Reg no :j20200776

University ID : w1831982

# Pima Indians Diabetes Database

Summary of the dataset:
The data set used for the purpose of this study is Pima Indians Diabetes Database of National Institute of Diabetes and Digestive and Kidney Diseases. This diabetes database, donated by Vincent Sigillito, is a collection of medical diagnostic reports of 768 examples from a population living near Phoenix, Arizona, USA. You can find more information about the dataset https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes or https://www.kaggle.com/uciml/pima-indians-diabetes-database.

The samples consist of examples with 8 attribute values and one of the two possible outcomes, namely whether the patient is tested positive for diabetes (indicated by output one) or not (indicated by zero).

## Exploratory Data Analysis

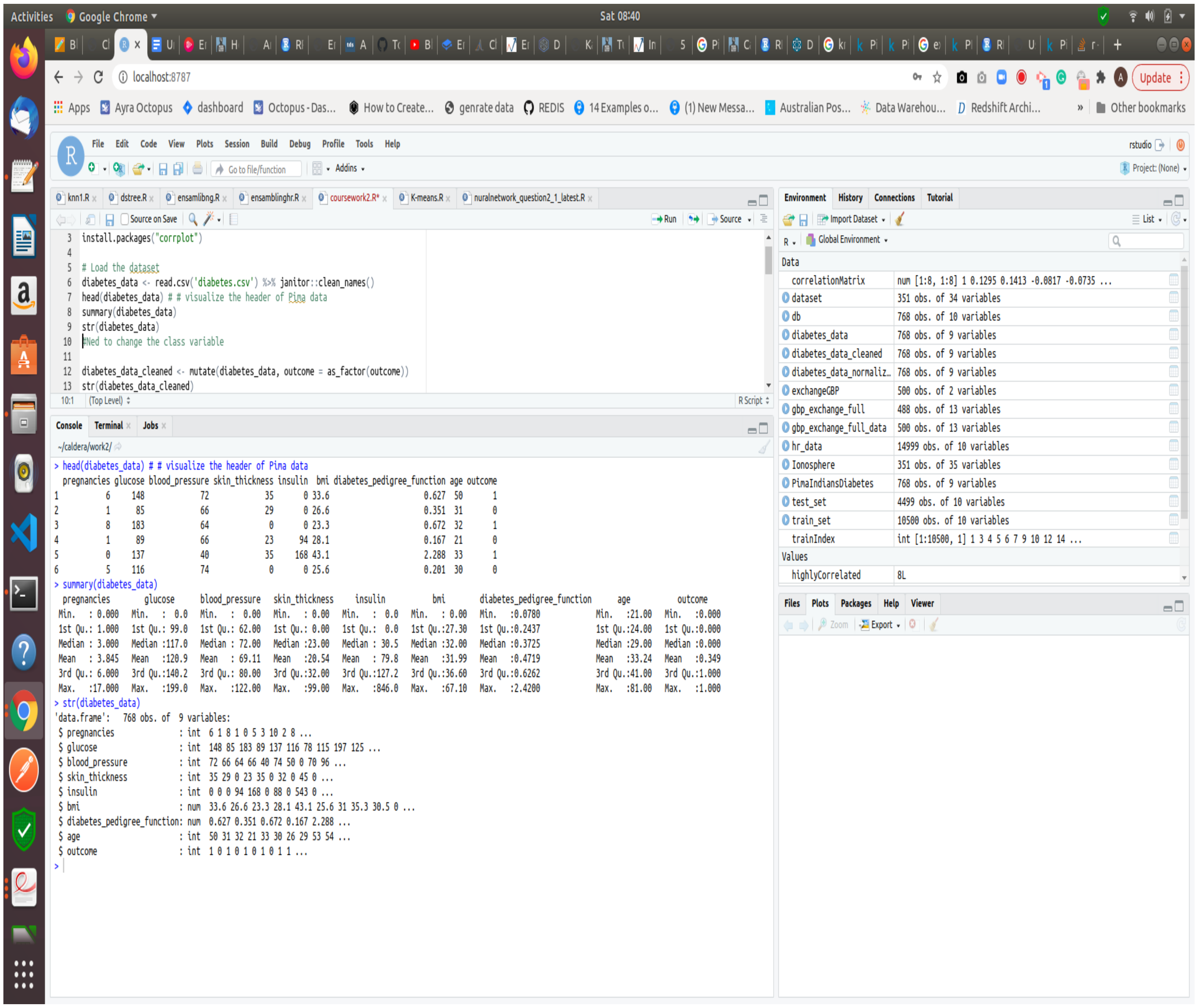
### 1. Data loading and cleaning

- Reading data from csv and observing the variables and data

```
# Load the dataset
diabetes_data <- read.csv('diabetes.csv') %>% janitor::clean_names()
```

```
head(diabetes_data) # # visualize the header of Pima data
summary(diabetes_data)
str(diabetes_data)
```

Results:



```
install.packages("corrplot")

# Load the dataset
diabetes_data <- read.csv('diabetes.csv') %>% janitor::clean_names()
head(diabetes_data) # # visualize the header of Pima data
summary(diabetes_data)
str(diabetes_data)
#Ned to change the class variable

diabetes_data_cleaned <- mutate(diabetes_data, outcome = as_factor(outcome))
str(diabetes_data_cleaned)
```

```
> head(diabetes_data) # # visualize the header of Pima data
  pregnancies glucose blood_pressure skin_thickness insulin  bmi diabetes_pedigree_function age outcome
1           6     148             72             35       0 33.6                      0.627  50       1
2           1      85             66             29       0 26.6                      0.351  31       0
3           8     183             64              0       0 23.3                      0.672  32       1
4           1      89             66             23      94 28.1                      0.167  21       0
5           0     137             40             35     168 43.1                      2.288  33       1
6           5     116             74              0       0 25.6                      0.201  30       0
> summary(diabetes_data)
  pregnancies        glucose      blood_pressure   skin_thickness     insulin          bmi        diabetes_pedigree_function      age           outcome
 Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00   Min.   :  0.0   Min.   : 0.00   Min.   :0.0780             Min.   :21.00   Min.   :0.000
 1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437             1st Qu.:24.00   1st Qu.:0.000
 Median : 3.000   Median :117.0   Median : 72.00   Median :23.00   Median : 30.5   Median :32.00   Median :0.3725             Median :29.00   Median :0.000
 Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54   Mean   : 79.8   Mean   :31.99   Mean   :0.4719             Mean   :33.24   Mean   :0.349
 3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262             3rd Qu.:41.00   3rd Qu.:1.000
 Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00   Max.   :846.0   Max.   :67.10   Max.   :2.4200             Max.   :81.00   Max.   :1.000
> str(diabetes_data)
'data.frame':	768 obs. of  9 variables:
 $ pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ blood_pressure          : int  72 66 64 66 40 74 50 0 70 96 ...
 $ skin_thickness          : int  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ bmi                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
 $ diabetes_pedigree_function: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
> 
```

| | |
|---|---|
| Data | |
| correlationMatrix | num [1:8, 1:8] 1 0.1295 0.1413 -0.0817 -0.0735 ... |
| dataset | 351 obs. of 34 variables |
| db | 768 obs. of 10 variables |
| diabetes_data | 768 obs. of 9 variables |
| diabetes_data_cleaned | 768 obs. of 9 variables |
| diabetes_data_normaliz… | 768 obs. of 9 variables |
| exchangeGBP | 500 obs. of 2 variables |
| gbp_exchange_full | 488 obs. of 13 variables |
| gbp_exchange_full_data | 500 obs. of 13 variables |
| hr_data | 14999 obs. of 10 variables |
| Ionosphere | 351 obs. of 35 variables |
| PimaIndiansDiabetes | 768 obs. of 9 variables |
| test_set | 4499 obs. of 10 variables |
| train_set | 10500 obs. of 10 variables |
| trainIndex | int [1:10500, 1] 1 3 4 5 6 7 9 10 12 14 ... |
| Values | |
| highlyCorrelated | 8L |

- Here it is notable that it is needed to be normalized as features are in different ranges

```
> str(diabetes_data)
```

```
'data.frame':	768 obs. of  9 variables:
 $ pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose                 : int  148 85 183 89 137 116 78 115 197 125
...
 $ blood_pressure          : int  72 66 64 66 40 74 50 0 70 96 ...
 $ skin_thickness          : int  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ bmi                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3
30.5 0 ...
 $ diabetes_pedigree_function: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

- Here the class  variable outcome is needed to be a factor variable.

```
#Need to change the class variable to be a factor variable
diabetes_data$outcome <- factor(diabetes_data$outcome, labels = c("No", "Yes"))
diabetes_data_cleaned =diabetes_data
str(diabetes_data_cleaned)
summary(diabetes_data_cleaned)
```

```
'data.frame':	768 obs. of  9 variables:
 $ pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
 $ glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
 $ blood_pressure          : int  72 66 64 66 40 74 50 0 70 96 ...
 $ skin_thickness          : int  35 29 0 23 35 0 32 0 45 0 ...
 $ insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
 $ bmi                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5
0 ...
 $ diabetes_pedigree_function: num  0.627 0.351 0.672 0.167 2.288 ...
 $ age                     : int  50 31 32 21 33 30 26 29 53 54 ...
 $ outcome                 : Factor w/ 2 levels "Yes","No": 2 1 2 1 2 1 2 1 2
2 ...
```

```
> summary(diabetes_data_cleaned)
pregnancies        glucose       blood_pressure    skin_thickness
 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
```

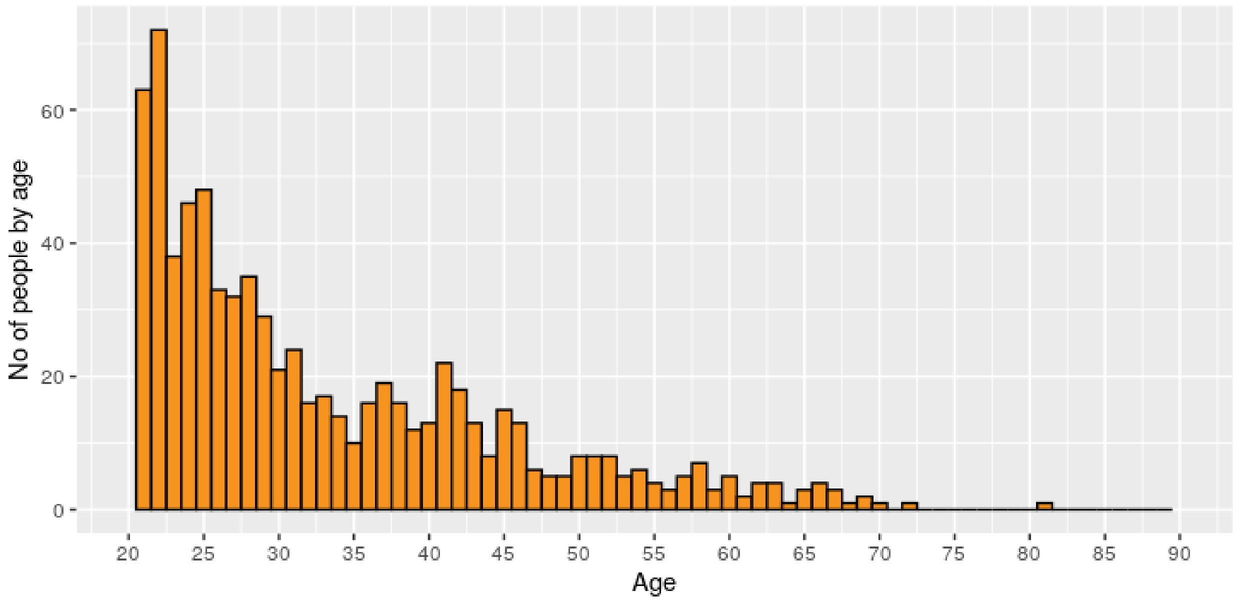
```
skin_thickness     insulin           bmi         diabetes_pedigree_function
 Min.   : 0.00   Min.   :  0.0   Min.   : 0.00   Min.   :0.0780
 1st Qu.: 0.00   1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437
 Median :23.00   Median : 30.5   Median :32.00   Median :0.3725
 Mean   :20.54   Mean   : 79.8   Mean   :31.99   Mean   :0.4719
 3rd Qu.:32.00   3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262
 Max.   :99.00   Max.   :846.0   Max.   :67.10   Max.   :2.4200
```

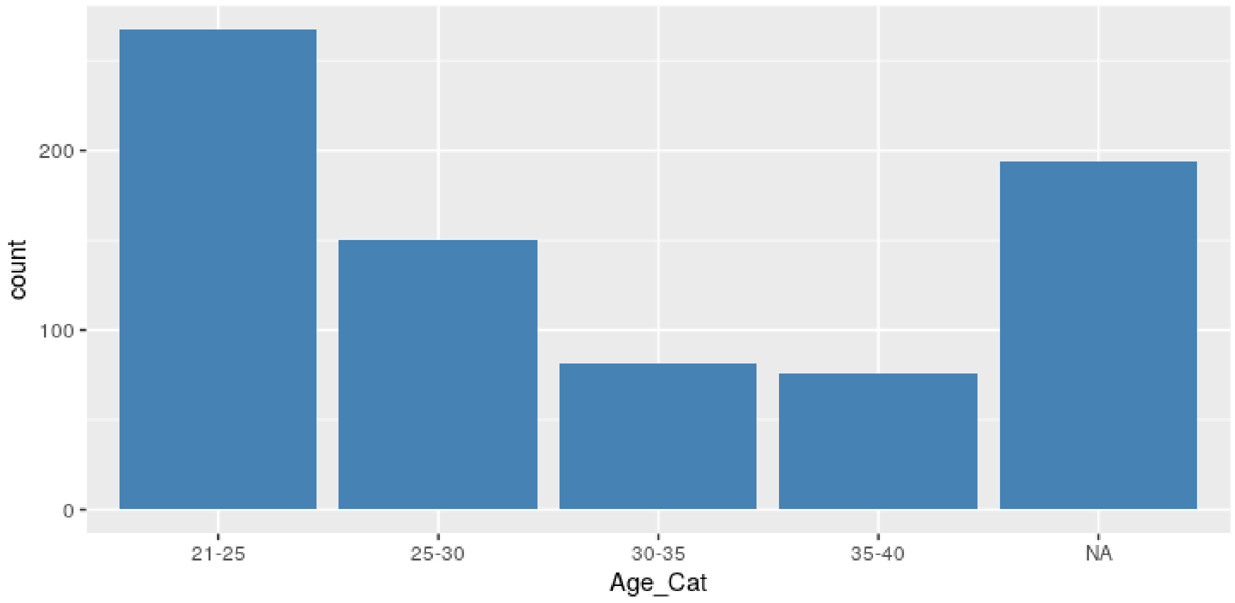
```
age              outcome
 Min.   :21.00   No:500
 1st Qu.:24.00   Yes :268
 Median :29.00
 Mean   :33.24
 3rd Qu.:41.00
 Max.   :81.00
```

- It says that 500 records are mentioned with no diabetes and 268 are having diabetes .

- Agewise analysing against diabetics .

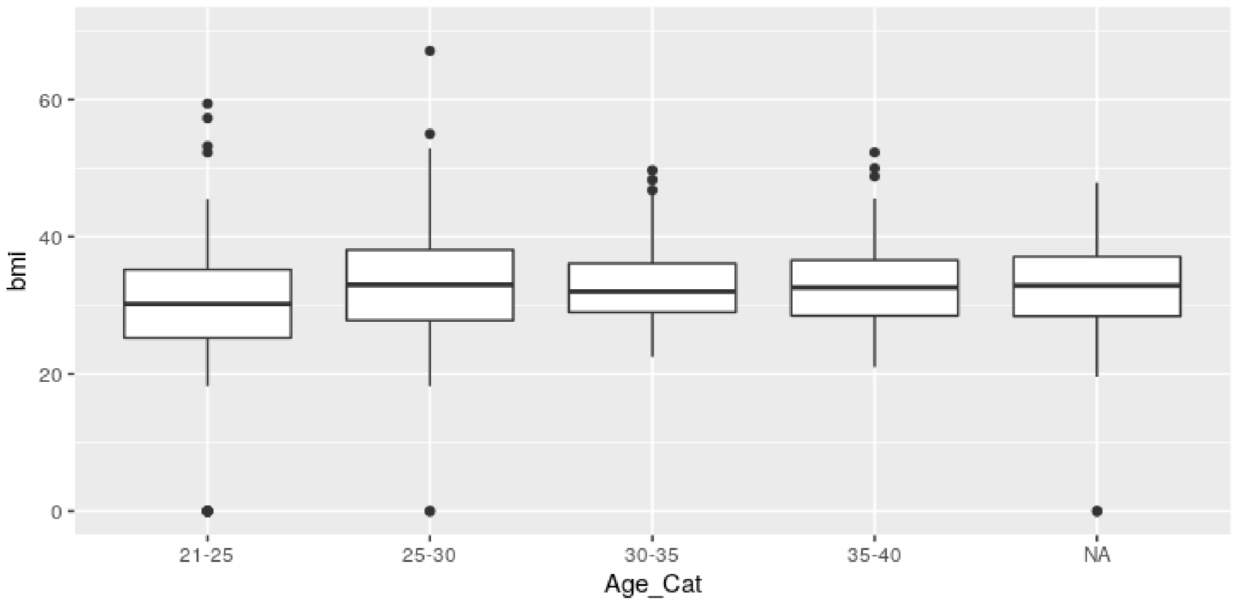
```
##Age waise analysis
db = diabetes_data_cleaned
db$Age_Cat <- ifelse(db$age < 21, "<21",
                     ifelse((db$age>=21) & (db$age<=25), "21-25",
                            ifelse((db$age>25) & (db$age<=30), "25-30",
                                   ifelse((db$age>30) & (db$age<=35),
"30-35",
                                          ifelse((db$age>35) &
(db$age<=40), "35-40",
                                                 ifelse((db$age>40) &
(db$Age<=50), "40-50",
                                                          ifelse((db$age>50)
& (db$age<=60), "50-60",">60")))))))

ggplot(aes(x = age), data=db) +
  geom_histogram(binwidth=1, color='black', fill = "#F79420") +
  scale_x_continuous(limits=c(20,90), breaks=seq(20,90,5)) +
  xlab("Age") +
  ylab("No of people by age")
Please refer the image , images/Agwise_analytcis.png
```





**Most of the subjects are in between the ages 21 - 30**

- 
```
 check outliers Age_Cat vs BMI
library(ggplot2)
ggplot(aes(x=Age_Cat, y = bmi), data = db) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0,70))
```



- Checking the number of missing values in each column.

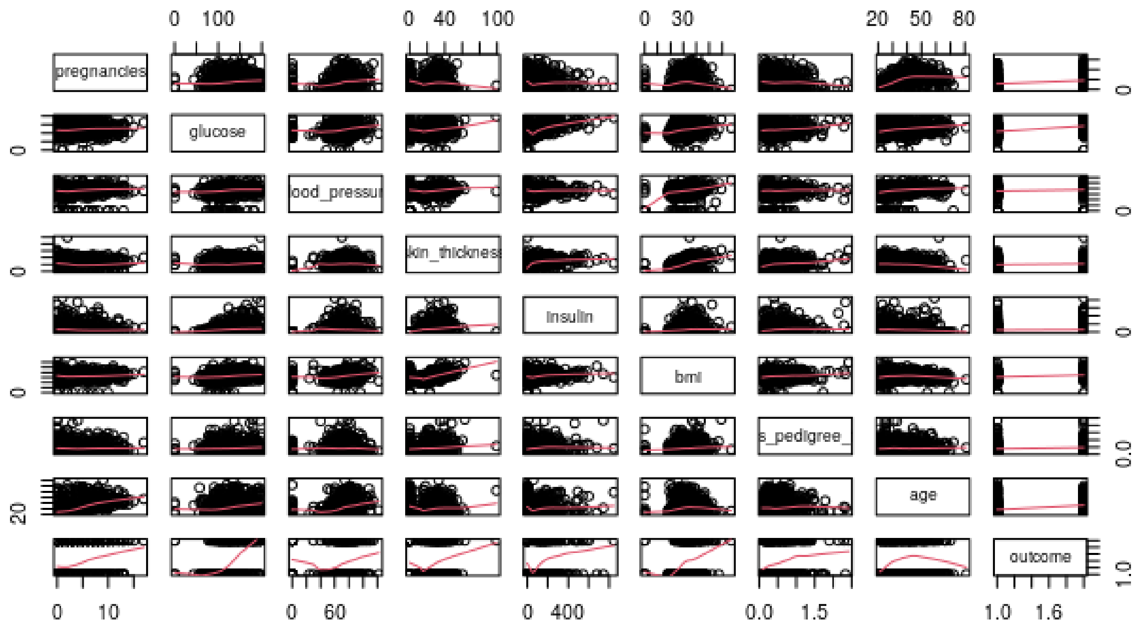
```
sapply(diabetes_data_cleaned, function(x) sum(is.na(x)))




 pregnancies            glucose              blood_pressure
0                         0                  0

 skin_thickness          insulin   bmi      diabetes_pedigree_function
                          0        0           0
0
age                    outcome

0                           0
```

- As the results there is no missing values on the data

- Let's produce the matrix of scatterplots

```
pairs(diabetes_data_cleaned, panel = panel.smooth)
```

```
Please refer the image in the path ,images/matrix_image1.png
```



- Even explicitly no missing values there are several features that have zero values that are not possible.
- Checking how many zero values are available

```
biological_data_investigation <-
diabetes_data_cleaned[,setdiff(names(diabetes_data_cleaned), c('outcome',
'pregnancies'))]
features_miss_num <- apply(biological_data_investigation, 2, function(x)
sum(x<=0))
features_miss <- names(biological_data_investigation)[ features_miss_num >
0]
Features_miss_num
```

- Result of zero values count are given below

```
glucose          blood_pressure          skin_thickness


5                  35                      227
```

```
insulin       bmi      diabetes_pedigree_function

374          11                0

age
```
0

- Investigate how many rows are affected

```
#how many rows are affected
rows_errors <- apply(biological_data_investigation, 1, function(x)
sum(x<=0)>1)
sum(rows_errors)
```

- Results:

```
234
```

```
> sum(rows_errors)/nrow(biological_data_investigation)
```

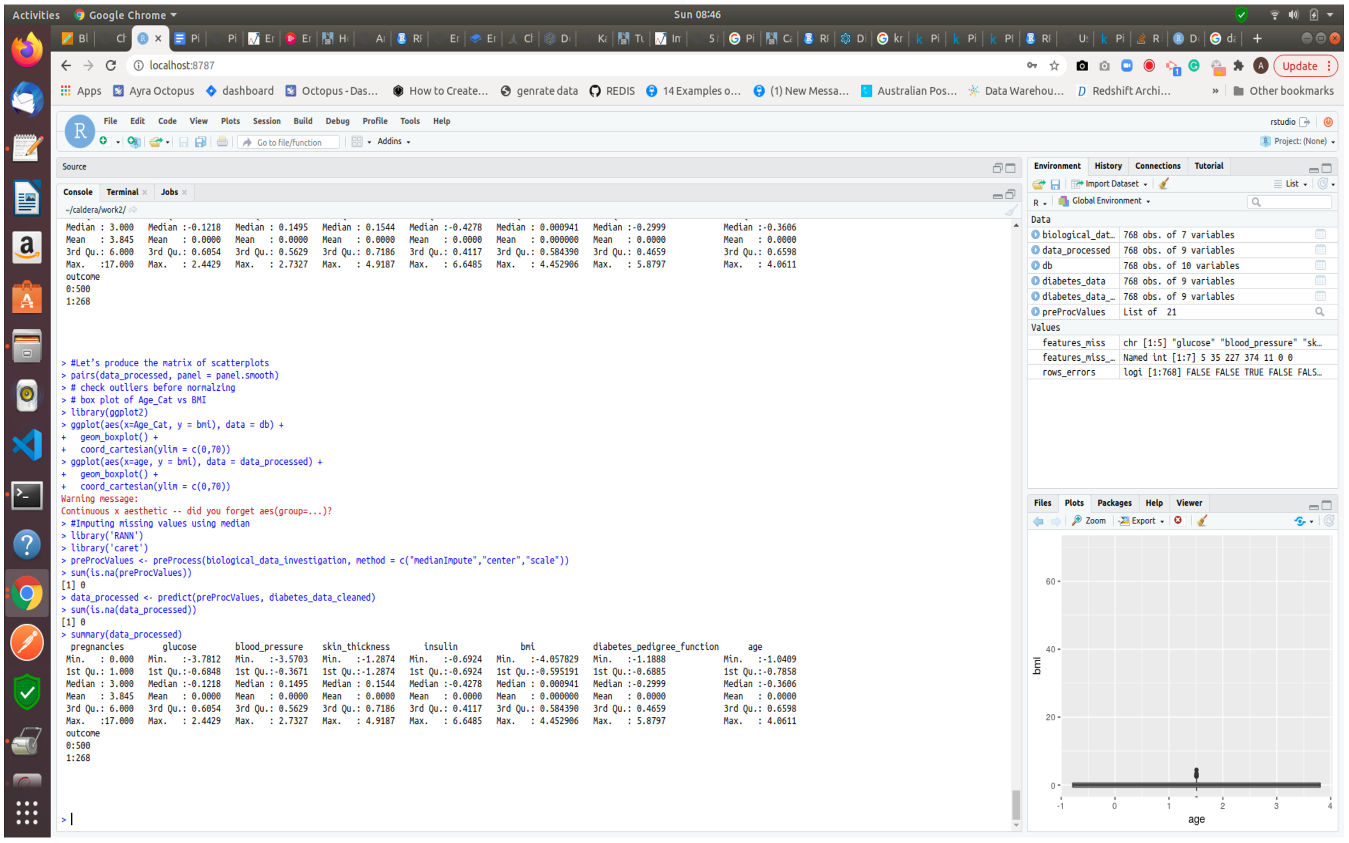
```
[1] 0.3046875

It is more than 30% .
```

- We are going to try to impute missing data

- we can't remove these rows. We are going to try to impute missing data

```
#Imputing missing values using median
preProcValues <- preProcess(biological_data_investigation, method =
c("medianImpute","center","scale"))
sum(is.na(preProcValues))
```

summary(preProcValues)



```
 Median : 3.000   Median :-0.1218   Median : 0.1495   Median : 0.1544   Median :-0.4278   Median : 0.000941   Median :-0.2999              Median :-0.3606
 Mean   : 3.845   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000000   Mean   : 0.0000              Mean   : 0.0000
 3rd Qu.: 6.000   3rd Qu.: 0.6054   3rd Qu.: 0.5629   3rd Qu.: 0.7186   3rd Qu.: 0.4117   3rd Qu.: 0.584390   3rd Qu.: 0.4659              3rd Qu.: 0.6598
 Max.   :17.000   Max.   : 2.4429   Max.   : 2.7327   Max.   : 4.9187   Max.   : 6.6485   Max.   : 4.452906   Max.   : 5.8797              Max.   : 4.0611
 outcome
 0:500
 1:268



> #Let's produce the matrix of scatterplots
> pairs(data_processed, panel = panel.smooth)
> # check outliers before normalzing
> # box plot of Age_Cat vs BMI
> library(ggplot2)
> ggplot(aes(x=Age_Cat, y = bmi), data = db) +
+   geom_boxplot() +
+   coord_cartesian(ylim = c(0,70))
> ggplot(aes(x=age, y = bmi), data = data_processed) +
+   geom_boxplot() +
+   coord_cartesian(ylim = c(0,70))
Warning message:
Continuous x aesthetic -- did you forget aes(group=...)?
> #Imputing missing values using median
> library('RANN')
> library('caret')
> preProcValues <- preProcess(biological_data_investigation, method = c("medianImpute","center","scale"))
> sum(is.na(preProcValues))
[1] 0
> data_processed <- predict(preProcValues, diabetes_data_cleaned)
> sum(is.na(data_processed))
[1] 0
> summary(data_processed)
  pregnancies        glucose         blood_pressure    skin_thickness       insulin              bmi            diabetes_pedigree_function      age
 Min.   : 0.000   Min.   :-3.7812   Min.   :-3.5703   Min.   :-1.2874   Min.   :-0.6924   Min.   :-4.057829   Min.   :-1.1888              Min.   :-1.0409
 1st Qu.: 1.000   1st Qu.:-0.6848   1st Qu.:-0.3671   1st Qu.:-1.2874   1st Qu.:-0.6924   1st Qu.:-0.595191   1st Qu.:-0.6885              1st Qu.:-0.7858
 Median : 3.000   Median :-0.1218   Median : 0.1495   Median : 0.1544   Median :-0.4278   Median : 0.000941   Median :-0.2999              Median :-0.3606
 Mean   : 3.845   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000000   Mean   : 0.0000              Mean   : 0.0000
 3rd Qu.: 6.000   3rd Qu.: 0.6054   3rd Qu.: 0.5629   3rd Qu.: 0.7186   3rd Qu.: 0.4117   3rd Qu.: 0.584390   3rd Qu.: 0.4659              3rd Qu.: 0.6598
 Max.   :17.000   Max.   : 2.4429   Max.   : 2.7327   Max.   : 4.9187   Max.   : 6.6485   Max.   : 4.452906   Max.   : 5.8797              Max.   : 4.0611
 outcome
 0:500
 1:268

>
```

|  |  |  |
|---|---|---|
| Data | | |
| biological_dat… | 768 obs. of 7 variables | |
| data_processed | 768 obs. of 9 variables | |
| db | 768 obs. of 10 variables | |
| diabetes_data | 768 obs. of 9 variables | |
| diabetes_data_… | 768 obs. of 9 variables | |
| preProcValues | List of 21 | |
| Values | | |
| features_miss | chr [1:5] "glucose" "blood_pressure" "sk… | |
| features_miss_… | Named int [1:7] 5 35 227 374 11 0 0 | |
| rows_errors | logi [1:768] FALSE FALSE TRUE FALSE FALS… | |

```
pregnancies           glucose          blood_pressure
 Min.   : 0.000    Min.   :-3.7812   Min.   :-3.5703
 1st Qu.: 1.000    1st Qu.:-0.6848   1st Qu.:-0.3671
 Median : 3.000    Median :-0.1218   Median : 0.1495
 Mean   : 3.845    Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.: 6.000    3rd Qu.: 0.6054   3rd Qu.: 0.5629
 Max.   :17.000    Max.   : 2.4429   Max.   : 2.7327


skin_thickness         insulin              bmi
 Min.   :-1.2874    Min.   :-0.6924    Min.   :-4.057829
 1st Qu.:-1.2874    1st Qu.:-0.6924    1st Qu.:-0.595191
 Median : 0.1544    Median :-0.4278    Median : 0.000941
 Mean   : 0.0000    Mean   : 0.0000    Mean   : 0.000000
 3rd Qu.: 0.7186    3rd Qu.: 0.4117    3rd Qu.: 0.584390
 Max.   : 4.9187    Max.   : 6.6485    Max.   : 4.452906
```

```
diabetes_pedigree_function       age             outcome
 Min.   :-1.1888            Min.   :-1.0409   No :500
 1st Qu.:-0.6885            1st Qu.:-0.7858   Yes:268
 Median :-0.2999            Median :-0.3606
 Mean   : 0.0000            Mean   : 0.0000
 3rd Qu.: 0.4659            3rd Qu.: 0.6598
 Max.   : 5.8797            Max.   : 4.0611
```

## 2.Variable analysis

- Let’s see the proportion of the outcome output .

```
prop.table(table(data_processed$outcome))
```

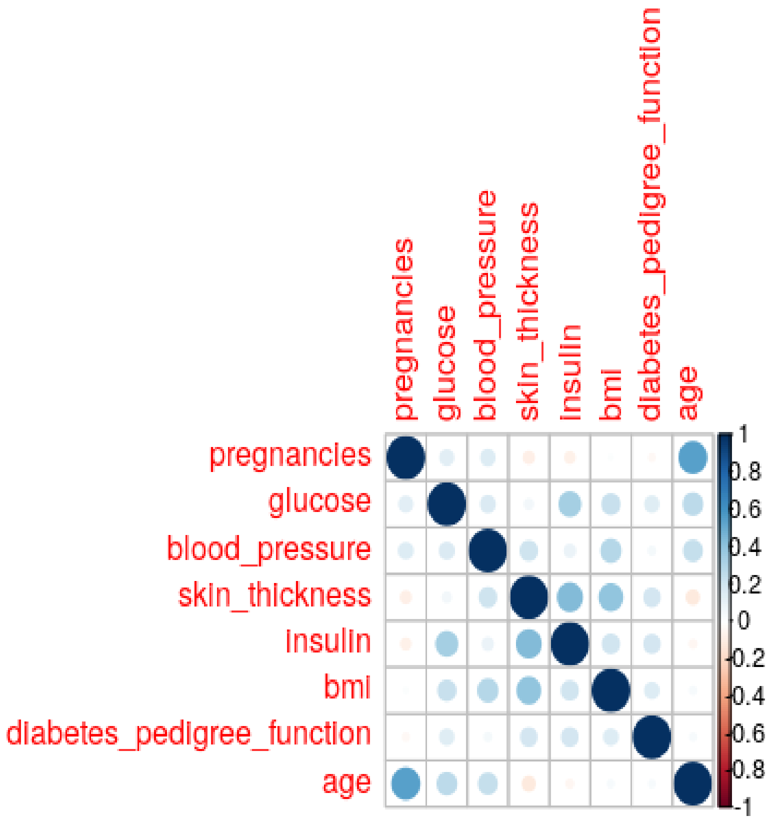
```
No        Yes
0.6510417 0.3489583
```

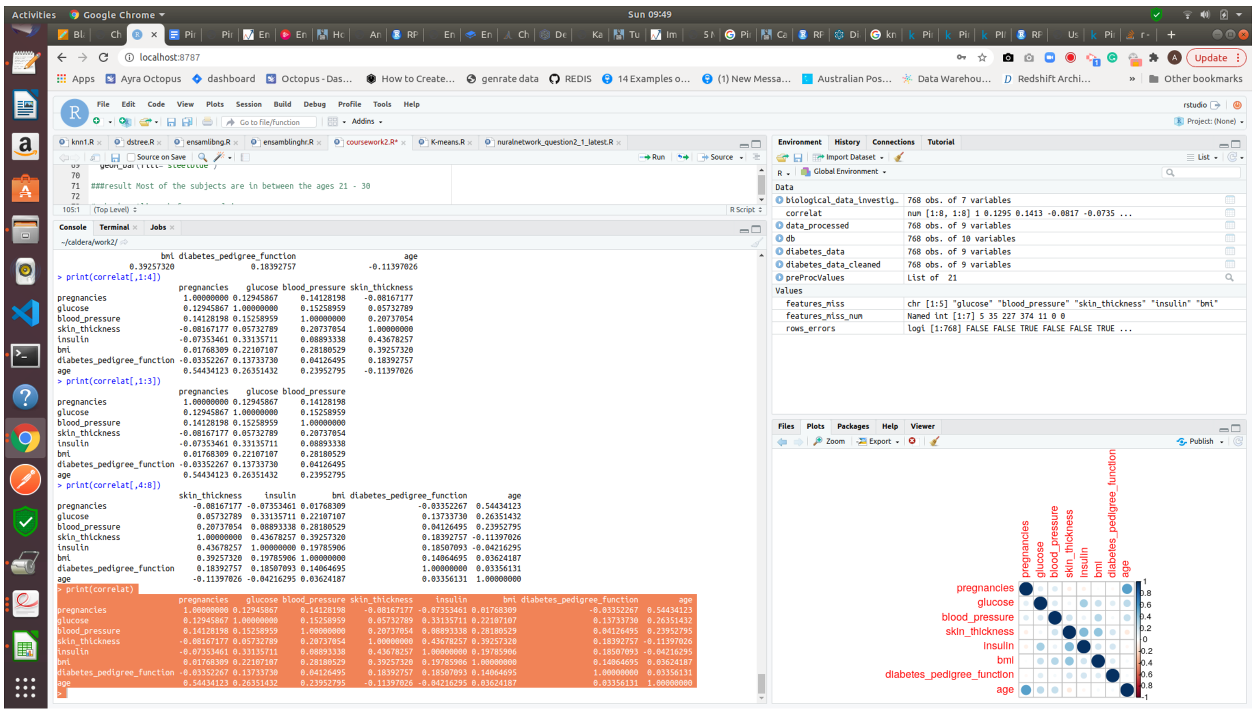
- Results:

It is quite unbalanced with twice the cases of non diabetes.

- Correlation between variable

```
##corerelatinos
# calculate correlation matrix
install.packages("corrplot")
library(corrplot)
correlat <- cor(data_processed[, setdiff(names(data_processed), 'outcome')])
corrplot(correlat)
```





```
###result Most of the subjects are in between the ages 21 - 30
```

```
                        bmi diabetes_pedigree_function                    age
                 0.39257320                 0.18392757            -0.11397026
> print(correlat[,1:4])
                           pregnancies    glucose blood_pressure skin_thickness
pregnancies                 1.00000000 0.12945867     0.14128198    -0.08167177
glucose                     0.12945867 1.00000000     0.15258959     0.05732789
blood_pressure              0.14128198 0.15258959     1.00000000     0.20737054
skin_thickness             -0.08167177 0.05732789     0.20737054     1.00000000
insulin                    -0.07353461 0.33135711     0.08893338     0.43678257
bmi                         0.01768309 0.22107107     0.28180529     0.39257320
diabetes_pedigree_function -0.03352267 0.13733730     0.04126495     0.18392757
age                         0.54434123 0.26351432     0.23952795    -0.11397026
> print(correlat[,1:3])
                           pregnancies    glucose blood_pressure
pregnancies                 1.00000000 0.12945867     0.14128198
glucose                     0.12945867 1.00000000     0.15258959
blood_pressure              0.14128198 0.15258959     1.00000000
skin_thickness             -0.08167177 0.05732789     0.20737054
insulin                    -0.07353461 0.33135711     0.08893338
bmi                         0.01768309 0.22107107     0.28180529
diabetes_pedigree_function -0.03352267 0.13733730     0.04126495
age                         0.54434123 0.26351432     0.23952795
> print(correlat[,4:8])
                           skin_thickness     insulin        bmi diabetes_pedigree_function         age
pregnancies                   -0.08167177 -0.07353461 0.01768309                -0.03352267  0.54434123
glucose                        0.05732789  0.33135711 0.22107107                 0.13733730  0.26351432
blood_pressure                 0.20737054  0.08893338 0.28180529                 0.04126495  0.23952795
skin_thickness                 1.00000000  0.43678257 0.39257320                 0.18392757 -0.11397026
insulin                        0.43678257  1.00000000 0.19785906                 0.18507093 -0.04216295
bmi                            0.39257320  0.19785906 1.00000000                 0.14064695  0.03624187
diabetes_pedigree_function     0.18392757  0.18507093 0.14064695                 1.00000000  0.03356131
age                           -0.11397026 -0.04216295 0.03624187                 0.03356131  1.00000000
> print(correlat)
                           pregnancies    glucose blood_pressure skin_thickness     insulin        bmi diabetes_pedigree_function         age
pregnancies                 1.00000000 0.12945867     0.14128198    -0.08167177 -0.07353461 0.01768309                -0.03352267  0.54434123
glucose                     0.12945867 1.00000000     0.15258959     0.05732789  0.33135711 0.22107107                 0.13733730  0.26351432
blood_pressure              0.14128198 0.15258959     1.00000000     0.20737054  0.08893338 0.28180529                 0.04126495  0.23952795
skin_thickness             -0.08167177 0.05732789     0.20737054     1.00000000  0.43678257 0.39257320                 0.18392757 -0.11397026
insulin                    -0.07353461 0.33135711     0.08893338     0.43678257  1.00000000 0.19785906                 0.18507093 -0.04216295
bmi                         0.01768309 0.22107107     0.28180529     0.39257320  0.19785906 1.00000000                 0.14064695  0.03624187
diabetes_pedigree_function -0.03352267 0.13733730     0.04126495     0.18392757  0.18507093 0.14064695                 1.00000000  0.03356131
age                         0.54434123 0.26351432     0.23952795    -0.11397026 -0.04216295 0.03624187                 0.03356131  1.00000000
>
```

```
Data
biological_data_investig…   768 obs. of 7 variables
correlat                    num [1:8, 1:8] 1 0.1295 0.1413 -0.0817 -0.0735 ...
data_processed              768 obs. of 9 variables
db                          768 obs. of 10 variables
diabetes_data               768 obs. of 9 variables
diabetes_data_cleaned       768 obs. of 9 variables
preProcValues               List of  21
Values
features_miss               chr [1:5] "glucose" "blood_pressure" "skin_thickness" "insulin" "bmi"
features_miss_num           Named int [1:7] 5 35 227 374 11 0 0
rows_errors                 logi [1:768] FALSE FALSE TRUE FALSE FALSE TRUE ...
```

Please refer ,images/matrix_image1.png

```
> print(correlat[,1:3])
```

```
                         pregnancies    glucose blood_pressure
pregnancies                1.00000000 0.12945867     0.14128198
glucose                    0.12945867 1.00000000     0.15258959
blood_pressure             0.14128198 0.15258959     1.00000000
skin_thickness            -0.08167177 0.05732789     0.20737054
insulin                   -0.07353461 0.33135711     0.08893338
bmi                        0.01768309 0.22107107     0.28180529
diabetes_pedigree_function -0.03352267 0.13733730     0.04126495
age                        0.54434123 0.26351432     0.23952795
```

- Output of the analysis:

In this case age is highly related with pregnancies (o.55) but it is not more than 75% so no need to eliminate any features.

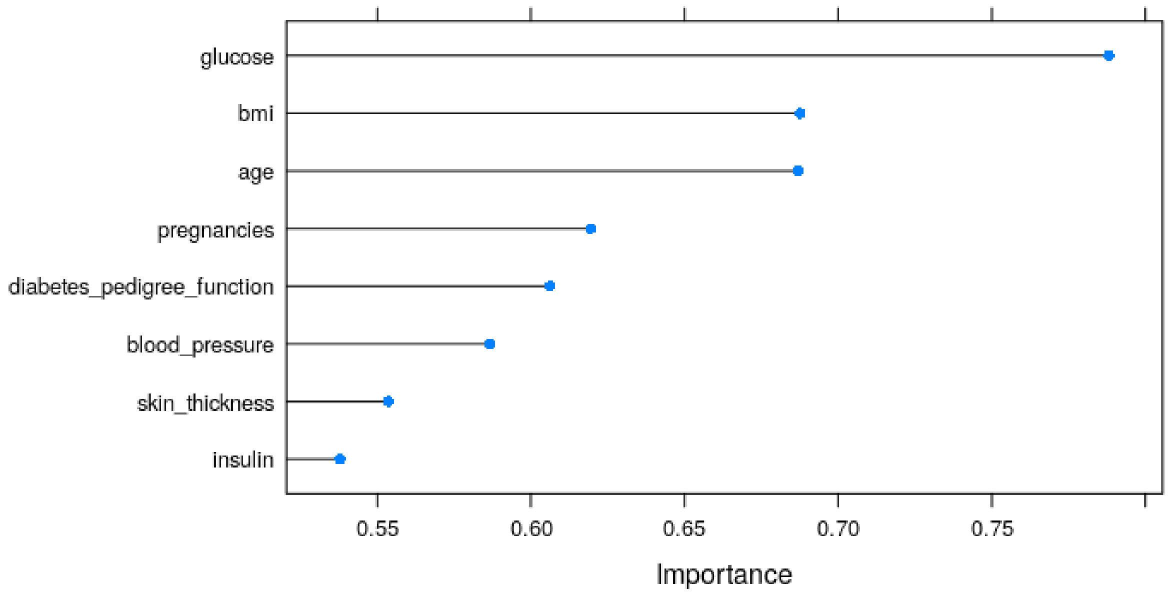
## 3.Feature analysis

```
print(importance)
```

```
ROC curve variable importance

                           Importance
glucose                        0.7881
bmi                            0.6876
age                            0.6869
pregnancies                    0.6195
diabetes_pedigree_function     0.6062
blood_pressure                 0.5865
```

```
skin_thickness                  0.5536
insulin                         0.5379
```

Please refer the image ,images/feature_analysis.png



It is notable that glucose ,bmi and age  contribute high at the sametime insulin contributes low .But here i dont eliminate any features here as my purpose is building ensemble.

## 4.Boosting algorithm

- Stochastic Gradient Boosting
- C5.0

```
# Create train and test data sets
trainIndex = createDataPartition(data_processed$outcome, p=0.7, list=FALSE)
train_set = data_processed[trainIndex,]
test_set = data_processed[-trainIndex,]
seed <- 10
```

Gradient Boosting

```
fit.gbm <- train(outcome~., data=train_set, method="gbm", metric=metric,
          trControl=bagcontrol, verbose=FALSE)
```

- Results

```
confusionMatrix(pred_gbm, test_set$outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  120  18
       Yes  30  62

               Accuracy : 0.7913
                 95% CI : (0.733, 0.8419)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 2.878e-06

                  Kappa : 0.5556
```

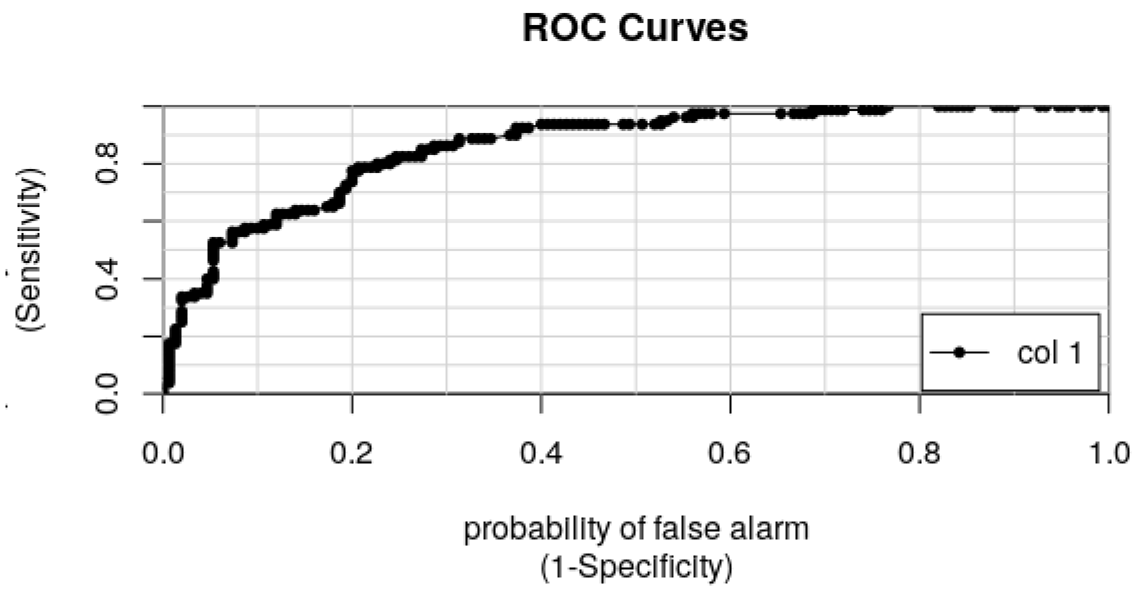
```
Mcnemar's Test P-Value : 0.1124

            Sensitivity : 0.8000
            Specificity : 0.7750
         Pos Pred Value : 0.8696
         Neg Pred Value : 0.6739
             Prevalence : 0.6522
         Detection Rate : 0.5217
   Detection Prevalence : 0.6000
      Balanced Accuracy : 0.7875

       'Positive' Class : No
```

- Analysis of the above results :

Accuracy 79% and Kappa 55%. This is not a bad result.

- ROC

```
No vs. Yes 0.8625417
```



After 0.86 sensitivity it gives more accurate results.

- C5.0

```
set.seed(seed)
fit.c50 <- train(outcome~., data=train_set, method="C5.0", metric=metric,
trControl=bagcontrol)

results_boost <- resamples(list( gbm = fit.gbm, c50 = fit.c50))
# Compare models
dotplot(results_boost)
```

Results:

```
> fit.c50
```

```
C5.0

538 samples
  8 predictor
  2 classes: 'No', 'Yes'

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 484, 484, 485, 484, 484, 485, ...
Addtional sampling using ROSE

Resampling results across tuning parameters:

  model  winnow  trials  ROC        Sens       Spec
  rules  FALSE    1      0.6923113  0.7647619  0.6058480
  rules  FALSE   10      0.7661487  0.7885714  0.6152047
  rules  FALSE   20      0.7661487  0.7885714  0.6152047
  rules   TRUE    1      0.6920663  0.7600000  0.6022417
  rules   TRUE   10      0.7690045  0.7495238  0.6382066
  rules   TRUE   20      0.7690045  0.7495238  0.6382066
  tree   FALSE    1      0.7359844  0.7380952  0.6475634
  tree   FALSE   10      0.7787204  0.7361905  0.6791423
  tree   FALSE   20      0.7787204  0.7361905  0.6791423
  tree    TRUE    1      0.7236647  0.7457143  0.6352827
  tree    TRUE   10      0.7639237  0.7542857  0.6320663
  tree    TRUE   20      0.7639237  0.7542857  0.6320663

ROC was used to select the optimal model using the largest value.
```

```
The final values used for the model were trials = 10, model = tree and winnow =
FALSE.
```

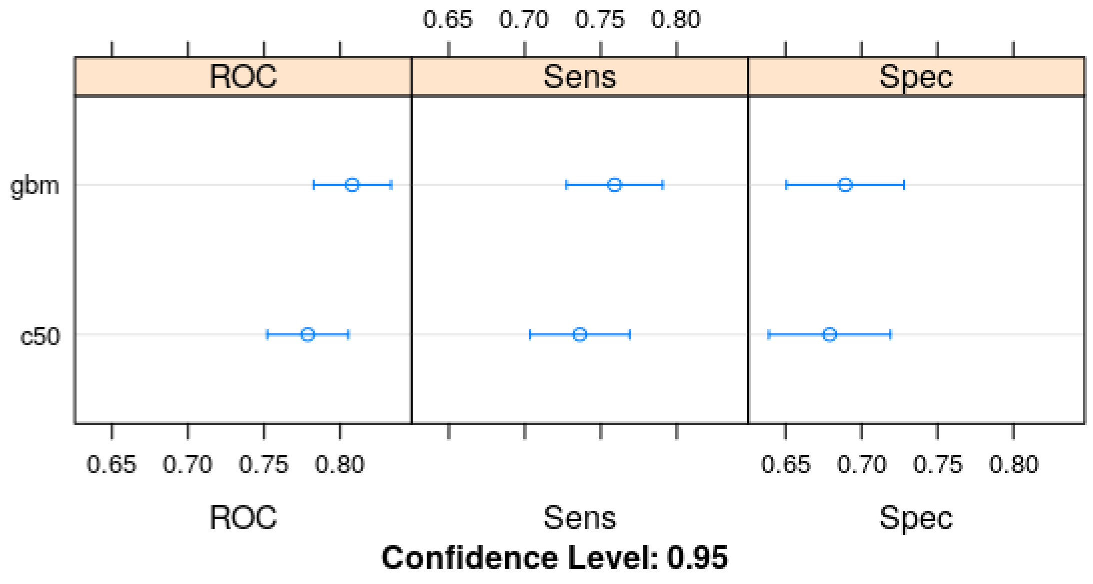
- Analysis of the above results

It represents that gbm works under this condition.

**Comparing both models :**

```
results_boost <- resamples(list( gbm = fit.gbm, c50 = fit.c50))
> # Compare models
> dotplot(results_boost)
> results_boost
```

```
Call:
resamples.default(x = list(gbm = fit.gbm, c50 = fit.c50))

Models: gbm, c50
Number of resamples: 30
Performance metrics: ROC, Sens, Spec
Time estimates for: everything, final model fit
```



- **Using gbm its possible to boost the accuracy**

## 5.Bagging Algorithms

- Random Forest

# Bagging Algorithm (Random Forest)

```
set.seed(seed)
fit.rf <- train(outcome~., data=train_set, method="rf", metric=metric,
trControl=bagcontrol)
# evaluate results on test set
test_set$pred <- predict(fit.rf, newdata=test_set)

#test_set$outcome <- as.factor(test_set$outcome)

str(test_set)
confusionMatrix(data = test_set$pred, test_set$outcome)
pred_fit.rf <- predict(fit.rf, test_set, type="prob")
roc_fit.rf <- roc(test_set$outcome, pred_fit.rf$Yes)
colAUC(pred_fit.rf$Yes, test_set$outcome, plotROC = TRUE)
```

- Result of Random forest bagging

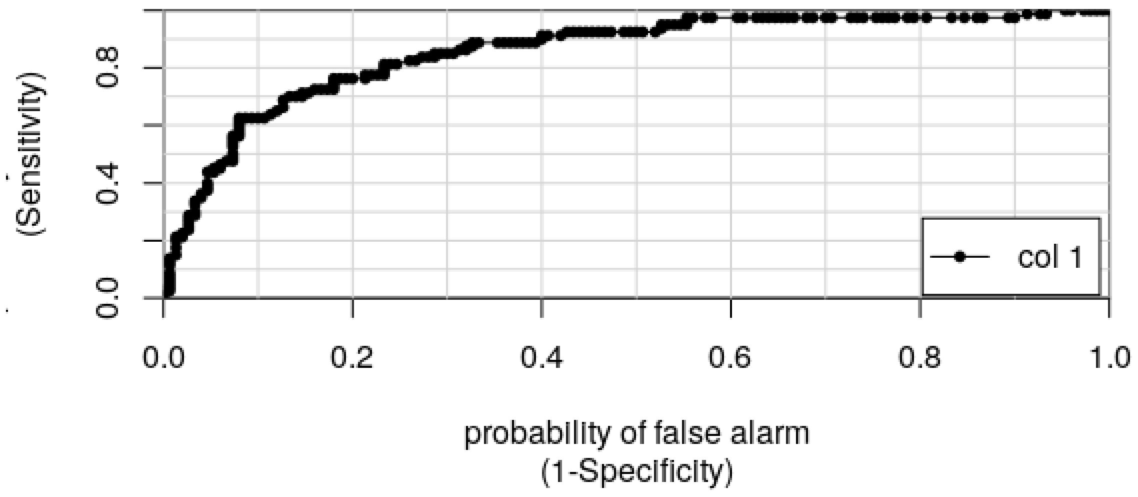
```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  118  18
       Yes  32  62

               Accuracy : 0.7826
                 95% CI : (0.7236, 0.8341)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 1.156e-05

                  Kappa : 0.5396

 Mcnemar's Test P-Value : 0.06599

            Sensitivity : 0.7867
            Specificity : 0.7750
         Pos Pred Value : 0.8676
         Neg Pred Value : 0.6596
```

```
            Prevalence : 0.6522
        Detection Rate : 0.5130
  Detection Prevalence : 0.5913
     Balanced Accuracy : 0.7808

      'Positive' Class : No
```

**Outcome of the analysis:**
**78% is accuracy and Kappa is 53% .Most positive to be No value for the class.**

**No vs. Yes 0.8582917**

ROC Curves



**This says that this model may be biased to predict to be No till improve the model sensitivity (0.85).**

- Bagged CART

```
# Bagged CART
```

```
set.seed(seed)
fit.treebag <- train(outcome~., data=train_set, method="treebag",
metric=metric, trControl=bagcontrol)
pred_fit.treebag <- predict(fit.treebag, newdata=test_set)
confusionMatrix(data = pred_fit.treebag, reference = test_set$outcome)
```

Results:

```
confusionMatrix(data = pred_fit.treebag, reference = test_set$outcome)
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  124  25
       Yes  26  55

               Accuracy : 0.7783
                 95% CI : (0.719, 0.8302)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 2.232e-05

                  Kappa : 0.5127

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8267
            Specificity : 0.6875
         Pos Pred Value : 0.8322
         Neg Pred Value : 0.6790
             Prevalence : 0.6522
         Detection Rate : 0.5391
   Detection Prevalence : 0.6478
      Balanced Accuracy : 0.7571

       'Positive' Class : No
```

**Outcome of the analysis:**
**77% is accuracy and Kappa is 51% .**
When comparing Random forest and `Bagged CART then Random forest gives high accuracy.`

`Finally` ,Random Forest behaves very well for this scenario .

- Evaluation with test data set using Random forest

```
# evaluate results on test set
test_set$pred <- predict(fit.rf, newdata=test_set)
confusionMatrix(data = test_set$pred, reference = test_set$outcome)
```

```
Results:

Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  118  18
       Yes  32  62

               Accuracy : 0.7826
                 95% CI : (0.7236, 0.8341)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 1.156e-05

                  Kappa : 0.5396

 Mcnemar's Test P-Value : 0.06599

            Sensitivity : 0.7867
            Specificity : 0.7750
         Pos Pred Value : 0.8676
         Neg Pred Value : 0.6596
             Prevalence : 0.6522
         Detection Rate : 0.5130
   Detection Prevalence : 0.5913
      Balanced Accuracy : 0.7808

       'Positive' Class : No
```

- **It gives good 78% accuracy results with testing data.**

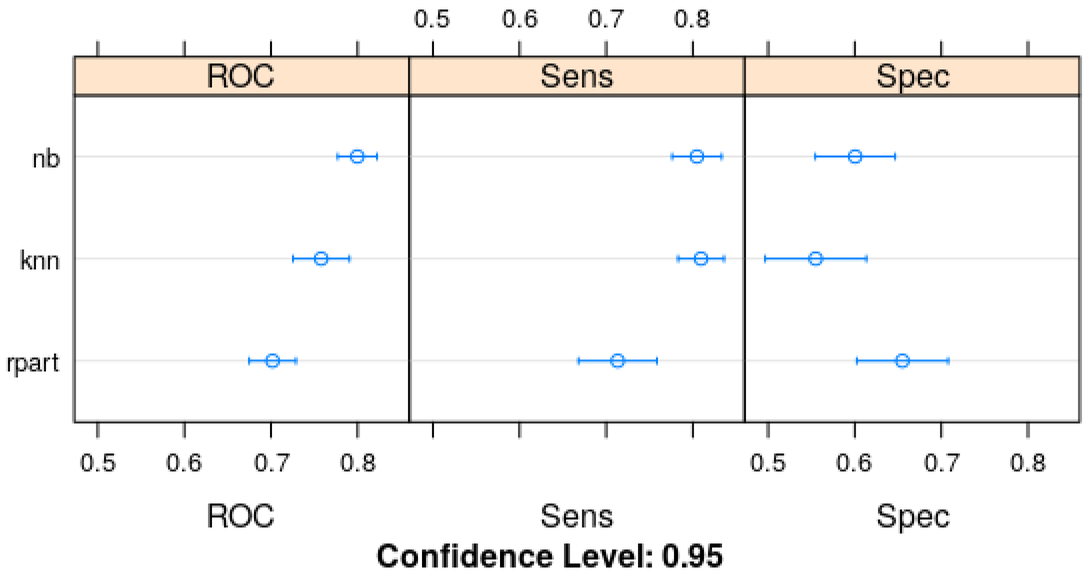
## 6.Stacking Algorithms

- Classification and Regression Trees (CART),
- K-Nearest Neighbors (KNN),
- Naïve Bayes (NB)

```
# Stacking Algorithms
stack_control <- trainControl(sampling="rose",method="repeatedcv", number=10,
repeats=3,
                        savePredictions='final', classProbs=TRUE,
summaryFunction = twoClassSummary)
algorithmList <- c( 'knn','rpart','nb')
set.seed(seed)
str(train_set);
#levels(train_set$outcome) <- make.names(levels(factor(train_set$outcome)))
stack_models <- caretList(outcome~., data=train_set, trControl=stack_control,
                          methodList=algorithmList, metric = "ROC" )
stacking_results <- resamples(stack_models)
summary(stacking_results)
dotplot(stacking_results)
names(stack_models)
lapply(stack_models,"[[","results")
# Check correlation between models to ensure the results are uncorrelated and
can be
modelCor(stacking_results)
splom(stacking_results)
```

- Results:

```
> summary(stacking_results)
Call:
summary.resamples(object = stacking_results)

Models: knn, rpart, nb
Number of resamples: 30

ROC
           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
knn   0.5488722 0.7001984 0.7620301 0.7579588 0.8287594 0.9365079    0
rpart 0.5406015 0.6549499 0.6962406 0.7018101 0.7701128 0.8157895    0
nb    0.6571429 0.7593985 0.7958229 0.7997271 0.8449457 0.9353383    0

Sens
           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
knn   0.6000000 0.7714286 0.8285714 0.8095238 0.8571429 0.9142857    0
rpart 0.4571429 0.6357143 0.7142857 0.7133333 0.7714286 1.0000000    0
nb    0.6000000 0.7500000 0.8285714 0.8047619 0.8571429 0.9428571    0

Spec
           Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
knn   0.3333333 0.4269006 0.5263158 0.5549708 0.6710526 0.9444444    0
rpart 0.3333333 0.5869883 0.6315789 0.6550682 0.7763158 0.8947368    0
nb    0.3333333 0.5336257 0.6052632 0.6005848 0.6842105 0.8421053    0
```



Confidence level 95%.

Please refer images/stacking_confidence.png

- Analysis outcomes：

It says the Naive Bayes algorithm works with high accuracy.

- Check correlations between models

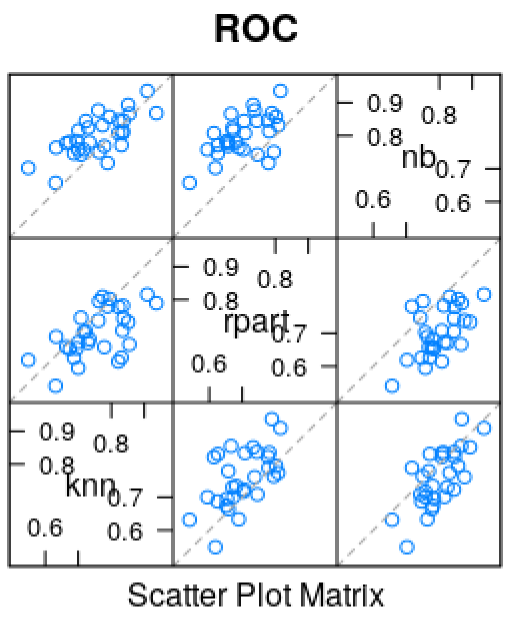
```
modelCor(stacking_results)
```

```
  knn      rpart        nb
knn   1.0000000 0.5827502 0.7126511
rpart 0.5827502 1.0000000 0.5650957
nb    0.7126511 0.5650957 1.0000000
```

Analysis outcome：

No one is more than 80% so no models are core-related.

- `splom(stacking_results)`



- Stack using naive Bayes

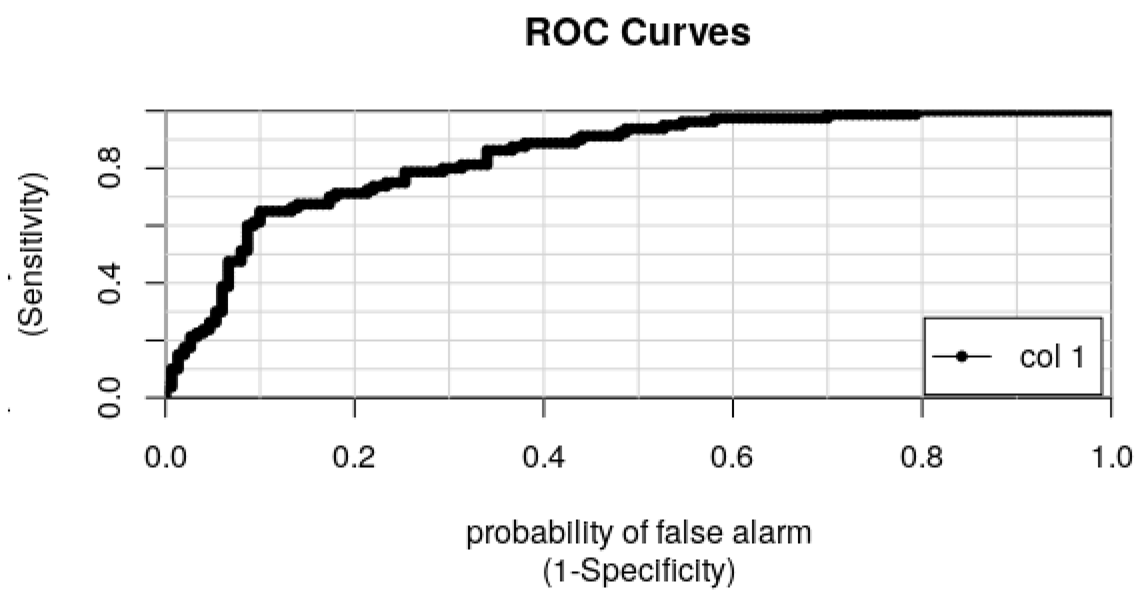
```
# stack using nb
set.seed(seed)
#start time
start_time <- Sys.time()
stack_nb_model <- caretStack(stack_models, method="nb", metric = metric,
                        trControl=stack_control)
end_time <- Sys.time()

timediffrences <- (end_time - start_time)
print(timediffrences)
```

Results:

```
Time difference of 6.942099 secs
```

```
confusionMatrix(data = stack.nb.pred, reference = test_set$outcome)
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  128  26
       Yes  22  54

               Accuracy : 0.7913
                 95% CI : (0.733, 0.8419)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 2.878e-06

                  Kappa : 0.5346

 Mcnemar's Test P-Value : 0.665

            Sensitivity : 0.8533
            Specificity : 0.6750
         Pos Pred Value : 0.8312
         Neg Pred Value : 0.7105
             Prevalence : 0.6522
         Detection Rate : 0.5565
   Detection Prevalence : 0.6696
      Balanced Accuracy : 0.7642

       'Positive' Class : No
```

Here Accuracy has been increased to 79% and kappa is 0.5346.This is a good results



```
No vs. Yes 0.8426667
```

After 0.6 (in X axis ) it gives a more accurate prediction .

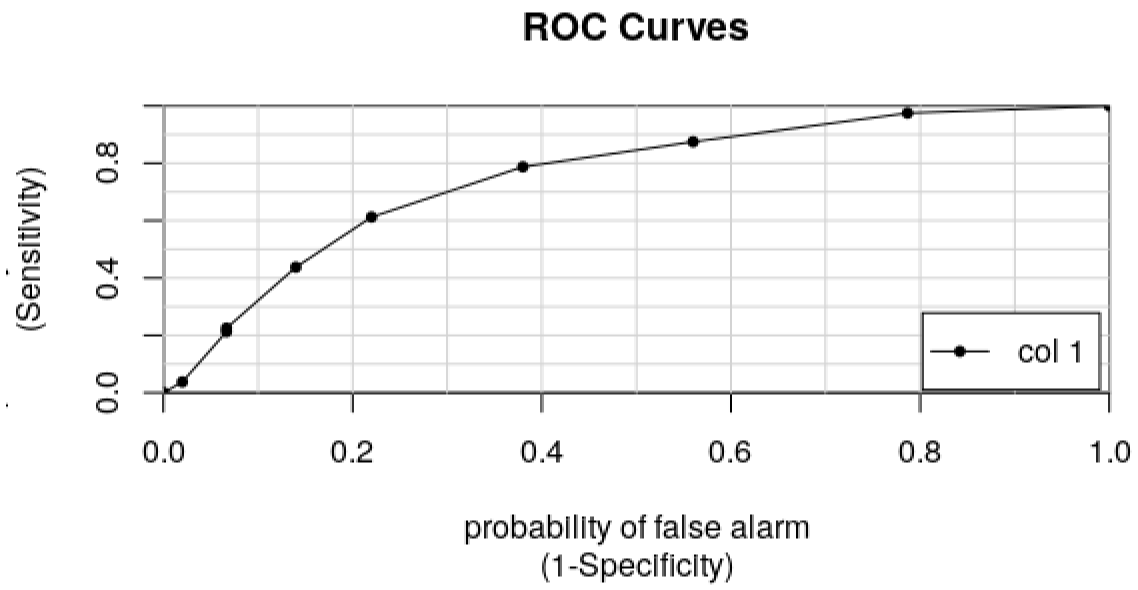
- Stack using KNN

```
# stack using knn
set.seed(seed)
#start time
start_time <- Sys.time()
stack_model.knn <- caretStack(stack_models, method="knn", metric = metric,
                       trControl=stack_control)
end_time <- Sys.time()

timediffrences <- (end_time - start_time)
print(timediffrences)
```

```
Time difference of 2.964403 secs
```

```
> confusionMatrix(data = pred_fit_stack.knn, reference = test_set$outcome)

Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  117  31
       Yes  33  49

               Accuracy : 0.7217
                 95% CI : (0.659, 0.7786)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.0148

                  Kappa : 0.3902

 Mcnemar's Test P-Value : 0.9005

            Sensitivity : 0.7800
            Specificity : 0.6125
         Pos Pred Value : 0.7905
         Neg Pred Value : 0.5976
             Prevalence : 0.6522
         Detection Rate : 0.5087
   Detection Prevalence : 0.6435
      Balanced Accuracy : 0.6963

       'Positive' Class : No
```

```
No vs. Yes 0.7544583
```



After 0.8 (in X axis ) it gives a more accurate prediction .

Here Accuracy has been increased to 72% and kappa is 39%.This is nood when comparing with Naive bayes.

- Stack using CART

```
#start time
start_time <- Sys.time()
stack_rpart_model.rpart <- caretStack(stack_models, method="rpart", metric
= metric,
                        trControl=stack_control)
end_time <- Sys.time()

timediffrences <- (end_time - start_time)
print(timediffrences)
```

```
Time difference of 2.542559 secs
```

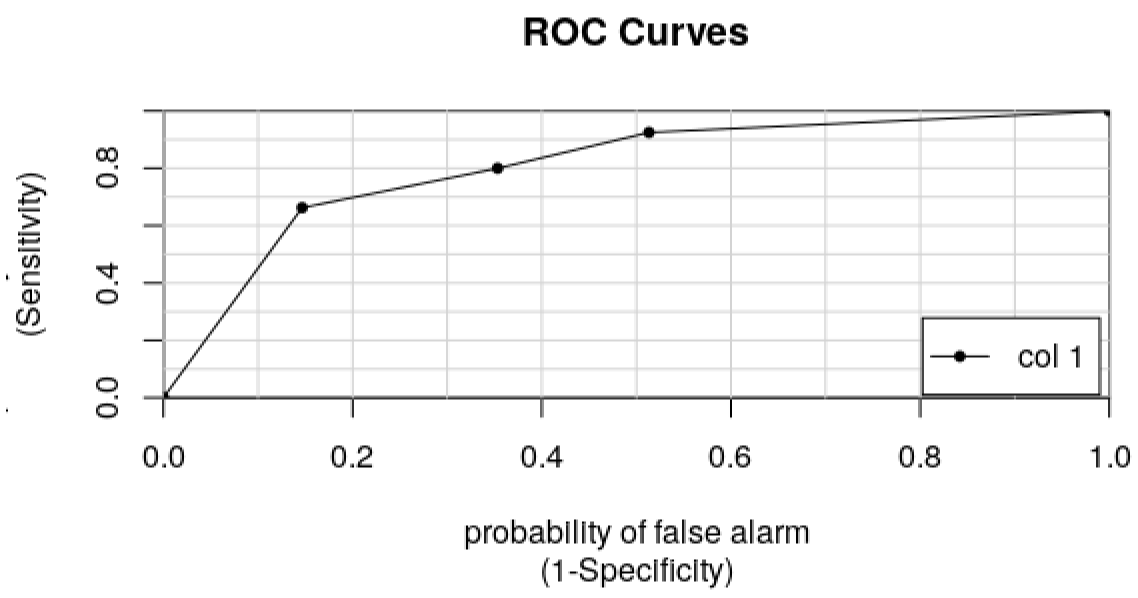
```
ConfusionMatrix(data =pred_fit_stack.rpart, reference = test_set$outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  124  25
       Yes  26  55

               Accuracy : 0.7783
                 95% CI : (0.719, 0.8302)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 2.232e-05

                  Kappa : 0.5127

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8267
            Specificity : 0.6875
         Pos Pred Value : 0.8322
         Neg Pred Value : 0.6790
             Prevalence : 0.6522
         Detection Rate : 0.5391
   Detection Prevalence : 0.6478
      Balanced Accuracy : 0.7571

       'Positive' Class : No
```

```
No vs. Yes 0.806125
```

After 0.85 (in X axis ) it gives a more accurate prediction .

Here Accuracy has been increased to 78% and kappa is 51%.This is nood when comparing with Naive bayes.



Compare all three models

When comparing the three stack models Naive Bayes works well 79% and kappa is 0.5346.
Training tIme comparing :Naive Nayes takes more time ,`6.942099 secs`

Just single Naive bayes results:

```
#naive bayes
set.seed(seed)
fit.nb <- train(outcome~., data=train_set, method="nb", metric=metric,
trControl=bagcontrol)
pred_fit.nb <- predict(fit.nb, newdata=test_set)
confusionMatrix(data = pred_fit.nb, reference = test_set$outcome)
```

```
Confusion Matrix and Statistics

          Reference
Prediction  No Yes
       No  111  23
       Yes  39  57

               Accuracy : 0.7304
                 95% CI : (0.6682, 0.7866)
    No Information Rate : 0.6522
    P-Value [Acc > NIR] : 0.006878

                  Kappa : 0.4323

 Mcnemar's Test P-Value : 0.056780

            Sensitivity : 0.7400
            Specificity : 0.7125
         Pos Pred Value : 0.8284
         Neg Pred Value : 0.5938
             Prevalence : 0.6522
         Detection Rate : 0.4826
   Detection Prevalence : 0.5826
      Balanced Accuracy : 0.7263

       'Positive' Class : No
```

Without stacking Naive Bayes gviesl 73% and kappa is 0.43

So it's more clear ensemble stacking gives more accuracy than non-ensembling method of NB .

How can we increase reliability and consistency?

1.As unwanted features were not removed, so if those are removed possible to get more accurate and decrease the training time.

2.Here imbalanced class data has been used ,if we can get  balanced class data possible to see more accurately.
3.Need to use proper boosting .

Refrences:
https://www.javaer101.com/en/article/15507128.html
https://www.rdocumentation.org/packages/RANN.L1/versions/2.5.2