Diabetes Prediction Project

José Javier Miñano Ramos

Contents

1 Overview

This project is related to the HarvardX Data Science Professional Certificate capstone project. The present report start with a general idea of the project and by representing its objectif.

Then, an exploratory data analysis is carried out in order to notice important trends in our data and then we will apply some of the machine learning algorithms that have been taught over the Professional Certificate to predict diagnosis of diabetes in a certain population of women. Finally the report ends with some concluding remarks.

1.1 Introduction

Some state-of-the-art machine learning techniques apply at the medicine field. There are relevant fields such as computer vision where these techniques are meant to change completely our idea of medicine and diagnostics. In this project, we will analyze a dataset which collects information of suspicious diabetes cases, labelled to perform supervised machine learning.

With diabetes, your body doesn't make enough insulin or can't use it as well as it should. When there isn't enough insulin or cells stop responding to insulin, too much blood sugar stays in your bloodstream. Over time, that can cause serious health problems, such as heart disease, vision loss, and kidney disease. According to the Diabetes International Federation, more than 500 million people suffer diabetes all over the world. As we can see, applying machine learning to this problem could optimize the way we diagnose diabetes and improve the life of millions of people.

Our objetive is to train some classification algorithms with several variables to predict two types of outcome: 1 (diabetic) or 0 (non diabetic).

Previously, in the movielens project we have introduced the terms “precise recommendation system”. As this is not a typical regression problem but a classification problem our loss function will be the percentaje of correctly clasificated cases.

2 Dataset:

First af all we are going to install the packages.

```
## Loading required package: tidyverse

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Loading required package: caret

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

## Loading required package: broom

## Loading required package: rpart

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following object is masked from 'package:dplyr':
##
##     count

## Loading required package: gam

## Loading required package: splines
```

```
## Loading required package: foreach

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## Loaded gam 1.20.2

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
##     transpose
```

We will start loading our dataset, (that we previously downloaded to our computer) using the read.csv function.

```
data = read.csv("diabetes_data.csv")
```

Now we will start exploring its information:

```
head(data)
```

```
##   Pregnancies Glucose BloodPressure SkinThickness Insulin  BMI
## 1           6     148            72            35       0 33.6
## 2           1      85            66            29       0 26.6
## 3           8     183            64             0       0 23.3
## 4           1      89            66            23      94 28.1
## 5           0     137            40            35     168 43.1
## 6           5     116            74             0       0 25.6
##   DiabetesPedigreeFunction Age Outcome
## 1                    0.627  50       1
## 2                    0.351  31       0
## 3                    0.672  32       1
## 4                    0.167  21       0
## 5                    2.288  33       1
## 6                    0.201  30       0
```

As we can see, we have 9 variables: Pregnancies, glucose, blood pressure, skin thickness, insulin, BMI, diabetes pedigree function and age. Finally we have the diagnosis: 1 for diabetic people and 0 for non diabetic people.

```
table(data$Outcome) #500 zeros, 268 ones
```

```
##
##   0   1
## 500 268
```

It is convenient to know how the outcome column is distributed, how many 0 and 1 appear and what is its proportion.

```
dim(data)
```

```
## [1] 768   9
```

We also known that our dataset dimension is 768 rows * 9 columns, it is a dataset much smaller than the movielens dataset but it is appropiate to apply machine learning algorithms and avoid computational problems caused by big datasets.

```
summary(data)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437           1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725           Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719           Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262           3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :81.00
##     Outcome
##  Min.   :0.000
##  1st Qu.:0.000
##  Median :0.000
##  Mean   :0.349
##  3rd Qu.:1.000
##  Max.   :1.000
```

Here we have the summary. We can see the most relevant characteristics of our distribution: for example, mean pregnancies over the dataset are between 3 and 4. However, it is more interesting to observe this variables grouped by diabetic or non diabetic. This is what we are going to do now:

```
diabetics <- data[data$Outcome == 1,]
  summary(diabetics)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
```

```
##  1st Qu.: 1.750   1st Qu.:119.0   1st Qu.: 66.00   1st Qu.: 0.00
##  Median : 4.000   Median :140.0   Median : 74.00   Median :27.00
##  Mean   : 4.866   Mean   :141.3   Mean   : 70.82   Mean   :22.16
##  3rd Qu.: 8.000   3rd Qu.:167.0   3rd Qu.: 82.00   3rd Qu.:36.00
##  Max.   :17.000   Max.   :199.0   Max.   :114.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0880           Min.   :21.00
##  1st Qu.:  0.0   1st Qu.:30.80   1st Qu.:0.2625           1st Qu.:28.00
##  Median :  0.0   Median :34.25   Median :0.4490           Median :36.00
##  Mean   :100.3   Mean   :35.14   Mean   :0.5505           Mean   :37.07
##  3rd Qu.:167.2   3rd Qu.:38.77   3rd Qu.:0.7280           3rd Qu.:44.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200           Max.   :70.00
##     Outcome
##  Min.   :1
##  1st Qu.:1
##  Median :1
##  Mean   :1
##  3rd Qu.:1
##  Max.   :1
```

```
non_diabetics <- data[data$Outcome == 0, ]
  summary(non_diabetics)
```

```
##   Pregnancies        Glucose    BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 93   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 2.000   Median :107   Median : 70.00   Median :21.00
##  Mean   : 3.298   Mean   :110   Mean   : 68.18   Mean   :19.66
##  3rd Qu.: 5.000   3rd Qu.:125   3rd Qu.: 78.00   3rd Qu.:31.00
##  Max.   :13.000   Max.   :197   Max.   :122.00   Max.   :60.00
##     Insulin            BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.00   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
##  1st Qu.:  0.00   1st Qu.:25.40   1st Qu.:0.2298           1st Qu.:23.00
##  Median : 39.00   Median :30.05   Median :0.3360           Median :27.00
##  Mean   : 68.79   Mean   :30.30   Mean   :0.4297           Mean   :31.19
##  3rd Qu.:105.00   3rd Qu.:35.30   3rd Qu.:0.5617           3rd Qu.:37.00
##  Max.   :744.00   Max.   :57.30   Max.   :2.3290           Max.   :81.00
##     Outcome
##  Min.   :0
##  1st Qu.:0
##  Median :0
##  Mean   :0
##  3rd Qu.:0
##  Max.   :0
```

As we can see above, the distributions differ from one case to another. Specifically, if we focus on the mean values:

```
sapply(diabetics, mean)
```

```
##             Pregnancies                 Glucose           BloodPressure
##                4.865672              141.257463               70.824627
```

```
##             SkinThickness                   Insulin                       BMI
##                 22.164179                100.335821                 35.142537
## DiabetesPedigreeFunction                       Age                   Outcome
##                  0.550500                 37.067164                  1.000000
```

```
sapply(non_diabetics, mean)
```

```
##               Pregnancies                   Glucose             BloodPressure
##                  3.298000                109.980000                 68.184000
##             SkinThickness                   Insulin                       BMI
##                 19.664000                 68.792000                 30.304200
## DiabetesPedigreeFunction                       Age                   Outcome
##                  0.429734                 31.190000                  0.000000
```

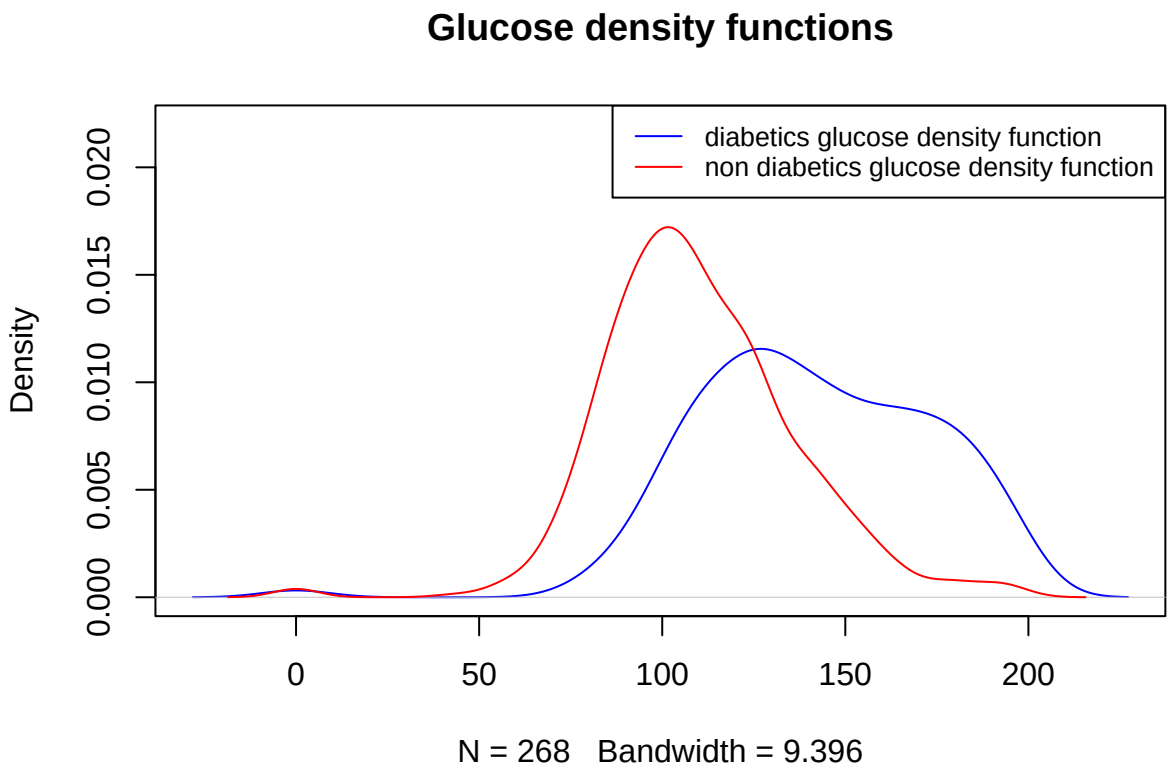
It is obvious that diabetes affects widely our body, for example, on average glucose levels are 30% higher for diabetic people. Pregnancies are also higher in diabetic women so we can suppose that more pregnancies imply more probabilities of having diabetes. We will later explore the importance of these variables using the principal component analysis (PCA).

3 Plots

Now we are going to visualize three important trends using plots:

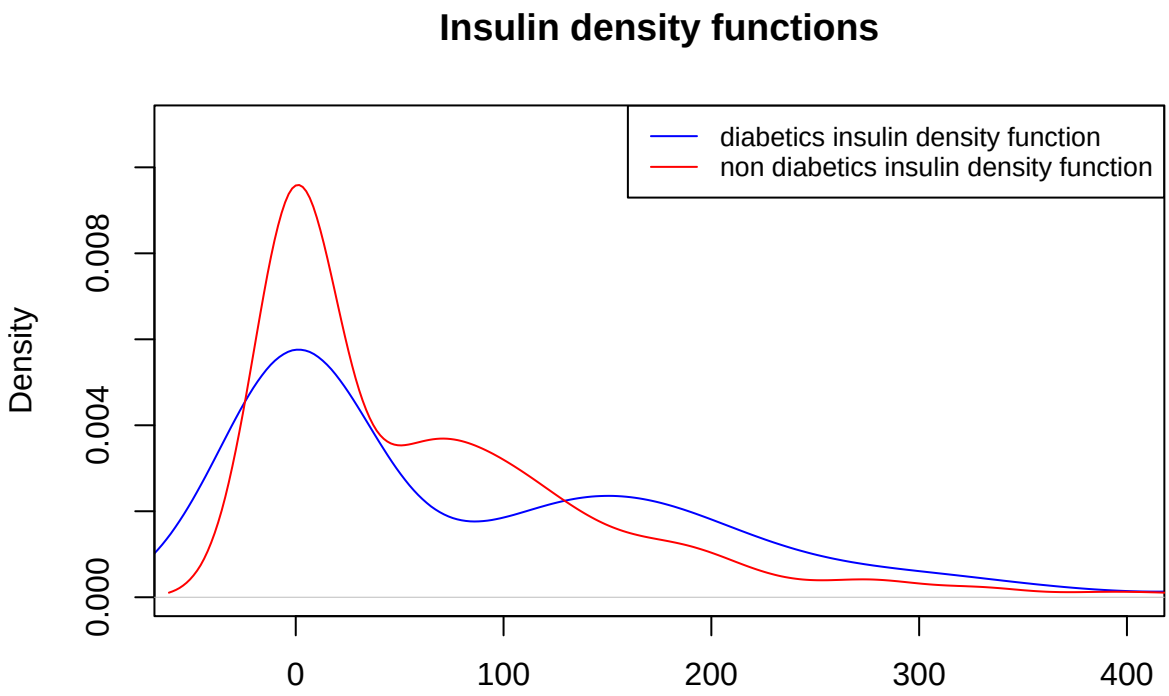
1.- In the plot below we can observe how the non diabetic density function follows a t-distribution centered at aproximately 100 while the diabetics function is right-winged and provoques a higher mean.

```
plot(density(diabetics$Glucose),ylim = c(0,0.022) ,
     col = "blue", main = "Glucose density functions")
lines(density(non_diabetics$Glucose), col = "red")
legend("topright", c("diabetics glucose density function",
                     "non diabetics glucose density function"),
       col = c("blue", "red"), lty =1, cex= 0.8)
```



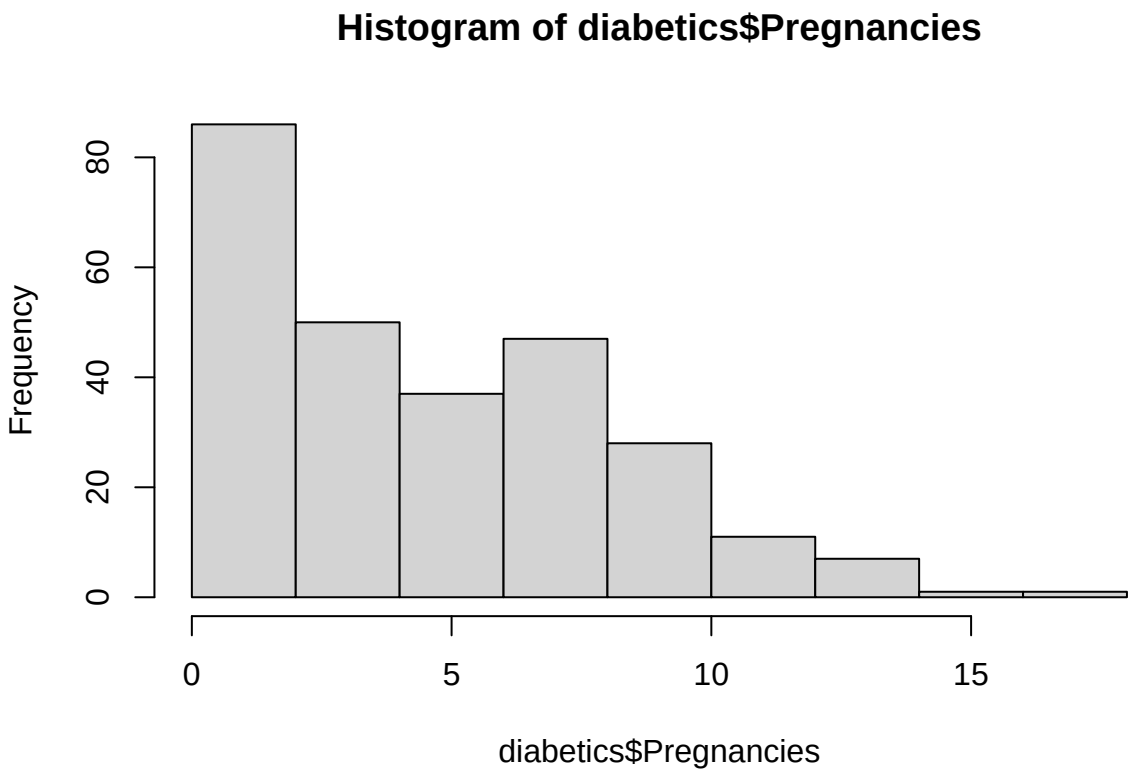
2.- In the second plot we have the insulin density function for both diabetics and non diabetics, we can see how the non diabetics function is left-winged and that reduces the mean. In particular, the non diabetics mean insulin is a 30% lower than for diabetics.

```
plot(density(diabetics$Insulin), ylim = c(0,0.011) , xlim = c(-50, 400),
     col = "blue", main = "Insulin density functions", xlab = "  ")
lines(density(non_diabetics$Insulin), col = "red")
legend("topright", c("diabetics insulin density function",
                     "non diabetics insulin density function"),
       col = c("blue", "red"), lty =1, cex= 0.8)
```

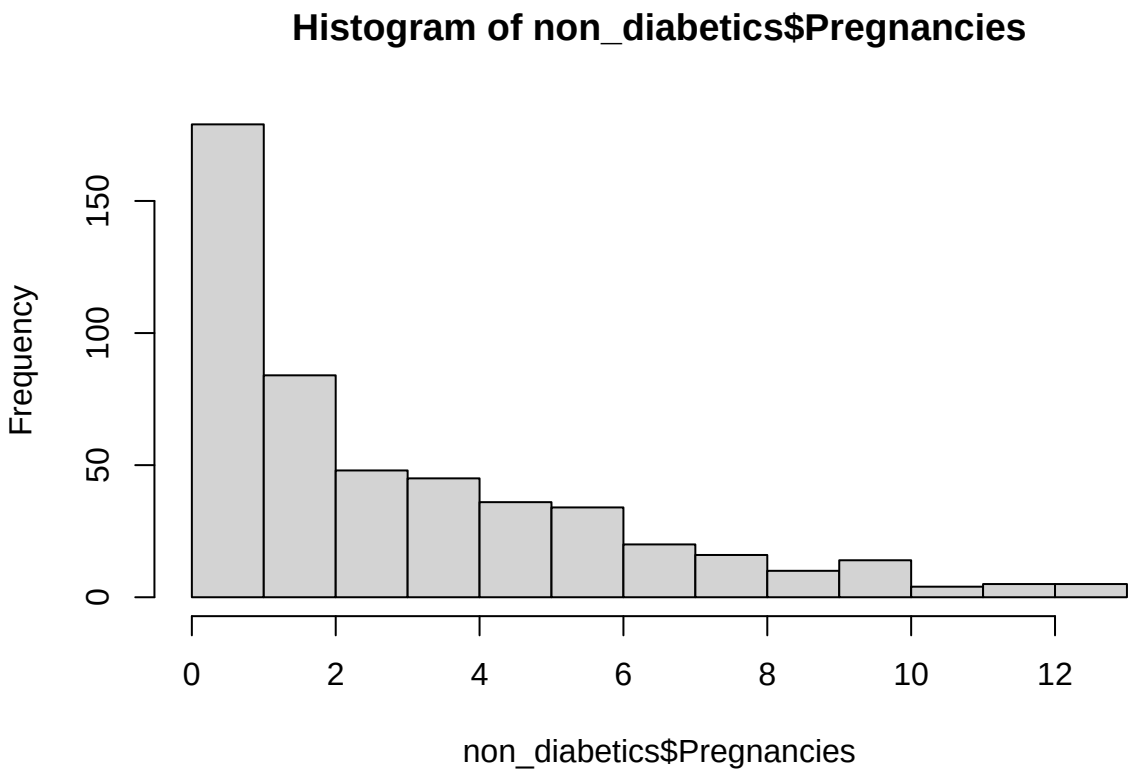


3.- In these plots we can see separately number of pregnancies for diabetics, non diabetics and then we have the most informative plot where we overlap the previous two. We can observe how non diabetic people tend to have less pregnancies.

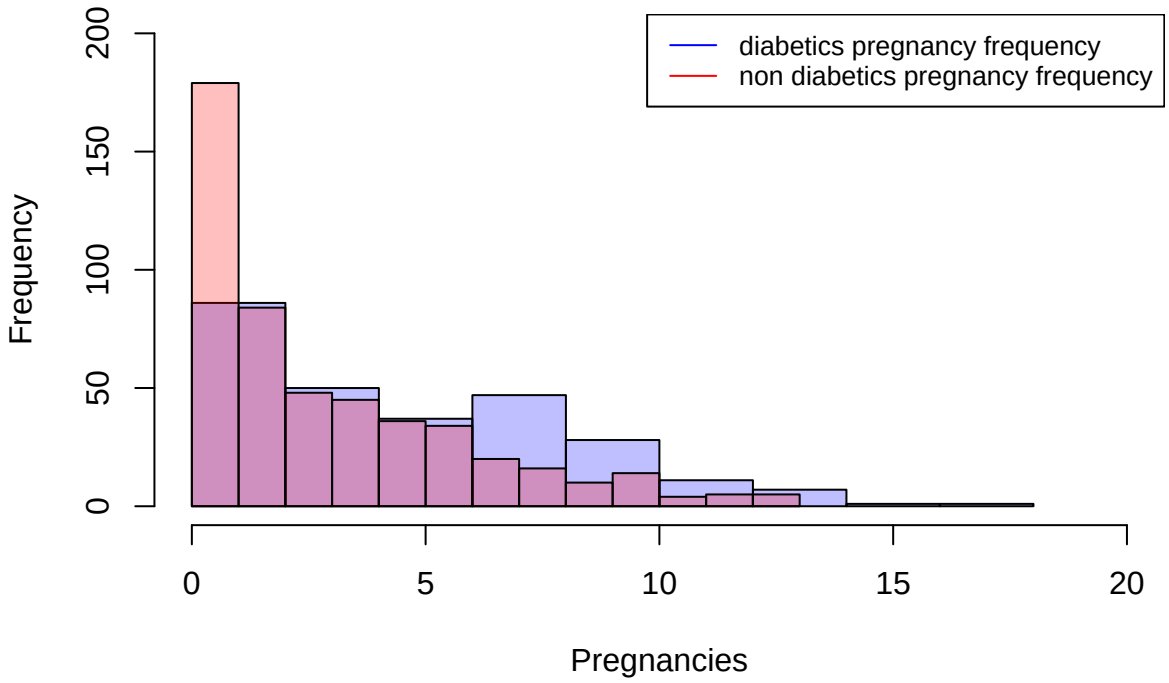
```
p1 <- hist(diabetics$Pregnancies)
```




```
p2 <- hist(non_diabetics$Pregnancies)
```




```
plot( p1, col=rgb(0,0,1,1/4), xlim=c(0,20),
      ylim=c(0,200), main = NULL, xlab = "Pregnancies")  # first histogram
plot( p2, col=rgb(1,0,0,1/4), xlim=c(0,10), add=T)  # second
legend("topright", c("diabetics pregnancy frequency",
                     "non diabetics pregnancy frequency"),
       col = c("blue", "red"), lty =1, cex= 0.8)
```



4 Applying machine learning algorithms

Now we are going to apply several machine learning algorithms. First of all we are going to start by separating the independent variables (X), from the dependent variable (the outcome we are trying to predict, Y):

```
X <- data[, 1:8]
Y <- data$Outcome
```

Now we are going to see how many diabetes cases are in the outcome column, we want to know it because when we separate it into training and test set we want the proportions to be similar, so the dataset is well-balanced. 35% of the cases are diabetics.

```
mean(Y == 1)
```

```
## [1] 0.3489583
```

To improve our predictions we are going to normalize our independent variables: that is, transform them into gaussian variables centered at zero with a standard deviation of 1:

```
X_centered <- sweep(X, 2, colMeans(X), FUN ="-") #subtracting the mean in every column
X_scaled <- sweep(X_centered, 2, colSds(as.matrix(X)), FUN = "/") #dividing by the standard deviation
data_scaled <- cbind(X_scaled, Y)
```

Now we set the seed to obtain the same random information every time we run the code:

```
set.seed(1, sample.kind = "Rounding")
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

Now we are going to separate the data into test and training sets. We will use the training sets to "train" the algorithms with labeled data and then we will examine their real precision using unlabeled data in the test set. We are going to train with the 75% of the data.

```
test_index <- createDataPartition(Y, times = 1, p=0.25, list = FALSE)
test_set <- data_scaled[test_index,]
train_set <- data_scaled[-test_index, ]
```

In the chunk below, we are checking that the proportion of diabetes cases is similar in both training and test sets.

```
mean(train_set$Y == 1)
```

```
## [1] 0.3506944
```

```
mean(test_set$Y == 1)
```

```
## [1] 0.34375
```

Our first model is the simplest one: guessing randomly 0 or 1. Even though we know that its performance will be low it is useful to have a baseline model that indicates us how much improvement are offered by other algorithms. With our guess model we have a 55% of correct predictions. This 55% is higher than the 50% we could suspect we would have had. This is because the outcome column is not evenly distributed and we have more zeros than ones.

```
guess_model <- sample(c(0,1), length(test_index), replace = TRUE)
mean(guess_model == test_set$Y)
```

```
## [1] 0.5572917
```

Now we are going to apply a general linear model, it is simple but it may provide a good performance:

```
fit_linear <- train(as.factor(Y)~., method = "glm", data = train_set)
linear_preds <- predict(fit_linear, test_set)

mean(linear_preds == test_set$Y)
```

```
## [1] 0.8072917
```

In fact, linear model has a 80% accuracy, not bad.

Before applying more models, we will realize PCA. With this technique we can discover which are the variables that influence the most the variance. That is, the variables that most explain the Outcome column and we could focus our efforts in theses columns:

```r
pca_matrix <- prcomp(train_set[,-Y])
summary(pca_matrix)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4     PC5     PC6     PC7
## Standard deviation     1.4779 1.1390 1.0189 0.9150 0.82731 0.76279 0.64999
## Proportion of Variance 0.3034 0.1802 0.1442 0.1163 0.09506 0.08081 0.05868
## Cumulative Proportion  0.3034 0.4835 0.6277 0.7440 0.83905 0.91987 0.97854
##                           PC8
## Standard deviation     0.39304
## Proportion of Variance 0.02146
## Cumulative Proportion  1.00000
```

After applying PCA we can see how PC1 (Pregnancies) explain 30% of the variance. We will perform now a LDA model using Pregnancies as the only predictor:

```r
fit_lda_preg <- train(as.factor(Y)~Pregnancies, method = "lda", data = train_set)
lda_preds_preg <- predict(fit_lda_preg, test_set)

mean(lda_preds_preg == test_set$Y)
```

```
## [1] 0.65625
```

LDA model using Pregnancies as predictor has a 66% accuracy. We will try this same model but using all the variables as predictors:

```r
fit_lda <- train(as.factor(Y)~., method = "lda", data = train_set)
lda_preds <- predict(fit_lda, test_set)

mean(lda_preds == test_set$Y)
```

```
## [1] 0.8125
```

Using all the variables as predictors we have a nicer result: 81% accuracy. However, we can assume that applying PCA before any model can be useful in some situations: For example, if we had a big dataset that provoques computational problems, instead of using all the variables we could apply PCA to discover the 3 or 4 most important variables and apply the models on them, speeding up the process.

Now we will apply QDA model:

```r
fit_qda <- train(as.factor(Y)~., method = "qda", data = train_set)
qda_preds <- predict(fit_qda, test_set)

mean(qda_preds == test_set$Y)
```

```
## [1] 0.7552083
```

For the diabetes dataset, QDA is less precise than LDA.

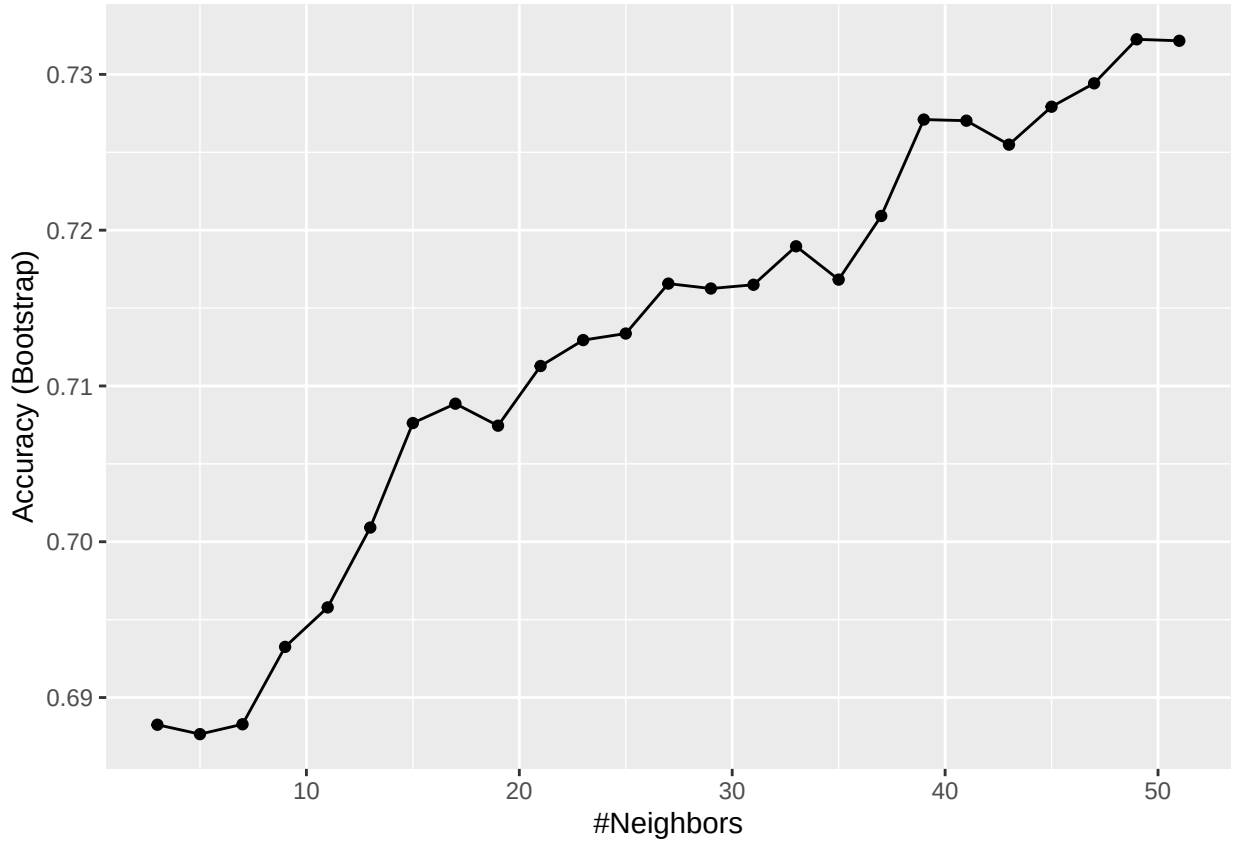
The next model we will try is k-Nearest Neighbours, firstly we will tune our algorithm to find the best K.

```
train_knn <- train(as.factor(Y) ~ .,
                   method = "knn",
                   data = train_set,
                   tuneGrid = data.frame(k = seq(3, 51, 2)))
train_knn$bestTune
```

```
##     k
## 24 49
```

A priori, the best k is 49, we can visualize with the next plot:

```
ggplot(train_knn)
```



KNN accuracy is 77% in our test set.

```
knn_pred <- predict(train_knn, test_set)
mean(knn_pred == test_set$Y)
```

```
## [1] 0.7760417
```

Finally, the last model is an ensemble model. For row, we will apply all the previous models and if the majority of them predict 1, our ensemble model will predict one.

```
ensemble <- cbind(glm = linear_preds == 1, lda = lda_preds == 1, qda = qda_preds == 1, knn = knn_pred ==

ensemble_preds <- ifelse(rowMeans(ensemble) > 0.5, 1, 0)
mean(ensemble_preds == test_set$Y)
```

```
## [1] 0.8125
```

```
models <- c("linear model", "LDA", "QDA", "K nearest neighbors","Ensemble")
accuracy <- c(mean(linear_preds == test_set$Y),
              mean(lda_preds == test_set$Y),
              mean(qda_preds == test_set$Y),
              mean(knn_pred == test_set$Y),
              mean(ensemble_preds == test_set$Y))

data.frame(Model = models, Accuracy = accuracy)
```

```
##                  Model  Accuracy
## 1        linear model 0.8072917
## 2                 LDA 0.8125000
## 3                 QDA 0.7552083
## 4 K nearest neighbors 0.7760417
## 5            Ensemble 0.8125000
```

Ensemle model's accuracy is 82%.

5 Conclusion

We have performed several key steps in any machine learning proyect: We have understood the dataset, focusing in the differences between diabetics and non diabetics. Then we have explored graphically three important trends such as glucose levels, insulin levels and number of pregnancies. Finally we have applied 5 machine learning algorithms to improve our baseline model and obtain good prediction results.