# Diabetes

**Context:**

This dataset is originally from the National Institute of Diabetes and Digestive and Kidney Diseases. The objective of the dataset is to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

**Problem Statement:**

Build a model to accurately predict whether the patients in the dataset have diabetes or not?

**Dataset Description:**

The datasets consists of several medical predictor variables and one target variable, Outcome. Predictor variables includes the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

**Pregnancies**: Number of times pregnant

**Glucose**: Plasma glucose concentration a 2 hours in an oral glucose tolerance test

**BloodPressure**: Diastolic blood pressure (mm Hg)

**SkinThickness**: Triceps skin fold thickness (mm)

**Insulin**: 2-Hour serum insulin (mu U/ml)

**BMI**: Body mass index (weight in kg/(height in m)^2)

**DiabetesPedigreeFunction**: Diabetes pedigree function

**Age**: Age (years)

**Outcome**: Class variable (0 or 1) 268 of 768 are 1, the others are 0

Perform descriptive analysis. It is very important to understand the variables and corresponding values. We need to think through - Can minimum value of below listed columns be zero (0)? On these columns, a value of zero does not make sense and thus indicates missing value.

library('dplyr')

df=read.csv('health care diabetes.csv')#importing the dataset

nrow(df)#there are 768 rows in total

ncol(df)#there are 9 columns in total

dim(df)

head(df)#there are some values which are 0 in SkinThickness and Insulin so we might have to impute them and we will treat them as null values

str(df)

# Missing Values ----------------------------------------------------------

colSums(is.na(df)) #there are no missing values

df[df == 0] <- NA #replacing 0 wih na values

colSums(is.na(df))

df$Outcome[is.na(df$Outcome)]=0# replacing Na with 0 in Outsome as it the target variable

df$Pregnancies[is.na(df$Pregnancies)]=0

colSums(is.na(df))

#the majority of missing value are in skin thickness and insulin which constitutes for more than 50 percent of the data

df <- df[rowSums(is.na(df)) < 3, ]

colSums(is.na(df))

dim(df)# removing rows with more than 3 Na values

colSums(is.na(df))

# BP,BMI,Glucose ----------------------------------------------------------

library(tidyr)

df1=df %>% drop\_na(BloodPressure,BMI,Glucose)

dim((df1))

# we drop the na values for these columns because they have a very small amount of na values

colSums(is.na(df1))

# Skin Thickness And Insulin ----------------------------------------------

df %>% group\_by(Outcome) %>% summarise(SkinThickness=mean(SkinThickness,na.rm = T))

Outcome SkinThickness

*<dbl>* *<dbl>*

1 0 27.2

2 1 33

#we see that there is hardly any difference between a diabetic and normal person we impute mean

str(df1)

Data.frame': 724 obs. of 10 variables:

$ Pregnancies : num 6 1 8 1 0 5 3 2 4 10 ...

$ Glucose : int 148 85 183 89 137 116 78 197 110 168 ...

$ BloodPressure : int 72 66 64 66 40 74 50 70 92 74 ...

$ SkinThickness : num 35 29 29 23 35 29 32 45 29 29 ...

$ BMI : num 33.6 26.6 23.3 28.1 43.1 25.6 31 30.5 37.6 38 ...

$ DiabetesPedigreeFunction: num 0.627 0.351 0.672 0.167 2.288 ...

$ Age : int 50 31 32 21 33 30 26 53 30 34 ...

$ Outcome : num 1 0 1 0 1 0 1 1 0 1 ...

$ replace\_insulin : num 2.23 2.02 2.23 1.97 2.23 ...

$ AgeBin : int 3 1 1 1 1 1 1 3 1 2 ...

library(Hmisc)

df1 %>% group\_by(Outcome) %>% summarise(SkinThickness=median(SkinThickness,na.rm=T))

median(df1$SkinThickness,na.rm=T)

df1$SkinThickness[is.na(df1$SkinThickness)]=29 #imputing the value with mean

colSums(is.na(df1))

df %>% group\_by(Outcome) %>% summarise(Insulin=median(Insulin,na.rm = T))

# The mean insulin for a healthy person 130

# The mean insulin for a diabetic person 207

#So we will impute the values accordingly

df1=df1%>%

mutate(replace\_insulin= ifelse(is.na(Insulin) & Outcome==0,

df1 %>% filter(Outcome==0) %>% summarise(median(Insulin,na.rm=TRUE)),

ifelse(is.na(Insulin) & Outcome==1,

df1 %>% filter(Outcome==1) %>% summarise(median(Insulin,na.rm=TRUE)),

Insulin))) #imputing the values according to the output

head(df1)

df1$Insulin=NULL

head(df1)

# EDA ---------------------------------------------------------------------

str(df1)

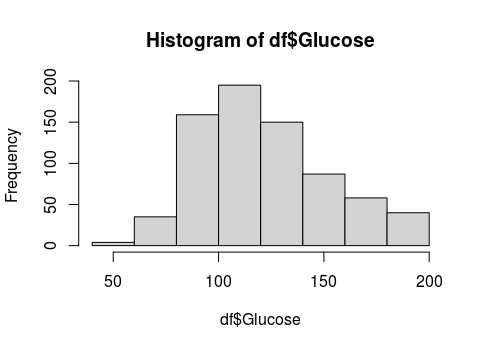
library(psych)

# Histogram --------------------------------------------------------------

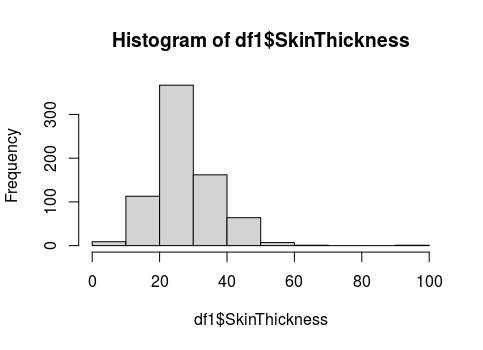
Visually explore these variable, you may need to look for the distribution of these variables using histograms. Treat the missing values accordingly.

head(df1)

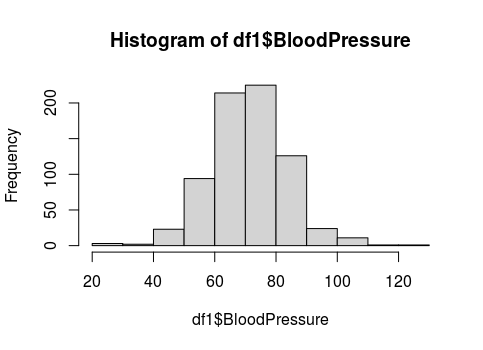
hist(df$Glucose)# the data is normal



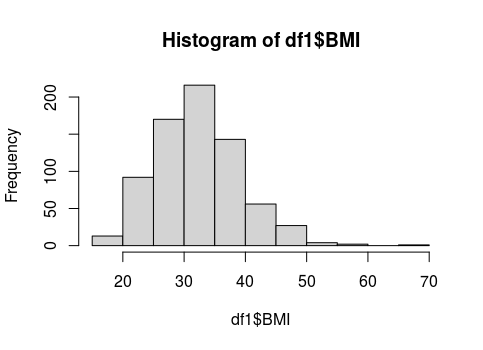
hist(df1$SkinThickness)# the data is normal



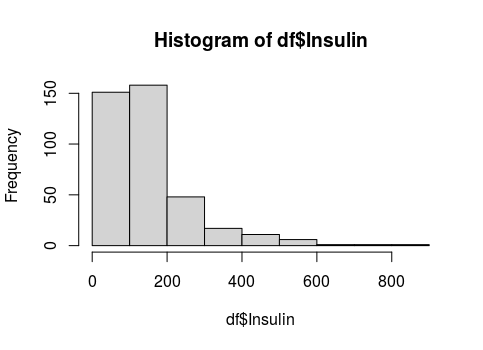
hist(df1$BloodPressure)# the data is normal



hist(df1$BMI)# the data is almost normal but there are some outliers which needs to treated



hist(df$Insulin)# the data is right skewed and we need to make it guassian



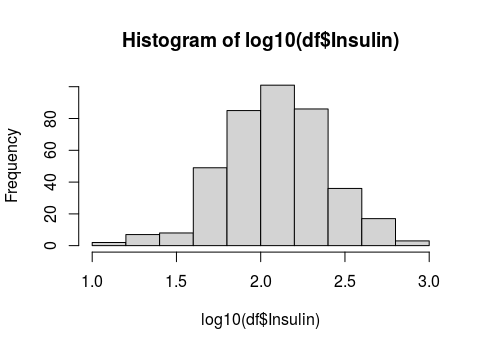
#Dealing with skewness

# as it was a right skewed data we have to take the log the data

df1$replace\_insulin=as.numeric(df1$replace\_insulin)# changing the data to numeric

str(df1)

hist(log10(df$Insulin))# the data became almost normal



df1$replace\_insulin=log10(df1$replace\_insulin)

head(df1)

hist(df1$replace\_insulin)# skewness is handled

# Data Description --------------------------------------------------------

t(summary(df1))

Pregnancies Min. : 0.000 1st Qu.: 1.000 Median : 3.000 Mean : 3.866 3rd Qu.: 6.000 Max. :17.000

Glucose Min. : 44.00 1st Qu.: 99.75 Median :117.00 Mean :121.88 3rd Qu.:142.00 Max. :199.00

BloodPressure Min. : 24.0 1st Qu.: 64.0 Median : 72.0 Mean : 72.4 3rd Qu.: 80.0 Max. :122.0

SkinThickness Min. : 7.00 1st Qu.:25.00 Median :29.00 Mean :29.13 3rd Qu.:33.00 Max. :99.00

BMI Min. :18.20 1st Qu.:27.50 Median :32.40 Mean :32.47 3rd Qu.:36.60 Max. :67.10

DiabetesPedigreeFunction Min. :0.0780 1st Qu.:0.2450 Median :0.3790 Mean :0.4748 3rd Qu.:0.6275 Max. :2.4200

Age Min. :21.00 1st Qu.:24.00 Median :29.00 Mean :33.35 3rd Qu.:41.00 Max. :81.00

Outcome Min. :0.0000 1st Qu.:0.0000 Median :0.0000 Mean :0.3439 3rd Qu.:1.0000 Max. :1.0000

replace\_insulin Min. :1.146 1st Qu.:2.021 Median :2.021 Mean :2.093 3rd Qu.:2.229 Max. :2.927

AgeBin Min. :1.000 1st Qu.:1.000 Median :1.000 Mean :1.605 3rd Qu.:2.000 Max. :5.000

#There are around 34 percent diabetic population the data set

# Majority of people are overweight which shows that there mught be a relationship between the Outcome and obesity

# The glucose value between 140 and 199 show that the person may suffer from prediabetes

#There are some outliers in Pregancies

# data types and the count of variables -----------------------------------

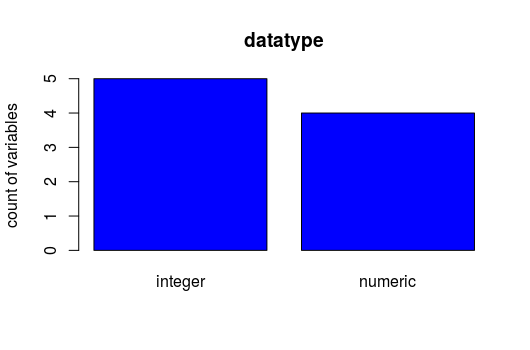
We observe integer as well as float data-type of variables in this dataset. Create a count (frequency) plot describing the data types and the count of variables.

datatype=lapply(df,class)#Generate List

class(datatype)

datatype=data.frame(unlist(datatype))#It will convert list in to data frame

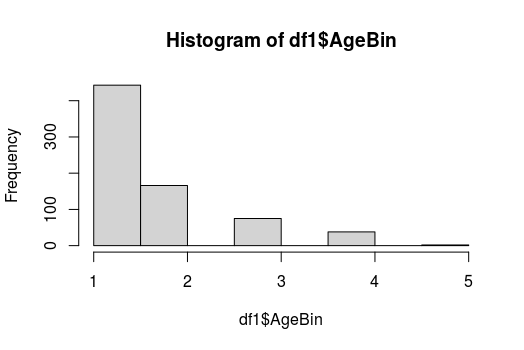
barplot(table(datatype),main = 'datatype',col='blue',ylab ='count of variables')



# Age ---------------------------------------------------------------------

df1$AgeBin=cut(df1$Age,breaks = 5,labels = FALSE)#binning age

hist(df1$AgeBin)#Data is biased towards Bin 1



Ap=df1 %>% group\_by(AgeBin) %>% summarise(Per=mean(Outcome))

Ap

|  |
| --- |
| AgeBin Per  *<int>* *<dbl>*  1 1 0.251  2 2 0.488  3 3 0.573  4 4 0.342  5 5 0.5 |
|  |
| |  | | --- | | > | |

#People above 35 are more prone to diabetes and need to care of themselves

# Glucose and Insulin-----------------------------------------------------------------

head(df1)

df1 %>% group\_by(Outcome) %>% summarise(Glucose=mean(Glucose))

|  |
| --- |
| Outcome Glucose  *<dbl>* *<dbl>*  1 0 111.  2 1 143. |
|  |
| |  | | --- | | > | |

#The diabetic person has high glucose level then a healthy person

df %>% group\_by(Outcome) %>% summarise(Insulin=mean(Insulin,na.rm = T))

# A tibble: 2 x 2

Outcome Insulin

*<dbl>* *<dbl>*

1 0 130.

2 1 207.

#this shows that glucose and insulin are correlated but we have to look further

# Pairplot----------------------------------------------------------------

Create scatter charts between the pair of variables to understand the relationships. Describe your findings.

library(ggplot2)

#install.packages('hrbrthemes')

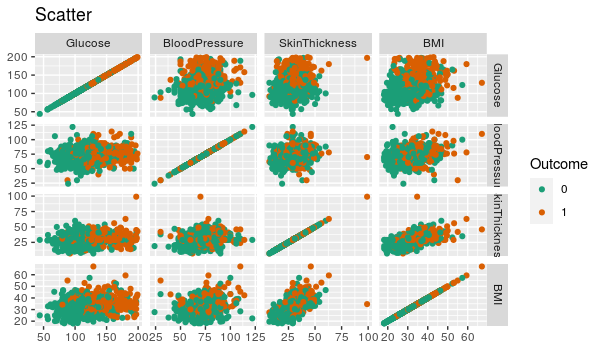
#install.packages("WVPlots")

library(WVPlots)

str(df1)

df1$Outcome=as.factor(df1$Outcome)

PairPlot(df1, colnames(df1)[2:5], "Scatter",group\_var = 'Outcome')



# as we can see that there is clear separtion between glucose and Bmi

#BMI and Skin thickness are correlated

# we can make the model with glucose there is a clear separtion

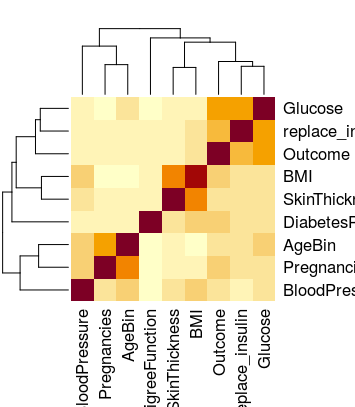
# HeatMap -----------------------------------------------------------------

Perform correlation analysis. Visually explore it using a heat map.

#library(corrplot)

df1$Outcome=as.numeric(df1$Outcome)

heatmap(cor(df1))



round(cor(df1),digits = 2)

|  |
| --- |
| Pregnancies Glucose BloodPressure SkinThickness BMI  Pregnancies 1.00 0.13 0.21 0.08 0.01  Glucose 0.13 1.00 0.22 0.20 0.22  BloodPressure 0.21 0.22 1.00 0.19 0.29  SkinThickness 0.08 0.20 0.19 1.00 0.56  BMI 0.01 0.22 0.29 0.56 1.00  DiabetesPedigreeFunction -0.03 0.14 0.00 0.11 0.15  Outcome 0.22 0.49 0.17 0.22 0.30  replace\_insulin 0.12 0.53 0.12 0.19 0.25  AgeBin 0.50 0.24 0.29 0.08 -0.02  DiabetesPedigreeFunction Outcome replace\_insulin AgeBin  Pregnancies -0.03 0.22 0.12 0.50  Glucose 0.14 0.49 0.53 0.24  BloodPressure 0.00 0.17 0.12 0.29  SkinThickness 0.11 0.22 0.19 0.08  BMI 0.15 0.30 0.25 -0.02  DiabetesPedigreeFunction 1.00 0.18 0.11 0.00  Outcome 0.18 1.00 0.44 0.19  replace\_insulin 0.11 0.44 1.00 0.17  AgeBin 0.00 0.19 0.17 1.00 |
|  |
| |  | | --- | | > | |

#We have to drop age colum as it highly correlated with AgeBin

#Most of the columns are important so we cant remove them

df1$Age=NULL

# Noramlizing the data ----------------------------------------------------------

normalize <- function(x) {

num <- x - min(x)

denom <- max(x) - min(x)

return(num/denom)

}

str(df1)

df1$Outcome=as.character(df1$Outcome)

head(df1)

df1$Outcome[df1$Outcome==1]=0

df1$Outcome[df1$Outcome==2]=1

lapply(df1[-7], normalize)

Norm=as.data.frame((lapply(df1[-7], normalize)))

head(Norm)

Norm$Outcome=df1$Outcome

head(Norm)

|  |
| --- |
|  |
| |  | | --- | |  | |

Pregnancies Glucose BloodPressure SkinThickness BMI DiabetesPedigreeFunction replace\_insulin

1 0.35294118 0.6709677 0.4897959 0.3043478 0.3149284 0.23441503 0.6080260

2 0.05882353 0.2645161 0.4285714 0.2391304 0.1717791 0.11656704 0.4912646

3 0.47058824 0.8967742 0.4081633 0.2391304 0.1042945 0.25362938 0.6080260

4 0.05882353 0.2903226 0.4285714 0.1739130 0.2024540 0.03800171 0.4642826

5 0.00000000 0.6000000 0.1632653 0.3043478 0.5092025 0.94363792 0.6058587

6 0.29411765 0.4645161 0.5102041 0.2391304 0.1513292 0.05251921 0.4912646

AgeBin Outcome

1 0.5 1

2 0.0 0

3 0.0 1

4 0.0 0

5 0.0 1

6 0.0 0

# Model Building ----------------------------------------------------------

# Spliting dataset

dim(Norm)

train <- Norm[1:600,]

test <- Norm[601:724,]

head(train)

head(test)

# KNN Model ---------------------------------------------------------------

As it is diabetic problem we need to maximize sensitivity as FN can be brutal for the person. Supppose a person has diabetes but he is predicted as a healthy person .Without proper medication and treatment hi health can deteriote further. So Recall is a better metric in this scenario.

library(mlbench)

library(caret)

#install.packages('caret')

library(class)

cl <- train$Outcome

model <- knn(train[-9], test[-9], cl, k = 5)

Accuracy <- mean(model == test$Outcome)

Accuracy

0.7983871

confusionMatrix(model,as.factor(test$Outcome),positive = '1')

35/(35+13)\*100

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 64 13

1 12 35

Accuracy : 0.7984

95% CI : (0.7169, 0.8651)

No Information Rate : 0.6129

P-Value [Acc > NIR] : 7.501e-06

Kappa : 0.5735

Mcnemar's Test P-Value : 1

Sensitivity : 0.7292

Specificity : 0.8421

Pos Pred Value : 0.7447

Neg Pred Value : 0.8312

Prevalence : 0.3871

Detection Rate : 0.2823

Detection Prevalence : 0.3790

Balanced Accuracy : 0.7856

'Positive' Class : 1

#Recall=74

# SVM ---------------------------------------------------------------------

#install.packages("e1071")

library(e1071)

library("caret")

library(ROCR)

#install.packages('ROCR')

tuned <- tune.svm(as.factor(Outcome) ~ ., data = train, gamma = 10^(-6:-1), cost = 10^(-1:1),type='C-classification',kernel='radial')

summary(tuned)

Parameter tuning of ‘svm’:

- sampling method: 10-fold cross validation

- best parameters:

gamma cost

0.1 1

- best performance: 0.1833333

- Detailed performance results:

gamma cost error dispersion

1 1e-06 0.1 0.3350000 0.06404570

2 1e-05 0.1 0.3350000 0.06404570

3 1e-04 0.1 0.3350000 0.06404570

4 1e-03 0.1 0.3350000 0.06404570

5 1e-02 0.1 0.3350000 0.06869597

6 1e-01 0.1 0.2166667 0.07535922

7 1e-06 1.0 0.3350000 0.06404570

8 1e-05 1.0 0.3350000 0.06404570

9 1e-04 1.0 0.3350000 0.06404570

10 1e-03 1.0 0.3066667 0.07745967

11 1e-02 1.0 0.2033333 0.05544433

12 1e-01 1.0 0.1833333 0.05211573

13 1e-06 10.0 0.3350000 0.06404570

14 1e-05 10.0 0.3350000 0.06404570

15 1e-04 10.0 0.3066667 0.07745967

16 1e-03 10.0 0.2050000 0.05776175

17 1e-02 10.0 0.1900000 0.07079792

18 1e-01 10.0 0.1850000 0.06687468

svmfit = svm(Outcome ~ ., data = train, kernel = "radial", cost = 1,type='C-classification',gamma=0.1)

print(svmfit)

Call:

svm(formula = Outcome ~ ., data = train, kernel = "radial", cost = 1,

type = "C-classification", gamma = 0.1)

Parameters:

SVM-Type: C-classification

SVM-Kernel: radial

cost: 1

Number of Support Vectors: 306

svm\_pred <- predict(svmfit, newdata = test,type='class')

confusionMatrix(svm\_pred,as.factor(test$Outcome),positive = '1')

Confusion Matrix and Statistics

Reference

Prediction 0 1

0 68 13

1 8 35

Accuracy : 0.8306

95% CI : (0.7528, 0.892)

No Information Rate : 0.6129

P-Value [Acc > NIR] : 1.207e-07

Kappa : 0.6361

Mcnemar's Test P-Value : 0.3827

Sensitivity : 0.7292

Specificity : 0.8947

Pos Pred Value : 0.8140

Neg Pred Value : 0.8395

Prevalence : 0.3871

Detection Rate : 0.2823

Detection Prevalence : 0.3468

Balanced Accuracy : 0.8120

'Positive' Class : 1

35/(35+13)\*100

#Recall 81

# Random forest -----------------------------------------------------------

#install.packages("randomForest")

library(randomForest)

str(train)

data.frame': 600 obs. of 9 variables:

$ Pregnancies : num 0.3529 0.0588 0.4706 0.0588 0 ...

$ Glucose : num 0.671 0.265 0.897 0.29 0.6 ...

$ BloodPressure : num 0.49 0.429 0.408 0.429 0.163 ...

$ SkinThickness : num 0.304 0.239 0.239 0.174 0.304 ...

$ BMI : num 0.315 0.172 0.104 0.202 0.509 ...

$ DiabetesPedigreeFunction: num 0.234 0.117 0.254 0.038 0.944 ...

$ replace\_insulin : num 0.608 0.491 0.608 0.464 0.606 ...

$ AgeBin : num 0.5 0 0 0 0 0 0 0.5 0 0.25 ...

$ Outcome : chr "1" "0" "1" "0" ...

library(plyr)

count(train$Outcome)

x freq

1 0 399

2 1 201

#install.packages('suOutcome

#library(superml)

library(caret)

library(mlbench)

tuned=tune.randomForest(as.factor(Outcome)~.,data = train,mtry = (1:8),ntree=100\*(1:5),seed=7)# GridSearch Cv to get the best params

summary(tuned)

Parameter tuning of ‘randomForest’:

- sampling method: 10-fold cross validation

- best parameters:

mtry ntree

8 400

- best performance: 0.1316667

- Detailed performance results:

mtry ntree error dispersion

1 1 100 0.1550000 0.03689324

2 2 100 0.1516667 0.03722637

3 3 100 0.1516667 0.04743416

4 4 100 0.1400000 0.03784308

5 5 100 0.1433333 0.05837300

6 6 100 0.1383333 0.05155005

7 7 100 0.1366667 0.05317105

8 8 100 0.1383333 0.04651430

9 1 200 0.1583333 0.04175916

10 2 200 0.1383333 0.02838231

11 3 200 0.1483333 0.04263685

12 4 200 0.1516667 0.04335470

13 5 200 0.1500000 0.04513355

14 6 200 0.1400000 0.04791969

15 7 200 0.1433333 0.05340273

16 8 200 0.1366667 0.03990730

17 1 300 0.1500000 0.02721655

18 2 300 0.1466667 0.02810913

19 3 300 0.1500000 0.03849002

20 4 300 0.1450000 0.04086261

21 5 300 0.1450000 0.04651430

22 6 300 0.1416667 0.04922235

23 7 300 0.1400000 0.04527011

24 8 300 0.1383333 0.04516773

25 1 400 0.1466667 0.02918650

26 2 400 0.1416667 0.03068156

27 3 400 0.1466667 0.04142523

28 4 400 0.1450000 0.04234631

29 5 400 0.1400000 0.05043025

30 6 400 0.1433333 0.04172219

31 7 400 0.1400000 0.04981447

32 8 400 0.1316667 0.04190672

33 1 500 0.1533333 0.02918650

34 2 500 0.1466667 0.03314763

35 3 500 0.1500000 0.04157397

36 4 500 0.1450000 0.04972145

37 5 500 0.1450000 0.05214533

38 6 500 0.1466667 0.04893937

39 7 500 0.1366667 0.04700932

40 8 500 0.1316667 0.04611446

# best parameters:

# mtry ntree

# 6 200

rf=randomForest(as.factor(Outcome) ~.,data=train,ntree=200,mtry=6,sampsize=c(200,200))

rf

Call:

randomForest(formula = as.factor(Outcome) ~ ., data = train, ntree = 200, mtry = 6, sampsize = c(200, 200))

Type of random forest: classification

Number of trees: 200

No. of variables tried at each split: 6

OOB estimate of error rate: 15.5%

Confusion matrix:

0 1 class.error

0 342 57 0.1428571

1 36 165 0.1791045

count(test$Outcome)

rft=randomForest(as.factor(Outcome) ~.,data=test,ntree=200,mtry=6)

rft

rf\_pred <- predict(rf, newdata = test,type='class')

confusionMatrix(rf\_pred,as.factor(test$Outcome),positive = '1')

(43/48)\*100

#recall 89%

Reference

Prediction 0 1

0 60 5

1 16 43

Accuracy : 0.8306

95% CI : (0.7528, 0.892)

No Information Rate : 0.6129

P-Value [Acc > NIR] : 1.207e-07

Kappa : 0.6575

Mcnemar's Test P-Value : 0.0291

Sensitivity : 0.8958

Specificity : 0.7895

Pos Pred Value : 0.7288

Neg Pred Value : 0.9231

Prevalence : 0.3871

Detection Rate : 0.3468

Detection Prevalence : 0.4758

Balanced Accuracy : 0.8427

'Positive' Class : 1

## Tablaeu Dashboard

