Indian Diabetes Analysis

CSIS3360-001 Group #5

25/03/2021

## ASSOCIATION RULES

### 1. Loading data from .csv file

Uploading a dataset in R workspace and take a look at the first 6 rows:

diabetes\_data <- read.csv('diabetes(data).csv')  
head(diabetes\_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

str(diabetes\_data) *# check the loaded data frame*

## 'data.frame': 771 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 ...  
## $ BloodPressure : int 72 66 64 66 40 74 50 0 70 96 ...  
## $ SkinThickness : 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 ...  
## $ DiabetesPedigreeFunction: 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 ...

### 2. Installing & opening required packages

*# Installing Packages*   
*#install.packages("arules")*   
*#install.packages("arulesViz")*   
  
*#Loading package*   
library(arules)

## Loading required package: Matrix

##   
## Attaching package: 'arules'

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

library(arulesViz)

### 3.Train the apriori and fit the model visualize results

*# Fitting model*   
*# Training Apriori on the dataset*   
set.seed = 220 *# Setting seed*   
associa\_rules = apriori(data = diabetes\_data,   
 parameter = list(support = 0.004,   
 confidence = 0.2))

## Warning: Column(s) 1, 2, 3, 4, 5, 6, 7, 8, 9 not logical or factor. Applying  
## default discretization (see '? discretizeDF').

## Warning in discretize(x = c(0L, 0L, 0L, 94L, 168L, 0L, 88L, 0L, 543L, 0L, : The calculated breaks are: 0, 0, 95, 846  
## Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

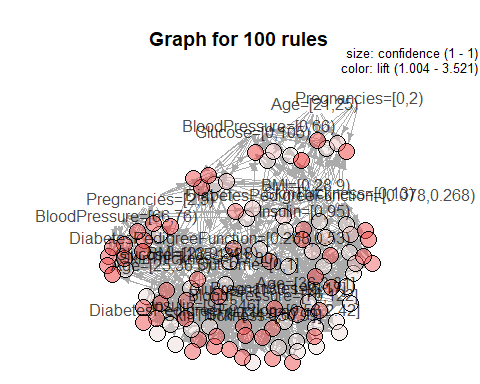
## Warning in discretize(x = c(1L, 0L, 1L, 0L, 1L, 0L, 1L, 0L, 1L, 1L, 0L, : The calculated breaks are: 0, 0, 1, 1  
## Only unique breaks are used reducing the number of intervals. Look at ? discretize for details.

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.2 0.1 1 none FALSE TRUE 5 0.004 1  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 3   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[24 item(s), 771 transaction(s)] done [0.00s].  
## sorting and recoding items ... [24 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 6 7 8 9 done [0.01s].  
## writing ... [119239 rule(s)] done [0.02s].  
## creating S4 object ... done [0.04s].

### 4. Visualizing the result

plot(associa\_rules, method = "graph",   
 measure = "confidence", shading = "lift")

## Warning: plot: Too many rules supplied. Only plotting the best 100 rules using  
## 'confidence' (change control parameter max if needed)



## Classification (Naive-Bayes model)

### 1. Installing and adjusting packages

*#install.packages("naivebayes")*  
*#install.packages("forecast")*  
*#install.packages("gains")*  
*#install.packages("caret")*  
*#install.packages("e1071")*  
*#install.packages("pROC")*  
  
*#loading packages for Classification*  
library("knitr")  
library("naivebayes")

## naivebayes 0.9.7 loaded

library("forecast")

## Registered S3 method overwritten by 'quantmod':  
## method from  
## as.zoo.data.frame zoo

library("gains")  
library("caret")

## Loading required package: lattice

## Loading required package: ggplot2

library("e1071")

## Registered S3 methods overwritten by 'proxy':  
## method from   
## print.registry\_field registry  
## print.registry\_entry registry

library("pROC")

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

##   
## Attaching package: 'pROC'

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

### 2. Loading data from .csv file

diabetes\_data <- read.csv('diabetes(data).csv', stringsAsFactors=T)  
str(diabetes\_data) *# check the loaded data frame*

## 'data.frame': 771 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 ...  
## $ BloodPressure : int 72 66 64 66 40 74 50 0 70 96 ...  
## $ SkinThickness : 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 ...  
## $ DiabetesPedigreeFunction: 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 ...

### 3. Data pre-processing works.

Factorizing each Column (because it is simple numerical data, not a factor).

***## Factorizing "Pregnancies" Column.***  
diabetes\_data <- within(diabetes\_data, {  
 stat\_Preg = character(0)  
 stat\_Preg[ Pregnancies == 0 ] = "0"  
 stat\_Preg[ Pregnancies >= 1 & Pregnancies < 4 ] = "Between 1 and 3"  
 stat\_Preg[ Pregnancies >= 4 & Pregnancies < 7 ] = "Between 4 and 6"  
 stat\_Preg[ Pregnancies >= 7 & Pregnancies < 10 ] = "Between 7 and 9"  
 stat\_Preg[ Pregnancies >= 10 ] = "Greater than or equal to 10"  
 stat\_Preg = factor(stat\_Preg, level = c("0", "Between 1 and 3", "Between 4 and 6", "Between 7 and 9", "Greater than or equal to 10"))  
})  
  
***## Factorizing "Glucose" Column.***  
diabetes\_data <- within(diabetes\_data, {  
 stat\_Glu = character(0)  
 stat\_Glu[ Glucose < 41 ] = "Less than 41"  
 stat\_Glu[ Glucose >= 41 & Glucose < 81 ] = "Between 41 and 80"  
 stat\_Glu[ Glucose >= 81 & Glucose < 121 ] = "Between 81 and 120"  
 stat\_Glu[ Glucose >= 121 & Glucose < 161 ] = "Between 121 and 160"  
 stat\_Glu[ Glucose >= 161 & Glucose < 201 ] = "Between 161 and 200"  
 stat\_Glu = factor(stat\_Glu, level = c("Less than 41", "Between 41 and 80", "Between 81 and 120", "Between 121 and 160", "Between 161 and 200"))  
})  
  
***## Factorizing "BloodPressure" Column.***  
diabetes\_data <- within(diabetes\_data, {  
 stat\_Blood = character(0)  
 stat\_Blood[ BloodPressure < 26 ] = "Less than 26"  
 stat\_Blood[ BloodPressure >= 26 & BloodPressure < 51 ] = "Between 26 and 50"  
 stat\_Blood[ BloodPressure >= 51 & BloodPressure < 76 ] = "Between 51 and 75"  
 stat\_Blood[ BloodPressure >= 76 & BloodPressure < 101 ] = "Between 76 and 100"  
 stat\_Blood[ BloodPressure >= 101 & BloodPressure < 126 ] = "Between 101 and 125"  
 stat\_Blood = factor(stat\_Blood, level = c("Less than 26", "Between 26 and 50", "Between 51 and 75", "Between 76 and 100", "Between 101 and 125"))  
})  
  
***## Factorizing "SkinThickness" Column.***  
diabetes\_data <- within(diabetes\_data, {  
 stat\_Skin = character(0)  
 stat\_Skin[ SkinThickness < 21 ] = "Less than 21"  
 stat\_Skin[ SkinThickness >= 21 & SkinThickness < 41 ] = "Between 21 and 40"  
 stat\_Skin[ SkinThickness >= 41 & SkinThickness < 61 ] = "Between 41 and 60"  
 stat\_Skin[ SkinThickness >= 61 & SkinThickness < 81 ] = "Between 61 and 80"  
 stat\_Skin[ SkinThickness >= 81 & SkinThickness < 101 ] = "Between 81 and 100"  
 stat\_Skin = factor(stat\_Skin, level = c("Less than 21", "Between 21 and 40", "Between 41 and 60", "Between 61 and 80", "Between 81 and 100"))  
})  
  
***## Factorizing "Insulin" Column.***  
diabetes\_data <- within(diabetes\_data, {  
 stat\_Ins = character(0)  
 stat\_Ins[ Insulin < 201 ] = "Less than 201"  
 stat\_Ins[ Insulin >= 201 & Insulin < 401 ] = "Between 201 and 400"  
 stat\_Ins[ Insulin >= 401 & Insulin < 601 ] = "Between 401 and 600"  
 stat\_Ins[ Insulin >= 601 & Insulin < 801 ] = "Between 601 and 800"  
 stat\_Ins[ Insulin >= 801 & Insulin < 1001 ] = "Between 801 and 1000"  
 stat\_Ins = factor(stat\_Ins, level = c("Less than 201", "Between 201 and 400", "Between 401 and 600", "Between 601 and 800", "Between 801 and 1000"))  
})  
  
***## Factorizing "BMI" Column.***  
diabetes\_data <- within(diabetes\_data, {  
 stat\_BMI = character(0)  
 stat\_BMI[ BMI < 16 ] = "Less than 16"  
 stat\_BMI[ BMI >= 16 & BMI < 31 ] = "Between 16 and 30"  
 stat\_BMI[ BMI >= 31 & BMI < 46 ] = "Between 31 and 45"  
 stat\_BMI[ BMI >= 46 & BMI < 61 ] = "Between 46 and 60"  
 stat\_BMI[ BMI >= 61 & BMI < 81 ] = "Between 61 and 80"  
 stat\_BMI = factor(stat\_BMI, level = c("Less than 16", "Between 16 and 30", "Between 31 and 45", "Between 46 and 60", "Between 61 and 80"))  
})  
  
***## Factorizing "DiabetesPedigreeFunction" Column.***  
diabetes\_data <- within(diabetes\_data, {  
 stat\_PedFunc = character(0)  
 stat\_PedFunc[ DiabetesPedigreeFunction < 0.6 ] = "Less than 0.6"  
 stat\_PedFunc[ DiabetesPedigreeFunction >= 0.6 & DiabetesPedigreeFunction < 1.1 ] = "Between 0.6 and 1.0"  
 stat\_PedFunc[ DiabetesPedigreeFunction >= 1.1 & DiabetesPedigreeFunction < 1.6 ] = "Between 1.1 and 1.5"  
 stat\_PedFunc[ DiabetesPedigreeFunction >= 1.6 & DiabetesPedigreeFunction < 2.1] = "Between 1.6 and 2.0"  
 stat\_PedFunc[ DiabetesPedigreeFunction >= 2.1 & DiabetesPedigreeFunction < 2.6 ] = "Between 2.1 and 2.5"  
 stat\_PedFunc = factor(stat\_PedFunc, level = c("Less than 0.6", "Between 0.6 and 1.0", "Between 1.1 and 1.5", "Between 1.6 and 2.0", "Between 2.1 and 2.5"))  
})  
  
***## Factorizing "Age" Column.***  
diabetes\_data <- within(diabetes\_data, {  
 stat\_Age = character(0)  
 stat\_Age[ Age < 21 ] = "Less than 21"  
 stat\_Age[ Age >= 21 & Age < 41 ] = "Between 21 and 40"  
 stat\_Age[ Age >= 41 & Age < 61 ] = "Between 41 and 60"  
 stat\_Age[ Age >= 61 & Age < 81 ] = "Between 61 and 80"  
 stat\_Age[ Age >= 81 & Age < 101 ] = "Between 81 and 100"  
 stat\_Age = factor(stat\_Age, level = c("Less than 21", "Between 21 and 40", "Between 41 and 60", "Between 61 and 80", "Between 81 and 100"))  
})  
  
diabetes\_data[1:10, c(10:12)]

## stat\_Preg stat\_Glu stat\_Blood  
## 1 Between 4 and 6 Between 121 and 160 Between 51 and 75  
## 2 Between 1 and 3 Between 81 and 120 Between 51 and 75  
## 3 Between 7 and 9 Between 161 and 200 Between 51 and 75  
## 4 Between 1 and 3 Between 81 and 120 Between 51 and 75  
## 5 0 Between 121 and 160 Between 26 and 50  
## 6 Between 4 and 6 Between 81 and 120 Between 51 and 75  
## 7 Between 1 and 3 Between 41 and 80 Between 26 and 50  
## 8 Greater than or equal to 10 Between 81 and 120 Less than 26  
## 9 Between 1 and 3 Between 161 and 200 Between 51 and 75  
## 10 Between 7 and 9 Between 121 and 160 Between 76 and 100

diabetes\_data[1:10, c(13:15)]

## stat\_Skin stat\_Ins stat\_BMI  
## 1 Between 21 and 40 Less than 201 Between 31 and 45  
## 2 Between 21 and 40 Less than 201 Between 16 and 30  
## 3 Less than 21 Less than 201 Between 16 and 30  
## 4 Between 21 and 40 Less than 201 Between 16 and 30  
## 5 Between 21 and 40 Less than 201 Between 31 and 45  
## 6 Less than 21 Less than 201 Between 16 and 30  
## 7 Between 21 and 40 Less than 201 Between 31 and 45  
## 8 Less than 21 Less than 201 Between 31 and 45  
## 9 Between 41 and 60 Between 401 and 600 Between 16 and 30  
## 10 Less than 21 Less than 201 Less than 16

diabetes\_data[1:10, c(16:17, 9)]

## stat\_PedFunc stat\_Age Outcome  
## 1 Between 0.6 and 1.0 Between 41 and 60 1  
## 2 Less than 0.6 Between 21 and 40 0  
## 3 Between 0.6 and 1.0 Between 21 and 40 1  
## 4 Less than 0.6 Between 21 and 40 0  
## 5 Between 2.1 and 2.5 Between 21 and 40 1  
## 6 Less than 0.6 Between 21 and 40 0  
## 7 Less than 0.6 Between 21 and 40 1  
## 8 Less than 0.6 Between 21 and 40 0  
## 9 Less than 0.6 Between 41 and 60 1  
## 10 Less than 0.6 Between 41 and 60 1

Set the Factors for Outcome column

diabetes\_data$Outcome <- factor(diabetes\_data$Outcome, levels=c("0","1"))

### 4. Presenting distribution table with X - Y factors

1. Distribution table (Pregnancies - Satisfaction)

xtabs(~stat\_Preg+Outcome, data = diabetes\_data)

## Outcome  
## stat\_Preg 0 1  
## 0 73 38  
## Between 1 and 3 238 75  
## Between 4 and 6 115 60  
## Between 7 and 9 46 65  
## Greater than or equal to 10 28 30

1. Distribution table (Glucose - Satisfaction)

xtabs(~stat\_Glu+Outcome, data = diabetes\_data)

## Outcome  
## stat\_Glu 0 1  
## Less than 41 3 2  
## Between 41 and 80 40 2  
## Between 81 and 120 303 69  
## Between 121 and 160 136 112  
## Between 161 and 200 18 83

1. Distribution table (BloodPressure - Satisfaction)

xtabs(~stat\_Blood+Outcome, data = diabetes\_data)

## Outcome  
## stat\_Blood 0 1  
## Less than 26 20 16  
## Between 26 and 50 20 8  
## Between 51 and 75 296 119  
## Between 76 and 100 159 117  
## Between 101 and 125 5 8

1. Distribution table (SkinThickness - Satisfaction)

xtabs(~stat\_Skin+Outcome, data = diabetes\_data)

## Outcome  
## stat\_Skin 0 1  
## Less than 21 246 105  
## Between 21 and 40 217 126  
## Between 41 and 60 37 35  
## Between 61 and 80 0 1  
## Between 81 and 100 0 1

1. Distribution table (Insulin - Satisfaction)

xtabs(~stat\_Ins+Outcome, data = diabetes\_data)

## Outcome  
## stat\_Ins 0 1  
## Less than 201 461 222  
## Between 201 and 400 31 34  
## Between 401 and 600 6 11  
## Between 601 and 800 2 0  
## Between 801 and 1000 0 1

1. Distribution table (BMI - Satisfaction)

xtabs(~stat\_BMI+Outcome, data = diabetes\_data)

## Outcome  
## stat\_BMI 0 1  
## Less than 16 9 2  
## Between 16 and 30 264 68  
## Between 31 and 45 218 182  
## Between 46 and 60 9 15  
## Between 61 and 80 0 1

1. Distribution table (DiabetesPedigreeFunction - Satisfaction)

xtabs(~stat\_PedFunc+Outcome, data = diabetes\_data)

## Outcome  
## stat\_PedFunc 0 1  
## Less than 0.6 390 172  
## Between 0.6 and 1.0 93 71  
## Between 1.1 and 1.5 11 21  
## Between 1.6 and 2.0 5 1  
## Between 2.1 and 2.5 1 3

1. Distribution table (Age - Satisfaction)

xtabs(~stat\_Age+Outcome, data = diabetes\_data)

## Outcome  
## stat\_Age 0 1  
## Less than 21 0 0  
## Between 21 and 40 408 166  
## Between 41 and 60 72 95  
## Between 61 and 80 19 7  
## Between 81 and 100 1 0

### 5. Creating training and validating data for Naive-Bayes modeling

Setting ‘select’ variable to fix columns which will be used for testing (Xs).

* 10th column = Pregnancies
* 11th column = Glucose
* 12th column = BloodPressure
* 13th column = SkinThickness
* 14th column = Insulin
* 15th column = BMI
* 16th column = DiabetesPedigreeFunction
* 17th column = Age
* 9th column = Outcome

selected.var <- c(10:17, 9)

making index variable for splitting up training and validating data frame.

* size of index = 768 (without 3 rows for testing)
* range of index = from 1 to 768 (without 3 rows for testing)

train.index <- sample(c(1:768), 768)

making data frame for training by using train.index and selected.var(columns)

train.df <- diabetes\_data[train.index, selected.var]

making data frame for validating by using -train.index and selected.var(columns)

valid.df <- diabetes\_data[-train.index, selected.var]

checking the structures of each data frame

str(train.df)

## 'data.frame': 768 obs. of 9 variables:  
## $ stat\_Preg : Factor w/ 5 levels "0","Between 1 and 3",..: 2 1 2 2 2 1 5 2 5 1 ...  
## $ stat\_Glu : Factor w/ 5 levels "Less than 41",..: 4 4 4 4 3 2 4 3 4 3 ...  
## $ stat\_Blood : Factor w/ 5 levels "Less than 26",..: 3 4 3 4 3 4 4 3 3 4 ...  
## $ stat\_Skin : Factor w/ 5 levels "Less than 21",..: 1 2 1 3 1 2 2 2 2 1 ...  
## $ stat\_Ins : Factor w/ 5 levels "Less than 201",..: 1 2 1 1 1 1 1 1 1 1 ...  
## $ stat\_BMI : Factor w/ 5 levels "Less than 16",..: 2 3 3 4 2 3 3 3 3 3 ...  
## $ stat\_PedFunc: Factor w/ 5 levels "Less than 0.6",..: 1 1 1 1 1 1 1 1 1 2 ...  
## $ stat\_Age : Factor w/ 5 levels "Less than 21",..: 2 2 2 3 2 2 3 2 3 2 ...  
## $ Outcome : Factor w/ 2 levels "0","1": 1 1 2 2 1 1 1 1 2 1 ...

str(valid.df)

## 'data.frame': 3 obs. of 9 variables:  
## $ stat\_Preg : Factor w/ 5 levels "0","Between 1 and 3",..: 2 3 3  
## $ stat\_Glu : Factor w/ 5 levels "Less than 41",..: 4 2 4  
## $ stat\_Blood : Factor w/ 5 levels "Less than 26",..: 4 3 4  
## $ stat\_Skin : Factor w/ 5 levels "Less than 21",..: 2 2 2  
## $ stat\_Ins : Factor w/ 5 levels "Less than 201",..: 1 1 1  
## $ stat\_BMI : Factor w/ 5 levels "Less than 16",..: 2 3 3  
## $ stat\_PedFunc: Factor w/ 5 levels "Less than 0.6",..: 1 2 1  
## $ stat\_Age : Factor w/ 5 levels "Less than 21",..: 3 2 3  
## $ Outcome : Factor w/ 2 levels "0","1": NA NA NA

### 6.Running Naive-Bayes classification

Initializing result variable with naiveBayes function by using train data frame

results.nb <- naive\_bayes(Outcome ~., data = train.df, laplace = 1)  
results.nb

##   
## ================================== Naive Bayes ==================================   
##   
## Call:   
## naive\_bayes.formula(formula = Outcome ~ ., data = train.df, laplace = 1)  
##   
## ---------------------------------------------------------------------------------   
##   
## Laplace smoothing: 1  
##   
## ---------------------------------------------------------------------------------   
##   
## A priori probabilities:   
##   
## 0 1   
## 0.6510417 0.3489583   
##   
## ---------------------------------------------------------------------------------   
##   
## Tables:   
##   
## ---------------------------------------------------------------------------------   
## ::: stat\_Preg (Categorical)   
## ---------------------------------------------------------------------------------   
##   
## stat\_Preg 0 1  
## 0 0.14653465 0.14285714  
## Between 1 and 3 0.47326733 0.27838828  
## Between 4 and 6 0.22970297 0.22344322  
## Between 7 and 9 0.09306931 0.24175824  
## Greater than or equal to 10 0.05742574 0.11355311  
##   
## ---------------------------------------------------------------------------------   
## ::: stat\_Glu (Categorical)   
## ---------------------------------------------------------------------------------   
##   
## stat\_Glu 0 1  
## Less than 41 0.007920792 0.010989011  
## Between 41 and 80 0.081188119 0.010989011  
## Between 81 and 120 0.601980198 0.256410256  
## Between 121 and 160 0.271287129 0.413919414  
## Between 161 and 200 0.037623762 0.307692308  
##   
## ---------------------------------------------------------------------------------   
## ::: stat\_Blood (Categorical)   
## ---------------------------------------------------------------------------------   
##   
## stat\_Blood 0 1  
## Less than 26 0.04158416 0.06227106  
## Between 26 and 50 0.04158416 0.03296703  
## Between 51 and 75 0.58811881 0.43956044  
## Between 76 and 100 0.31683168 0.43223443  
## Between 101 and 125 0.01188119 0.03296703  
##   
## ---------------------------------------------------------------------------------   
## ::: stat\_Skin (Categorical)   
## ---------------------------------------------------------------------------------   
##   
## stat\_Skin 0 1  
## Less than 21 0.489108911 0.388278388  
## Between 21 and 40 0.431683168 0.465201465  
## Between 41 and 60 0.075247525 0.131868132  
## Between 61 and 80 0.001980198 0.007326007  
## Between 81 and 100 0.001980198 0.007326007  
##   
## ---------------------------------------------------------------------------------   
## ::: stat\_Ins (Categorical)   
## ---------------------------------------------------------------------------------   
##   
## stat\_Ins 0 1  
## Less than 201 0.914851485 0.816849817  
## Between 201 and 400 0.063366337 0.128205128  
## Between 401 and 600 0.013861386 0.043956044  
## Between 601 and 800 0.005940594 0.003663004  
## Between 801 and 1000 0.001980198 0.007326007  
##   
## ---------------------------------------------------------------------------------  
##   
## # ... and 3 more tables  
##   
## ---------------------------------------------------------------------------------

### 7. Creating probability tables to compare values of each factor

results\_Preg.t <- table(train.df$stat\_Preg, train.df$Outcome)  
results\_Glu.t <- table(train.df$stat\_Glu, train.df$Outcome)  
results\_Blood.t <- table(train.df$stat\_Blood, train.df$Outcome)  
results\_Skin.t <- table(train.df$stat\_Skin, train.df$Outcome)  
results\_Ins.t <- table(train.df$stat\_Ins, train.df$Outcome)  
results\_BMI.t <- table(train.df$stat\_BMI, train.df$Outcome)  
results\_Ped.t <- table(train.df$stat\_PedFunc, train.df$Outcome)  
results\_Age.t <- table(train.df$stat\_Age, train.df$Outcome)  
  
*# margin=2 means that considering 'column' as 100%.*  
results.p1 <- prop.table(results\_Preg.t, margin = 2)   
results.p2 <- prop.table(results\_Glu.t, margin = 2)  
results.p3 <- prop.table(results\_Blood.t, margin = 2)  
results.p4 <- prop.table(results\_Skin.t, margin = 2)  
results.p5 <- prop.table(results\_Ins.t, margin = 2)  
results.p6 <- prop.table(results\_BMI.t, margin = 2)  
results.p7 <- prop.table(results\_Ped.t, margin = 2)  
results.p8 <- prop.table(results\_Age.t, margin = 2)

1. Probability table (Pregnancies - Satisfaction)

results.p1

##   
## 0 1  
## 0 0.1460000 0.1417910  
## Between 1 and 3 0.4760000 0.2798507  
## Between 4 and 6 0.2300000 0.2238806  
## Between 7 and 9 0.0920000 0.2425373  
## Greater than or equal to 10 0.0560000 0.1119403

1. Probability table (Glucose - Satisfaction)

results.p2

##   
## 0 1  
## Less than 41 0.006000000 0.007462687  
## Between 41 and 80 0.080000000 0.007462687  
## Between 81 and 120 0.606000000 0.257462687  
## Between 121 and 160 0.272000000 0.417910448  
## Between 161 and 200 0.036000000 0.309701493

1. Probability table (BloodPressure - Satisfaction)

results.p3

##   
## 0 1  
## Less than 26 0.04000000 0.05970149  
## Between 26 and 50 0.04000000 0.02985075  
## Between 51 and 75 0.59200000 0.44402985  
## Between 76 and 100 0.31800000 0.43656716  
## Between 101 and 125 0.01000000 0.02985075

1. Probability table (SkinThickness - Satisfaction)

results.p4

##   
## 0 1  
## Less than 21 0.492000000 0.391791045  
## Between 21 and 40 0.434000000 0.470149254  
## Between 41 and 60 0.074000000 0.130597015  
## Between 61 and 80 0.000000000 0.003731343  
## Between 81 and 100 0.000000000 0.003731343

1. Probability table (Insulin - Satisfaction)

results.p5

##   
## 0 1  
## Less than 201 0.922000000 0.828358209  
## Between 201 and 400 0.062000000 0.126865672  
## Between 401 and 600 0.012000000 0.041044776  
## Between 601 and 800 0.004000000 0.000000000  
## Between 801 and 1000 0.000000000 0.003731343

1. Probability table (BMI - Satisfaction)

results.p6

##   
## 0 1  
## Less than 16 0.018000000 0.007462687  
## Between 16 and 30 0.528000000 0.253731343  
## Between 31 and 45 0.436000000 0.679104478  
## Between 46 and 60 0.018000000 0.055970149  
## Between 61 and 80 0.000000000 0.003731343

1. Probability table (DiabetesPedigreeFunction - Satisfaction)

results.p7

##   
## 0 1  
## Less than 0.6 0.780000000 0.641791045  
## Between 0.6 and 1.0 0.186000000 0.264925373  
## Between 1.1 and 1.5 0.022000000 0.078358209  
## Between 1.6 and 2.0 0.010000000 0.003731343  
## Between 2.1 and 2.5 0.002000000 0.011194030

1. Probability table (Age - Satisfaction)

results.p8

##   
## 0 1  
## Less than 21 0.0000000 0.0000000  
## Between 21 and 40 0.8160000 0.6194030  
## Between 41 and 60 0.1440000 0.3544776  
## Between 61 and 80 0.0380000 0.0261194  
## Between 81 and 100 0.0020000 0.0000000

### 8. Predicting probabilities with validating data

* Q1. Assuming results as a probability of a 46-year-old patient with only one pregnancy with 121 glucose concentration, 88 blood pressure, 32 skin thickness, 82 insulin, 24.2 BMI and 0.286 diabetesPedigreeFunction,
* Q2. Assuming results as a probability of a 37-year-old patient with 4 pregnancies with 57 glucose concentration, 61 blood pressure, 24 skin thickness, 190 insulin, 35.5 BMI and 0.871 diabetesPedigreeFunction,
* Q3. Assuming results as a probability of a 52-year-old patient with 5 pregnancies with 154 glucose concentration, 89 blood pressure, 33 skin thickness, 164 insulin, 42.7 BMI and 0.423 diabetesPedigreeFunction,

*# newdata = new data for prediction*  
*# type = raw(or, prob = probability prediction) , class (belonging class)*  
  
pred.prob <- predict(results.nb, newdata = valid.df, type = "prob")  
pred.class <- predict(results.nb, newdata = valid.df, type = "class")  
  
*# actual = actual data*  
*# predicted = predicted data*   
  
df <- data.frame(actual = valid.df$Outcome,  
 predicted = pred.class,   
 pred.prob)  
  
df *# check the result*

## actual predicted X0 X1  
## 1 <NA> 0 0.6228334 0.37716657  
## 2 <NA> 0 0.9225480 0.07745199  
## 3 <NA> 1 0.2373145 0.76268549

#### ANSWER : This result makes it possible to predict that the first two patients will not have diabates, and the last patient will have diabetes. [0, 0, 1]

### 9. Confusion Matrix

The misclassification is 0.2395833

***## Confusion Matrix - train data***  
p1 <- predict(results.nb, train.df)  
tab1 <- table(p1, train.df$Outcome)  
  
tab1 *# check misclassification table*

##   
## p1 0 1  
## 0 424 108  
## 1 76 160

*# misclassification*  
1 - sum(diag(tab1)) / sum(tab1)

## [1] 0.2395833

The misclassification is NaN ()

***## Confusion Matrix - test data***  
p2 <- predict(results.nb, valid.df)  
tab2 <- table(p2, valid.df$Outcome)  
  
tab2 *# check misclassification table*

##   
## p2 0 1  
## 0 0 0  
## 1 0 0

*# misclassification*  
1 - sum(diag(tab2)) / sum(tab2)

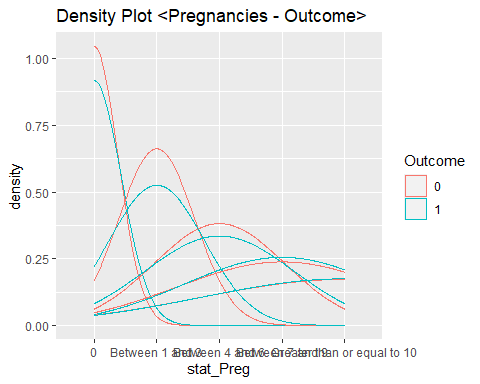
## [1] NaN

## Plots

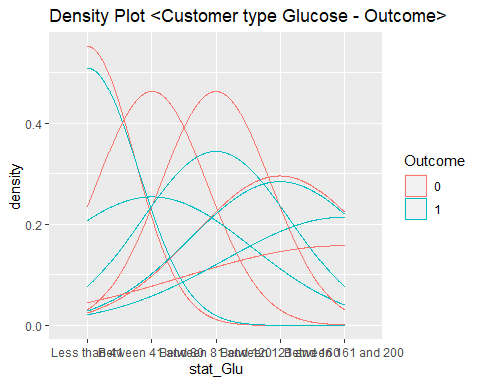
### 1. Density Plots for each distribution table

Result plot of each distribution table

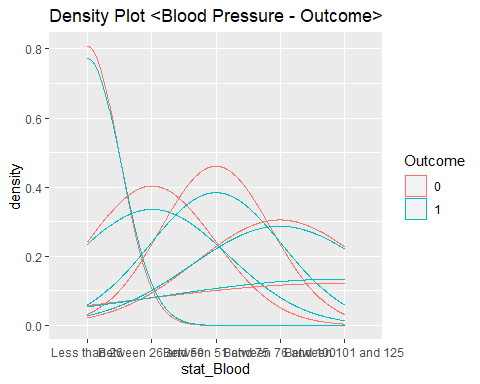
ggplot(data=train.df) +   
 geom\_density(mapping=aes(x=stat\_Preg, colour = Outcome)) +  
 ggtitle("Density Plot <Pregnancies - Outcome>")



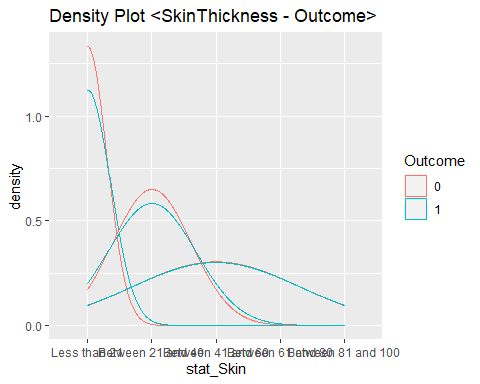
ggplot(data=train.df) +   
 geom\_density(mapping=aes(x=stat\_Glu, colour = Outcome)) +  
 ggtitle("Density Plot <Customer type Glucose - Outcome>")



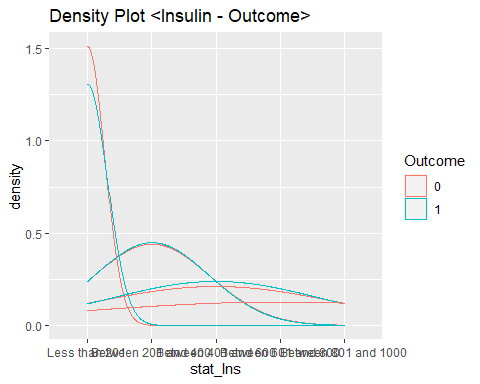
ggplot(data=train.df) +   
 geom\_density(mapping=aes(x=stat\_Blood, colour = Outcome)) +  
 ggtitle("Density Plot <Blood Pressure - Outcome>")



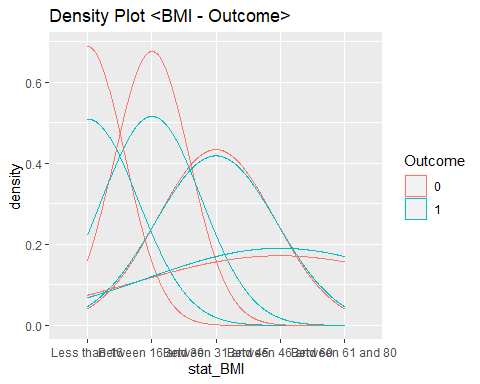
ggplot(data=train.df) +   
 geom\_density(mapping=aes(x=stat\_Skin, colour = Outcome)) +  
 ggtitle("Density Plot <SkinThickness - Outcome>")



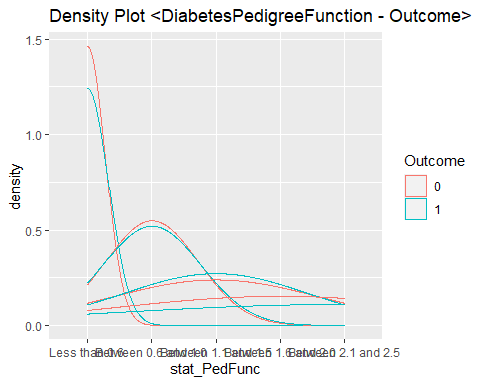
ggplot(data=train.df) +   
 geom\_density(mapping=aes(x=stat\_Ins, colour = Outcome)) +  
 ggtitle("Density Plot <Insulin - Outcome>")



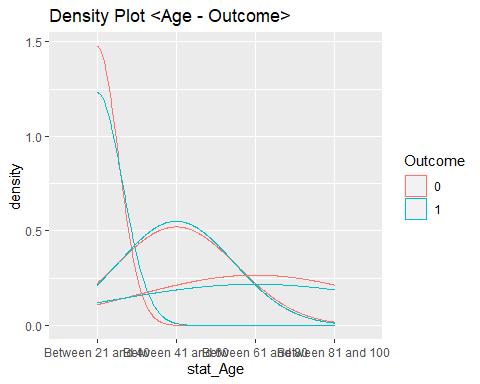
ggplot(data=train.df) +   
 geom\_density(mapping=aes(x=stat\_BMI, colour = Outcome)) +  
 ggtitle("Density Plot <BMI - Outcome>")



ggplot(data=train.df) +   
 geom\_density(mapping=aes(x=stat\_PedFunc, colour = Outcome)) +  
 ggtitle("Density Plot <DiabetesPedigreeFunction - Outcome>")



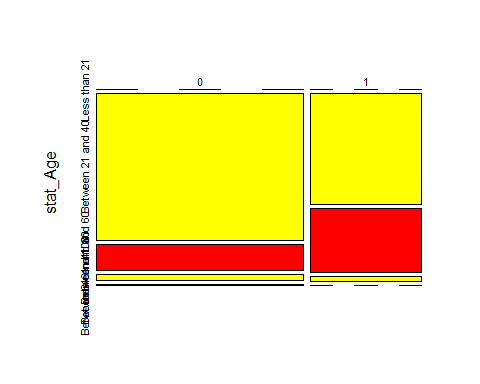
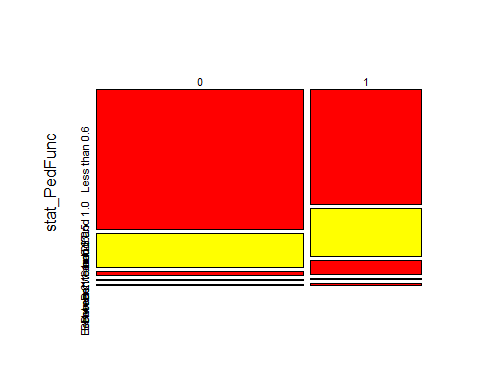
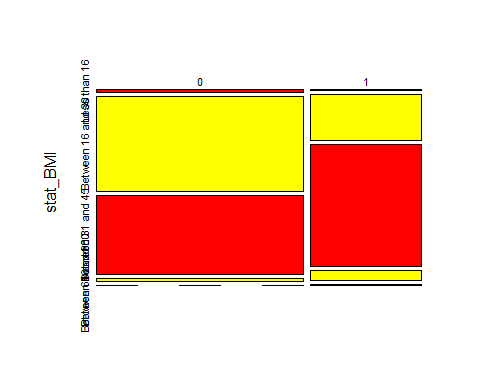
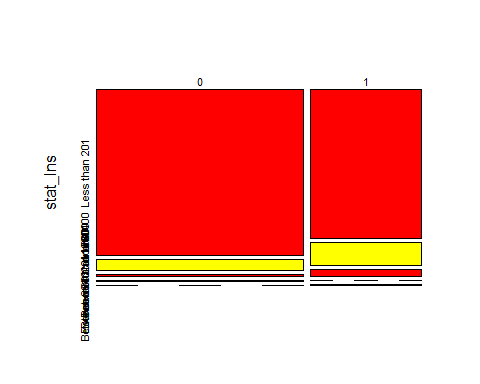
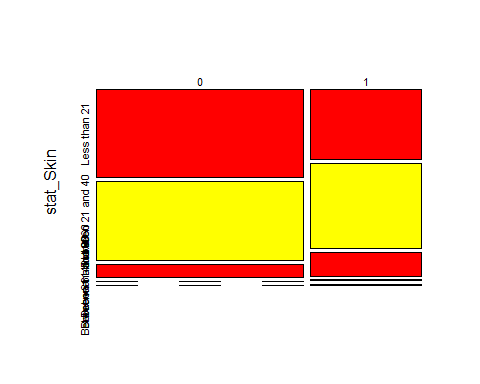
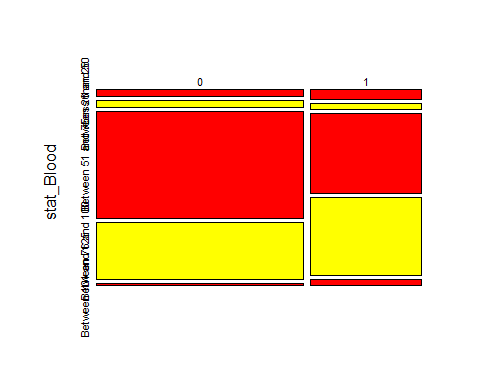
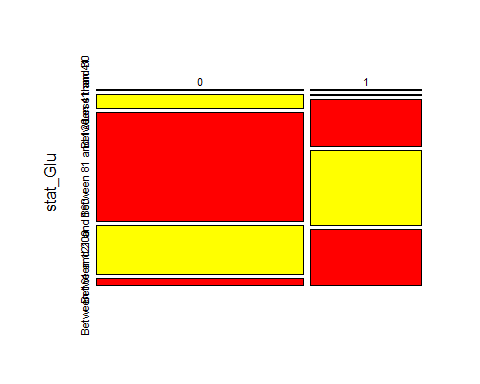
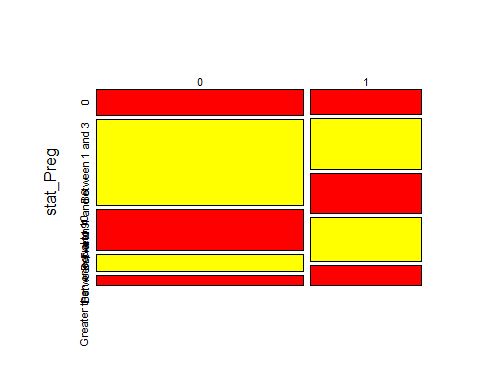
ggplot(data=train.df) +   
 geom\_density(mapping=aes(x=stat\_Age, colour = Outcome)) +  
 ggtitle("Density Plot <Age - Outcome>")



### 2. Result plot for Naive-Bayes classification

Result plot of Naive-Bayes classification

results.nb <- naive\_bayes(Outcome ~., data = train.df) *# for making simple plots*  
plot(results.nb)



## CLUSTERING

### Partitioning Algorithms

### 1. Loading packages

*#install.packages("factoextra")*  
*#install.packages("cluster")*  
*#install.packages("magrittr")*  
  
library("cluster")  
library("factoextra")

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library("magrittr")

### 2. Data Preparation

Handling Null values (missing data).

*#Read csv file*  
diabetes\_data <- read.csv("diabetes(data).csv", header=TRUE)  
  
sapply(diabetes\_data, **function**(x) sum(is.na(x)))

## Pregnancies Glucose BloodPressure   
## 0 0 0   
## SkinThickness Insulin BMI   
## 0 0 0   
## DiabetesPedigreeFunction Age Outcome   
## 0 0 3

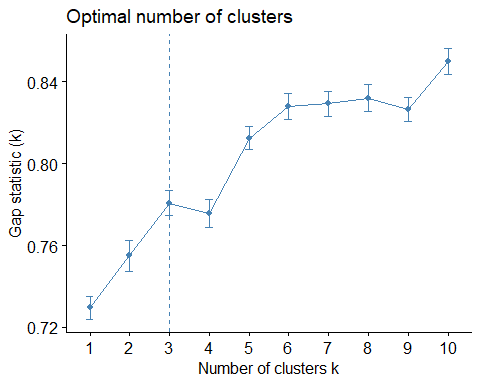
A result without Null values. However, in this dataset is still exist some value of 0.   
Therefore, we will treat them as Null value. By using %>%  
  
  
```r  
diabetes\_data = diabetes\_data %>%  
 na.omit() %>%  
 scale()

### 3.Determine kmeans and compute the cluster

Evaluate the optimal number of cluster

fviz\_nbclust(diabetes\_data, kmeans, method = "gap\_stat")

## Warning: did not converge in 10 iterations



From this result, k will set as 3, nstart will be 100, in which:

Seeds are used to create a starting point for numbers that are created randomly. nstart option that attempts multiple initial configurations and reports on the best one within the kmeans function

set.seed(120)  
km.res <- kmeans(diabetes\_data, 3, nstart = 100)

### 4. Visualize Partitioning Clustering

fviz\_cluster(km.res, data = diabetes\_data,  
 ellipse.type = "convex",  
 palette = "jco",  
 ggtheme = theme\_minimal())

