalence : 0.6609
##       Balanced Accuracy : 0.7575
##
##        'Positive' Class : neg
##
```

```r
# Matriz de confusion con datos de validacion
print("Accuracy Test Data Random Forest")
```

```
## [1] "Accuracy Test Data Random Forest"
```

```r
pred <- predict(newdata=data_test,fit.rf)
real <- as.factor(data_test$diabetes)
cm_train_tot <- caret::confusionMatrix(data=pred,reference=real)
print(cm_train_tot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 128  25
##        pos  22  55
##
##                Accuracy : 0.7957
##                  95% CI : (0.7377, 0.8458)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 1.383e-06
##
##                   Kappa : 0.5456
##
##  Mcnemar's Test P-Value : 0.7705
##
##             Sensitivity : 0.8533
##             Specificity : 0.6875
##          Pos Pred Value : 0.8366
##          Neg Pred Value : 0.7143
##              Prevalence : 0.6522
##          Detection Rate : 0.5565
##    Detection Prevalence : 0.6652
##       Balanced Accuracy : 0.7704
##
##        'Positive' Class : neg
##
```

```
# Matriz de confusion con datos de validacion
print("Accuracy Test Data Stochastic Gradient Boosting")
```

```
## [1] "Accuracy Test Data Stochastic Gradient Boosting"
```

```
pred <- predict(newdata=data_test,fit.gbm)
real <- as.factor(data_test$diabetes)
cm_train_tot <- caret::confusionMatrix(data=pred,reference=real)
print(cm_train_tot)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction neg pos
##        neg 130  28
##        pos  20  52
##
##                Accuracy : 0.7913
##                  95% CI : (0.733, 0.8419)
##     No Information Rate : 0.6522
##     P-Value [Acc > NIR] : 2.878e-06
##
##                   Kappa : 0.529
##
##  Mcnemar's Test P-Value : 0.3123
##
##             Sensitivity : 0.8667
##             Specificity : 0.6500
##          Pos Pred Value : 0.8228
##          Neg Pred Value : 0.7222
##              Prevalence : 0.6522
##          Detection Rate : 0.5652
##    Detection Prevalence : 0.6870
##       Balanced Accuracy : 0.7583
##
##        'Positive' Class : neg
##
```

1.1 4.- Final Conclusions and limitations

As final conclusions, it is worth highlighting the good performance of all the models in the TRUE POSITIVE RATE (in this case the detection of negative cases of diabetes), exceeding in all cases the 80% success rate, which gives great security in the negative prediction of diabetes.

While the specificity in some cases is not as high as would be desired. This may be due to the fact that the dataset is umbalanced, with negative cases that exceed 3 times the rate of positive cases.

For future lines of research, the datasets could be balanced using artificial sampling techniques in order to balance the predictions. Another possibility would be with the use of the ROC curve, to estimate a different cut-off point.