HW2-R

Nima Kelidari

2023-05-02

I decided to use the second dataset which data is 50-50 for diabetic and non-diabetic persons. this balance will help us to decrease bias in our models. And it is cleaner and has less same row and NA rows so much. (But we try to detect and delete them anyway)

```
#install.packages('caret')
#install.packages('pROC')
#install.packages("ggplot2")
#install.packages("tidyverse")
#install.packages("leaps")
#install.packages("MASS")
#install.packages("tree")
library("tree")
```

```
## Warning: package 'tree' was built under R version 4.2.3
```

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.3

## Warning: package 'ggplot2' was built under R version 4.2.3

## Warning: package 'tibble' was built under R version 4.2.3

## Warning: package 'tidyr' was built under R version 4.2.3

## Warning: package 'readr' was built under R version 4.2.3

## Warning: package 'purrr' was built under R version 4.2.3

## Warning: package 'dplyr' was built under R version 4.2.3

## Warning: package 'stringr' was built under R version 4.2.3

## Warning: package 'forcats' was built under R version 4.2.3

## Warning: package 'lubridate' was built under R version 4.2.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(leaps)
```

```
## Warning: package 'leaps' was built under R version 4.2.3
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(ggplot2)
library(pROC)
```

```
## Warning: package 'pROC' was built under R version 4.2.3
```

```
## Type 'citation("pROC")' for a citation.
##
## Attaching package: 'pROC'
##
## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.2.3
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
## The following object is masked from 'package:purrr':
##
##     lift
```

```
diabets = read.csv("D:/Terme8/Regression/HW2/Data/d2.csv")
```

```
#diabets
```

```
diabets <- diabets[!duplicated(diabets), ]
diabets <- na.omit(diabets)
diabets$output <- as.factor(diabets$Diabetes_binary)

#diabets
```

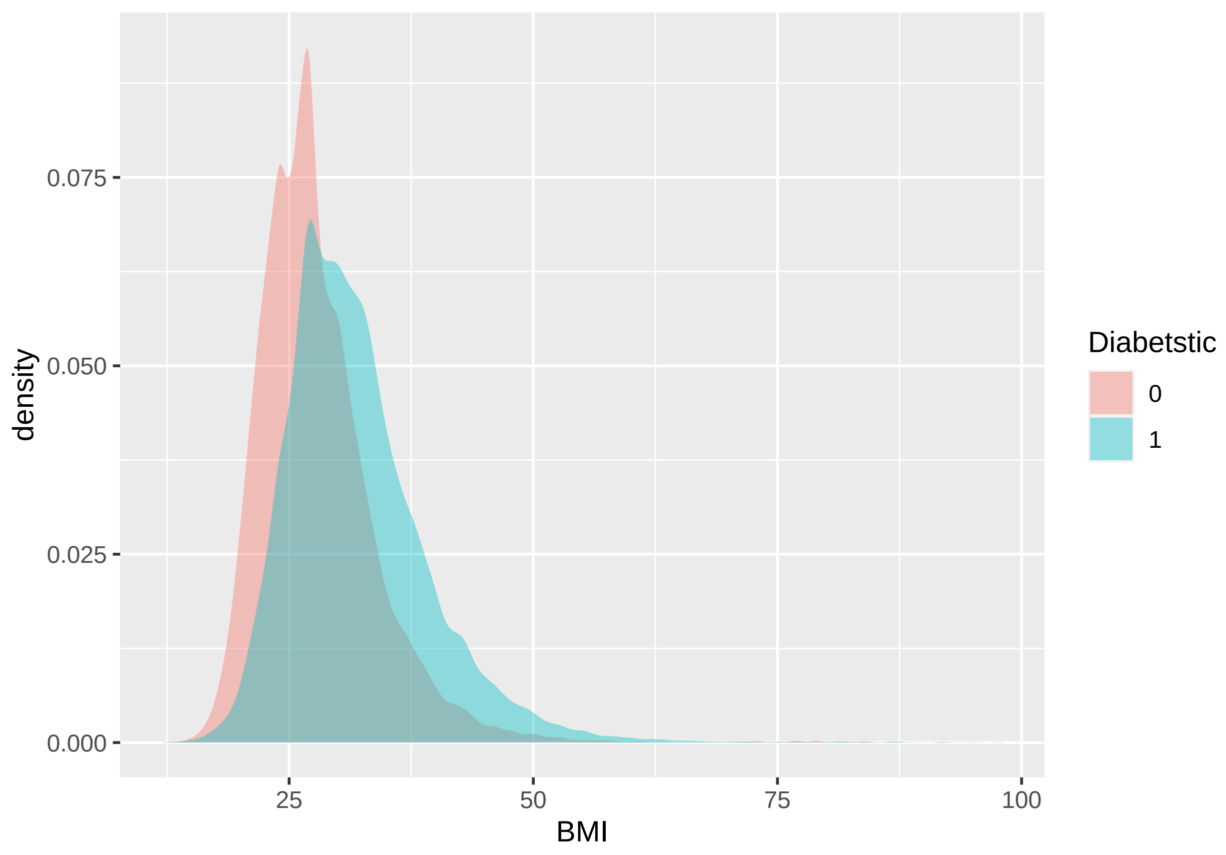
PART 1

Yes, by using the proper features and model, we surely can get a nice prediction and then in the next part, use it for inference. The main reason for this is the data given to us, has some features which probably have some relationship with the probability of whether one has diabetes or not. We can show it by plotting those features by the diabetes factor. For example, we plotted BMI, high blood pressure, high cholesterol, general health, age, education, and income by diabetes factor in the type of histogram density, histogram, and box plot. Here we can see the different distribution of these factors for two groups of diabetic and non-diabetic persons. Moreover, we fit a logistic regression to show t-value is so big for this problem and that these factors and responses are correlated strongly. At last, we fit a model by these factors and see almost all of them, have strongly related to the diabetes factor. So as least some of them can be used for the prediction of whether one has diabetes or not. In the second part, we introduce some of these features as good ones for prediction.

```r
model <- glm(Diabetes_binary ~ BMI, data = diabets)
summary(model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ BMI, data = diabets)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.86763  -0.44918   0.05137   0.47091   0.83051
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0902293  0.0078552  -11.49   <2e-16 ***
## BMI          0.0199781  0.0002551   78.33   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2295464)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 15851  on 69055  degrees of freedom
## AIC: 94352
##
## Number of Fisher Scoring iterations: 2
```

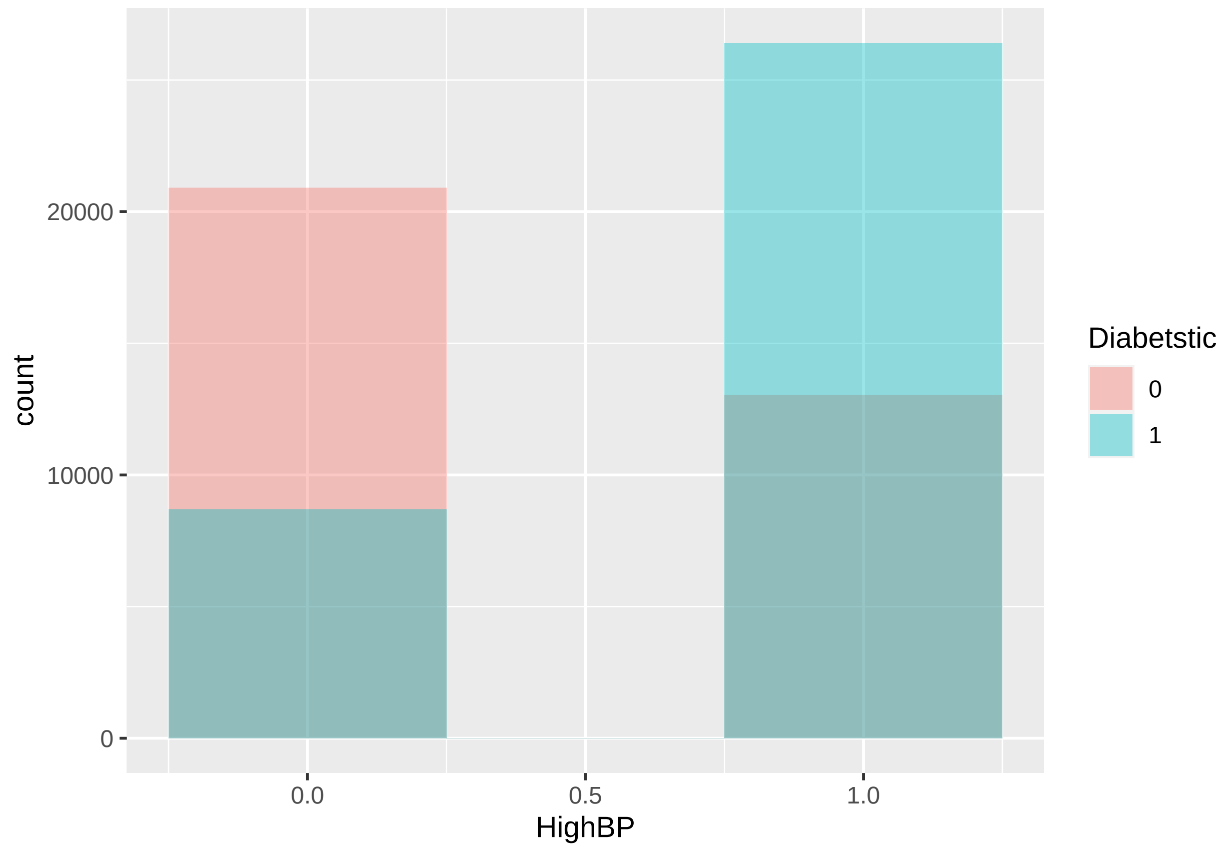
```r
Diabetstic = as.factor(diabets$Diabetes_binary)
# Density areas without lines
ggplot(diabets, aes(x = BMI, fill = Diabetstic , colours = Diabetstic)) +
  geom_density(alpha = 0.4, color = NA)
```



```
model <- glm(Diabetes_binary ~ HighBP, data = diabets)
summary(model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ HighBP, data = diabets)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6694  -0.2935   0.3306   0.3306   0.7065
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.293549   0.002697   108.9   <2e-16 ***
## HighBP      0.375830   0.003568   105.3   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.215343)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 14871  on 69055  degrees of freedom
## AIC: 89941
##
## Number of Fisher Scoring iterations: 2
```

```
Diabetstic = as.factor(diabets$Diabetes_binary)
# Density areas without lines
ggplot(diabets, aes(x = HighBP, fill = Diabetstic , colours = Diabetstic)) +
  geom_histogram(alpha = 0.4, position = "identity",bins = 3)
```



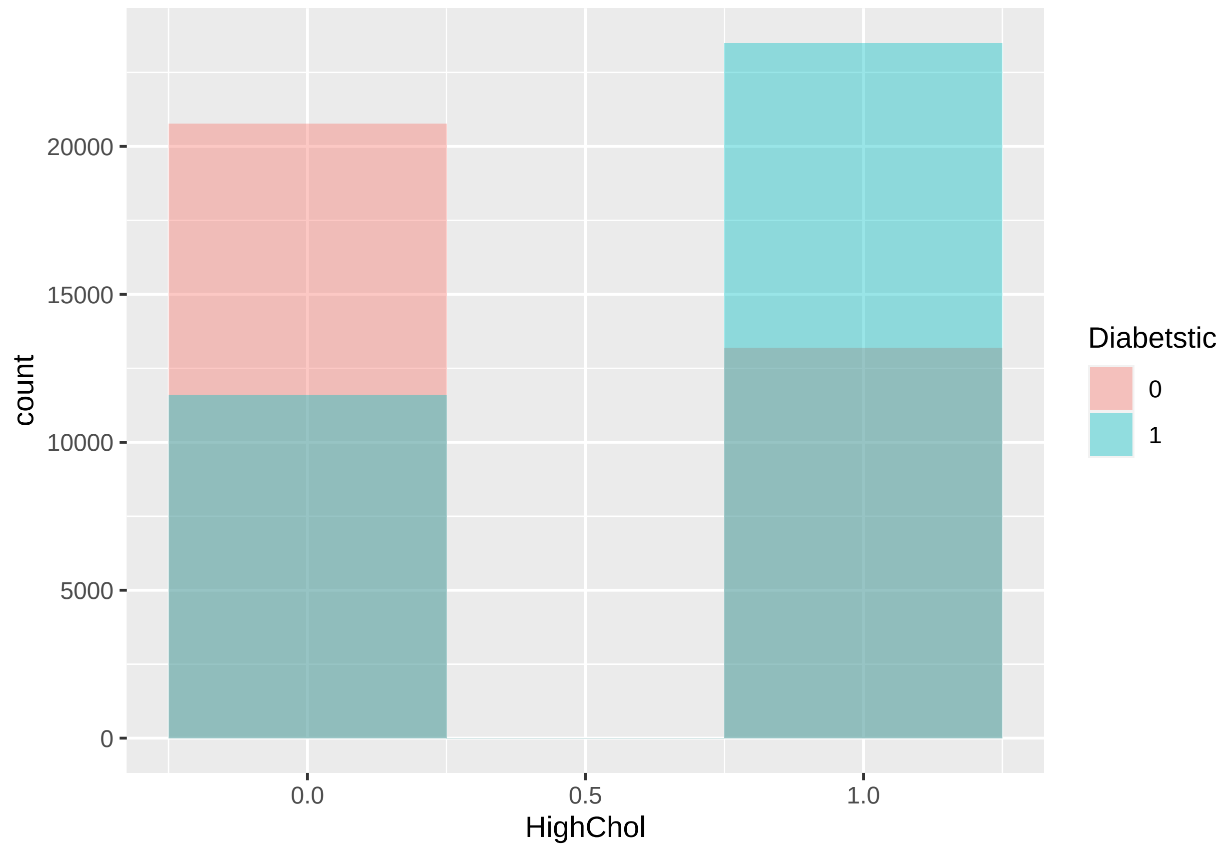
(legend.position = "none")

```
model <- glm(Diabetes_binary ~ HighChol, data = diabets)
summary(model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ HighChol, data = diabets)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.6404  -0.3584   0.3596   0.3596   0.6416
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.358443   0.002667  134.42   <2e-16 ***
## HighChol    0.281915   0.003658   77.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2301479)
##
```

```
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 15893  on 69055  degrees of freedom
## AIC: 94532
##
## Number of Fisher Scoring iterations: 2
```

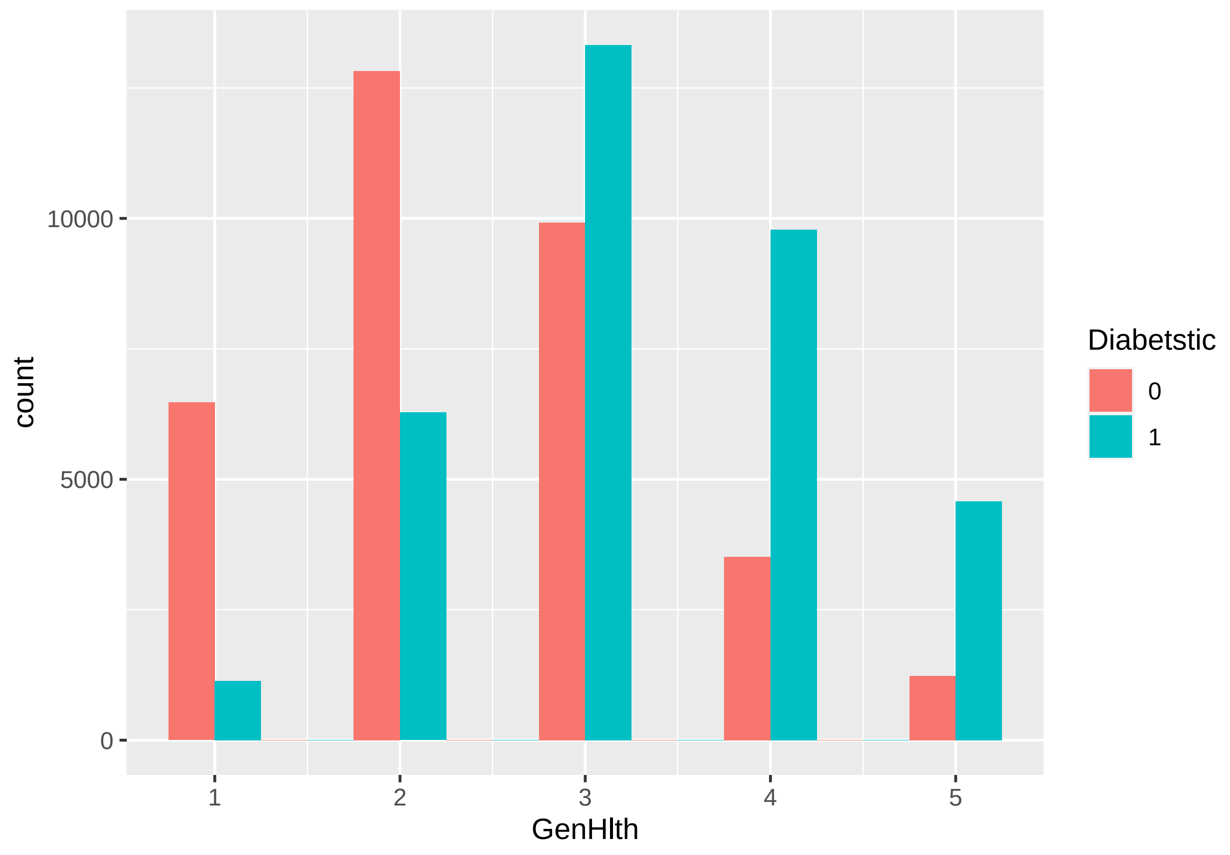
```
Diabetstic = as.factor(diabets$Diabetes_binary)
# Density areas without lines
ggplot(diabets, aes(x = HighChol, fill = Diabetstic , colours = Diabetstic)) +
  geom_histogram(alpha = 0.4, position = "identity",bins = 3)
```



```
model <- glm(Diabetes_binary ~ GenHlth, data = diabets)
summary(model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ GenHlth, data = diabets)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.8905  -0.3537   0.1095   0.4674   0.8253
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.004206   0.004840  -0.869    0.385
## GenHlth      0.178943   0.001576 113.520   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2106319)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 14545  on 69055  degrees of freedom
## AIC: 88413
##
## Number of Fisher Scoring iterations: 2
```

```
Diabetstic = as.factor(diabets$Diabetes_binary)
# Density areas without lines
ggplot(diabets, aes(x = GenHlth, fill = Diabetstic , colours = Diabetstic)) +
  geom_histogram(alpha = 1, position = "dodge",bins = 9)
```

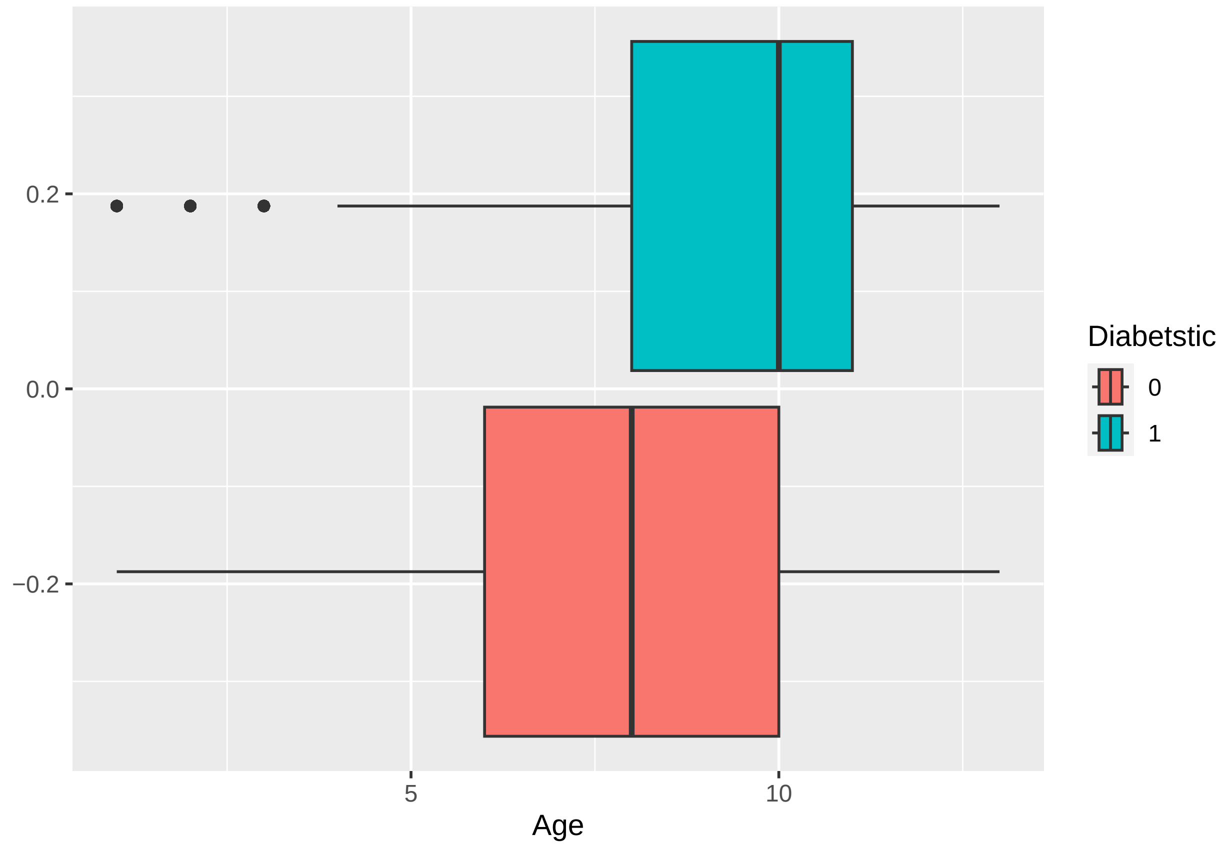


```
model <- glm(Diabetes_binary ~ Age, data = diabets)
summary(model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ Age, data = diabets)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7193  -0.4792   0.2807   0.4247   0.8569
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.095058   0.005803   16.38   <2e-16 ***
## Age         0.048021   0.000640   75.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2310996)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 15959  on 69055  degrees of freedom
## AIC: 94817
##
## Number of Fisher Scoring iterations: 2
```

```
Diabetstic = as.factor(diabets$Diabetes_binary)
# Density areas without lines
ggplot(diabets, aes(x = Age, fill = Diabetstic , colours = Diabetstic)) +
  geom_boxplot(alpha = 1)
```

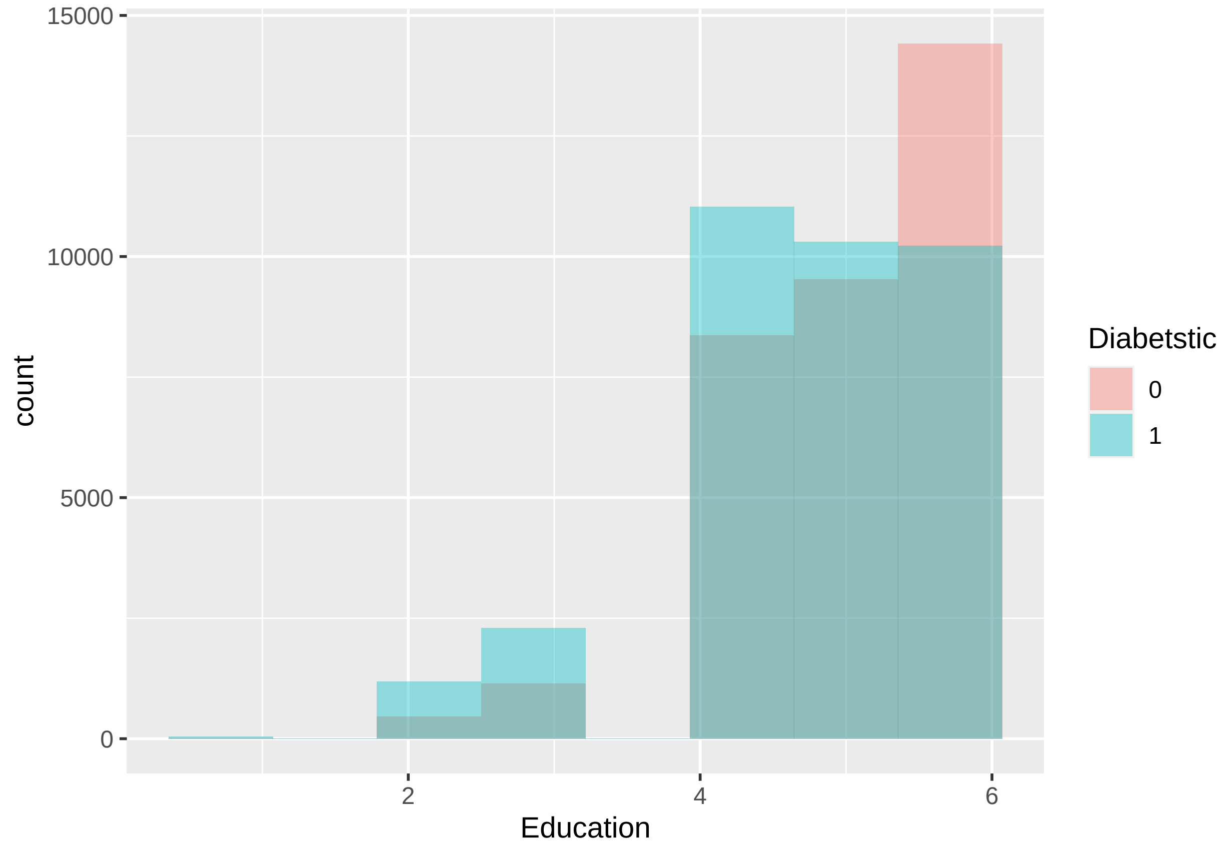


```
model <- glm(Diabetes_binary ~ Education, data = diabets)
summary(model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ Education, data = diabets)
##
## Deviance Residuals:
```

```
##     Min       1Q   Median       3Q      Max
## -0.8085  -0.5006   0.2685   0.4994   0.5764
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.885516   0.009138   96.91   <2e-16 ***
## Education   -0.076992   0.001825  -42.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2436587)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 16826  on 69055  degrees of freedom
## AIC: 98472
##
## Number of Fisher Scoring iterations: 2
```

```
Diabetstic = as.factor(diabets$Diabetes_binary)
# Density areas without lines
ggplot(diabets, aes(x = Education, fill = Diabetstic , colours = Diabetstic)) +
  geom_histogram(alpha = 0.4, position = "identity",bins = 8)
```

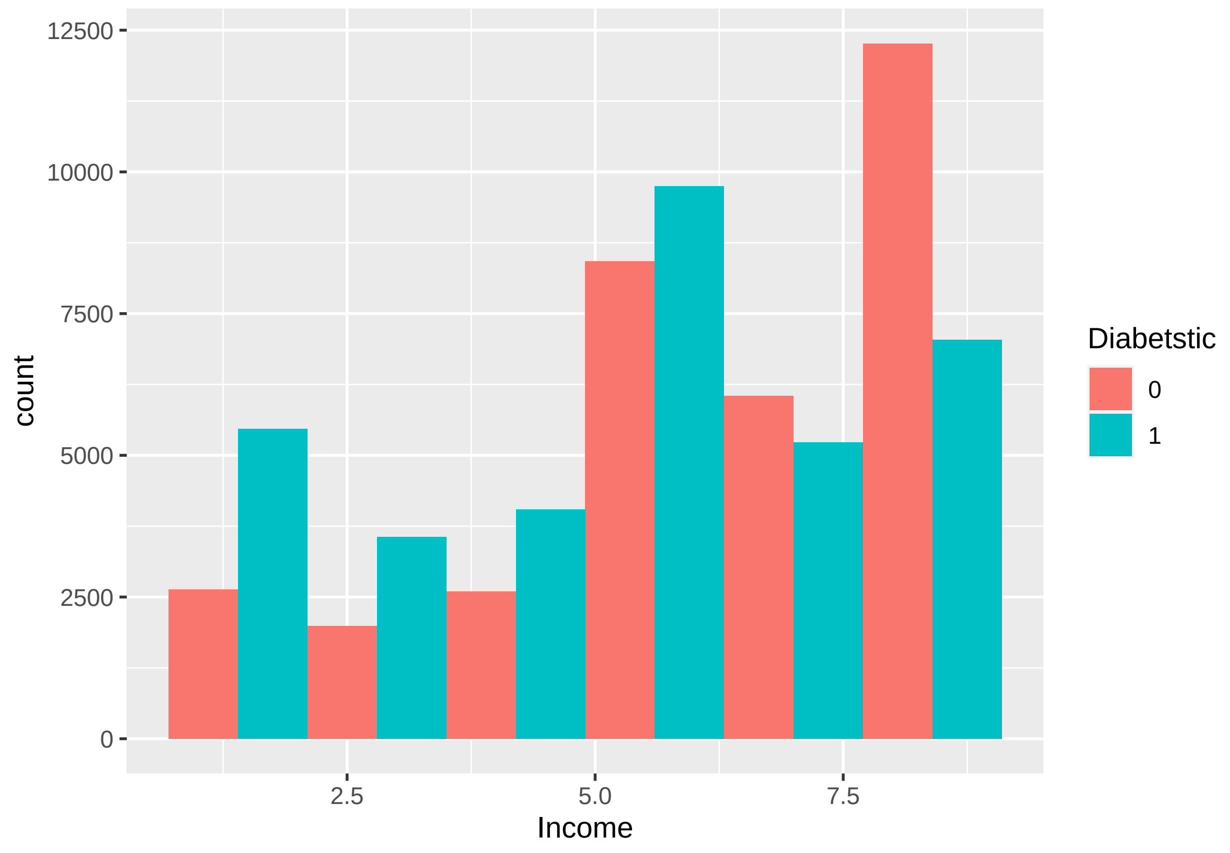


```
model <- glm(Diabetes_binary ~ Income, data = diabets)
summary(model)
```

```
##
```

```
## Call:
## glm(formula = Diabetes_binary ~ Income, data = diabets)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7357  -0.4423   0.2643   0.5088   0.6066
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.7846393  0.0051740  151.65   <2e-16 ***
## Income      -0.0489101  0.0008544  -57.24   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.2386164)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 16478  on 69055  degrees of freedom
## AIC: 97028
##
## Number of Fisher Scoring iterations: 2
```

```
Diabetstic = as.factor(diabets$Diabetes_binary)
# Density areas without lines
ggplot(diabets, aes(x = Income, fill = Diabetstic , colours = Diabetstic)) +
  geom_histogram(alpha = 1, position = "dodge",bins = 6)
```



```
model <- glm(Diabetes_binary ~ Income+Education+Age+GenHlth+HighChol+HighBP+BMI, data = diabets)
summary(model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ Income + Education + Age + GenHlth +
##     HighChol + HighBP + BMI, data = diabets)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.62192  -0.34359   0.03701   0.34423   1.16969
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.4658267  0.0134692  -34.59  < 2e-16 ***
## Income      -0.0091642  0.0008643  -10.60  < 2e-16 ***
## Education   -0.0077290  0.0017604   -4.39 1.13e-05 ***
## Age          0.0280524  0.0006106   45.94  < 2e-16 ***
## GenHlth      0.1050222  0.0016701   62.88  < 2e-16 ***
## HighChol     0.1139753  0.0034549   32.99  < 2e-16 ***
## HighBP       0.1633319  0.0037298   43.79  < 2e-16 ***
## BMI          0.0122764  0.0002384   51.51  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1775591)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 12260  on 69049  degrees of freedom
## AIC: 76624
##
## Number of Fisher Scoring iterations: 2
```

```
Diabetstic = as.factor(diabets$Diabetes_binary)
```

PART 2

We can answer this question by feat a model like logistic regression on whole data and all of the features, then detect those features which have big enough t-values. Here we can see BMI, high blood pressure, high cholesterol, general health, age, cholesterol check, history of heart attack, sex, age, heavy alcohol consumption, and income have an absolute value of t-value more than 10; We can detect these factors as important factors compared to others and check them more carefully. we will see how we can select features more reliably by feature selection in part 3.

```
md  <- glm(Diabetes_binary ~., data = diabets)
summary(md)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ ., data = diabets)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.554e-14  -4.219e-15  -4.892e-16   4.161e-15   1.920e-14
##
## Coefficients:
```

```
##                          Estimate Std. Error    t value Pr(>|t|)
## (Intercept)             7.367e-15  2.229e-16  3.305e+01  < 2e-16 ***
## HighBP                 -1.949e-15  4.689e-17 -4.157e+01  < 2e-16 ***
## HighChol               -1.342e-15  4.323e-17 -3.104e+01  < 2e-16 ***
## CholCheck              -2.073e-15  1.287e-16 -1.611e+01  < 2e-16 ***
## BMI                    -1.464e-16  3.050e-18 -4.799e+01  < 2e-16 ***
## Smoker                  4.170e-17  4.110e-17  1.015e+00 0.310308
## Stroke                 -3.051e-16  8.477e-17 -3.598e+00 0.000320 ***
## HeartDiseaseorAttack -6.008e-16  6.042e-17 -9.944e+00  < 2e-16 ***
## PhysActivity            7.286e-17  4.634e-17  1.572e+00 0.115872
## Fruits                  4.770e-17  4.241e-17  1.125e+00 0.260690
## Veggies                 1.339e-16  5.043e-17  2.654e+00 0.007945 **
## HvyAlcoholConsump       1.548e-15  9.834e-17  1.574e+01  < 2e-16 ***
## AnyHealthcare          -1.709e-16  9.879e-17 -1.730e+00 0.083687 .
## NoDocbcCost             1.806e-17  7.173e-17  2.520e-01 0.801257
## GenHlth                -1.294e-15  2.483e-17 -5.212e+01  < 2e-16 ***
## MentHlth                1.096e-17  2.702e-18  4.055e+00 5.02e-05 ***
## PhysHlth                1.940e-17  2.540e-18  7.639e+00 2.22e-14 ***
## DiffWalk               -3.559e-16  5.687e-17 -6.257e+00 3.95e-10 ***
## Sex                    -5.652e-16  4.153e-17 -1.361e+01  < 2e-16 ***
## Age                    -3.069e-16  8.088e-18 -3.794e+01  < 2e-16 ***
## Education               7.614e-17  2.203e-17  3.456e+00 0.000548 ***
## Income                  1.340e-16  1.122e-17  1.194e+01  < 2e-16 ***
## output1                 1.000e+00  4.733e-17  2.113e+16  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 2.706099e-29)
##
##     Null deviance: 1.7260e+04  on 69056  degrees of freedom
## Residual deviance: 1.8681e-24  on 69034  degrees of freedom
## AIC: -4346532
##
## Number of Fisher Scoring iterations: 1
```

PART 3

Yes, absolutely we can do it by feature selection methods. As best subset selection can be very time-consuming and computationally heavy in practice (in real cases), we just use two methods forward and backward selection by AIC. In each one, we want to use 5 features maximum utmost. so we make a full model and null model first and try to extract the five most important features. By each of these two methods, we can see five features below BMI, high blood pressure, high cholesterol, general health, and age, have the most importance in prediction as we guss before in part 2; in other words, these have most dependency to pred factor. On another hand, these do have not a significant relation to each other, because, in a model that we created before, they didn't decrease each other t-value strongly. so we use these five features for use in the next parts.

```r
full.model <- glm(Diabetes_binary ~.-output, data = diabets)
step.model <- stepAIC(full.model, direction = "both", trace = FALSE)
summary(step.model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ HighBP + HighChol + CholCheck +
##     BMI + Stroke + HeartDiseaseorAttack + PhysActivity + Veggies +
##     HvyAlcoholConsump + AnyHealthcare + GenHlth + MentHlth +
##     PhysHlth + DiffWalk + Sex + Age + Education + Income, data = diabets)
```

```
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.55229  -0.33889   0.03907   0.33968   1.25189
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.5923088  0.0173947 -34.051  < 2e-16 ***
## HighBP                0.1540678  0.0037242  41.369  < 2e-16 ***
## HighChol              0.1088292  0.0034467  31.575  < 2e-16 ***
## CholCheck             0.1667005  0.0103164  16.159  < 2e-16 ***
## BMI                   0.0118230  0.0002408  49.091  < 2e-16 ***
## Stroke                0.0252910  0.0068154   3.711 0.000207 ***
## HeartDiseaseorAttack  0.0482946  0.0048483   9.961  < 2e-16 ***
## PhysActivity         -0.0063787  0.0037116  -1.719 0.085698 .
## Veggies              -0.0118939  0.0039675  -2.998 0.002720 **
## HvyAlcoholConsump    -0.1252300  0.0078558 -15.941  < 2e-16 ***
## AnyHealthcare         0.0139394  0.0078103   1.785 0.074306 .
## GenHlth               0.1037024  0.0019532  53.093  < 2e-16 ***
## MentHlth             -0.0008754  0.0002161  -4.051 5.11e-05 ***
## PhysHlth             -0.0015695  0.0002039  -7.697 1.41e-14 ***
## DiffWalk              0.0286401  0.0045682   6.269 3.65e-10 ***
## Sex                   0.0455012  0.0032980  13.797  < 2e-16 ***
## Age                   0.0247627  0.0006350  38.996  < 2e-16 ***
## Education            -0.0060606  0.0017639  -3.436 0.000591 ***
## Income               -0.0107495  0.0008965 -11.990  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1749733)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 12080  on 69038  degrees of freedom
## AIC: 75622
##
## Number of Fisher Scoring iterations: 2
```

```
train.control <- trainControl(method = "cv", number = 10)
step.model <- train(Diabetes_binary ~.-output, data = diabets,
                    method = "leapBackward",
                    tuneGrid = data.frame(nvmax = 1:5),
                    trControl = train.control
                    )
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
step.model$results
```

```
##   nvmax      RMSE  Rsquared       MAE      RMSESD  RsquaredSD       MAESD
## 1     1 0.4589429 0.1573782 0.4212581 0.002282365 0.008347025 0.001917050
## 2     2 0.4451980 0.2072301 0.3970476 0.002754008 0.009545242 0.003708062
## 3     3 0.4338239 0.2471322 0.3818945 0.003863398 0.013234345 0.002302323
## 4     4 0.4252467 0.2765826 0.3678374 0.003860724 0.013067806 0.003697475
## 5     5 0.4219822 0.2876314 0.3631322 0.003694688 0.012444030 0.003502273
```

```
step.model$bestTune
```

```
##   nvmax
## 5     5
```

```
summary(step.model$finalModel)
```

```
## Subset selection object
## 21 Variables  (and intercept)
##                      Forced in Forced out
## HighBP                   FALSE      FALSE
## HighChol                 FALSE      FALSE
## CholCheck                FALSE      FALSE
## BMI                      FALSE      FALSE
## Smoker                   FALSE      FALSE
## Stroke                   FALSE      FALSE
## HeartDiseaseorAttack     FALSE      FALSE
## PhysActivity             FALSE      FALSE
## Fruits                   FALSE      FALSE
## Veggies                  FALSE      FALSE
## HvyAlcoholConsump        FALSE      FALSE
## AnyHealthcare            FALSE      FALSE
## NoDocbcCost              FALSE      FALSE
## GenHlth                  FALSE      FALSE
## MentHlth                 FALSE      FALSE
## PhysHlth                 FALSE      FALSE
## DiffWalk                 FALSE      FALSE
## Sex                      FALSE      FALSE
## Age                      FALSE      FALSE
## Education                FALSE      FALSE
## Income                   FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: backward
##          HighBP HighChol CholCheck BMI Smoker Stroke HeartDiseaseorAttack
## 1  ( 1 ) " "    " "      " "       " " " "    " "    " "
## 2  ( 1 ) " "    " "      " "       " " " "    " "    " "
## 3  ( 1 ) " "    " "      " "       "*" " "    " "    " "
## 4  ( 1 ) "*"    " "      " "       "*" " "    " "    " "
## 5  ( 1 ) "*"    "*"      " "       "*" " "    " "    " "
##          PhysActivity Fruits Veggies HvyAlcoholConsump AnyHealthcare
## 1  ( 1 ) " "          " "    " "     " "               " "
## 2  ( 1 ) " "          " "    " "     " "               " "
## 3  ( 1 ) " "          " "    " "     " "               " "
## 4  ( 1 ) " "          " "    " "     " "               " "
## 5  ( 1 ) " "          " "    " "     " "               " "
##          NoDocbcCost GenHlth MentHlth PhysHlth DiffWalk Sex Age Education
## 1  ( 1 ) " "         "*"     " "      " "      " "      " " " " " "
## 2  ( 1 ) " "         "*"     " "      " "      " "      " " "*" " "
## 3  ( 1 ) " "         "*"     " "      " "      " "      " " "*" " "
## 4  ( 1 ) " "         "*"     " "      " "      " "      " " "*" " "
## 5  ( 1 ) " "         "*"     " "      " "      " "      " " "*" " "
##          Income
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
```

```
coef(step.model$finalModel, 5)
```

```
## (Intercept)      HighBP    HighChol         BMI     GenHlth         Age
## -0.58663321  0.16583644  0.11341426  0.01235041  0.11299873  0.02862784
```

```
mod <- glm(Diabetes_binary ~ HighBP + HighChol + BMI + GenHlth + Age,
   data = diabets)
summary(mod)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ HighBP + HighChol + BMI + GenHlth +
##     Age, data = diabets)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.60267  -0.34499   0.04069   0.34305   1.16937
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.5866332  0.0090232  -65.01   <2e-16 ***
## HighBP       0.1658364  0.0037307   44.45   <2e-16 ***
## HighChol     0.1134143  0.0034596   32.78   <2e-16 ***
## BMI          0.0123504  0.0002386   51.76   <2e-16 ***
## GenHlth      0.1129987  0.0015729   71.84   <2e-16 ***
## Age          0.0286278  0.0006100   46.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1780615)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 12295  on 69051  degrees of freedom
## AIC: 76817
##
## Number of Fisher Scoring iterations: 2
```

```
full.model <- glm(Diabetes_binary ~.-output, data = diabets)
step.model <- stepAIC(full.model, direction = "both", trace = FALSE)
summary(step.model)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ HighBP + HighChol + CholCheck +
##     BMI + Stroke + HeartDiseaseorAttack + PhysActivity + Veggies +
##     HvyAlcoholConsump + AnyHealthcare + GenHlth + MentHlth +
##     PhysHlth + DiffWalk + Sex + Age + Education + Income, data = diabets)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.55229  -0.33889   0.03907   0.33968   1.25189
```

```
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -0.5923088  0.0173947 -34.051  < 2e-16 ***
## HighBP                0.1540678  0.0037242  41.369  < 2e-16 ***
## HighChol              0.1088292  0.0034467  31.575  < 2e-16 ***
## CholCheck             0.1667005  0.0103164  16.159  < 2e-16 ***
## BMI                   0.0118230  0.0002408  49.091  < 2e-16 ***
## Stroke                0.0252910  0.0068154   3.711 0.000207 ***
## HeartDiseaseorAttack  0.0482946  0.0048483   9.961  < 2e-16 ***
## PhysActivity         -0.0063787  0.0037116  -1.719 0.085698 .
## Veggies              -0.0118939  0.0039675  -2.998 0.002720 **
## HvyAlcoholConsump    -0.1252300  0.0078558 -15.941  < 2e-16 ***
## AnyHealthcare         0.0139394  0.0078103   1.785 0.074306 .
## GenHlth               0.1037024  0.0019532  53.093  < 2e-16 ***
## MentHlth             -0.0008754  0.0002161  -4.051 5.11e-05 ***
## PhysHlth             -0.0015695  0.0002039  -7.697 1.41e-14 ***
## DiffWalk              0.0286401  0.0045682   6.269 3.65e-10 ***
## Sex                   0.0455012  0.0032980  13.797  < 2e-16 ***
## Age                   0.0247627  0.0006350  38.996  < 2e-16 ***
## Education            -0.0060606  0.0017639  -3.436 0.000591 ***
## Income               -0.0107495  0.0008965 -11.990  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1749733)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 12080  on 69038  degrees of freedom
## AIC: 75622
##
## Number of Fisher Scoring iterations: 2
```

```
set.seed(123)
train.control <- trainControl(method = "cv", number = 10)
step.model <- train(Diabetes_binary ~.-output, data = diabets,
                    method = "leapForward",
                    tuneGrid = data.frame(nvmax = 1:5),
                    trControl = train.control
                    )
```

```
## Warning in train.default(x, y, weights = w, ...): You are trying to do
## regression and your outcome only has two possible values Are you trying to do
## classification? If so, use a 2 level factor as your outcome column.
```

```
step.model$results
```

```
##   nvmax      RMSE  Rsquared       MAE      RMSESD  RsquaredSD       MAESD
## 1     1 0.4589449 0.1573952 0.4212585 0.002299933 0.008421811 0.001758425
## 2     2 0.4397916 0.2262143 0.3868186 0.002309520 0.008198404 0.001832603
## 3     3 0.4334785 0.2482743 0.3780406 0.002330834 0.008151682 0.001870903
## 4     4 0.4252604 0.2765075 0.3678396 0.002230627 0.007686012 0.002104354
## 5     5 0.4219997 0.2875429 0.3631400 0.002065983 0.007077088 0.002079634
```

```
step.model$bestTune
```

```
##   nvmax
## 5     5
```

```
summary(step.model$finalModel)
```

```
## Subset selection object
## 21 Variables  (and intercept)
##                      Forced in Forced out
## HighBP                   FALSE      FALSE
## HighChol                 FALSE      FALSE
## CholCheck                FALSE      FALSE
## BMI                      FALSE      FALSE
## Smoker                   FALSE      FALSE
## Stroke                   FALSE      FALSE
## HeartDiseaseorAttack     FALSE      FALSE
## PhysActivity             FALSE      FALSE
## Fruits                   FALSE      FALSE
## Veggies                  FALSE      FALSE
## HvyAlcoholConsump        FALSE      FALSE
## AnyHealthcare            FALSE      FALSE
## NoDocbcCost              FALSE      FALSE
## GenHlth                  FALSE      FALSE
## MentHlth                 FALSE      FALSE
## PhysHlth                 FALSE      FALSE
## DiffWalk                 FALSE      FALSE
## Sex                      FALSE      FALSE
## Age                      FALSE      FALSE
## Education                FALSE      FALSE
## Income                   FALSE      FALSE
## 1 subsets of each size up to 5
## Selection Algorithm: forward
##          HighBP HighChol CholCheck BMI Smoker Stroke HeartDiseaseorAttack
## 1  ( 1 ) " "    " "      " "       " " " "    " "    " "
## 2  ( 1 ) "*"    " "      " "       " " " "    " "    " "
## 3  ( 1 ) "*"    " "      " "       "*" " "    " "    " "
## 4  ( 1 ) "*"    " "      " "       "*" " "    " "    " "
## 5  ( 1 ) "*"    "*"      " "       "*" " "    " "    " "
##          PhysActivity Fruits Veggies HvyAlcoholConsump AnyHealthcare
## 1  ( 1 ) " "          " "    " "     " "               " "
## 2  ( 1 ) " "          " "    " "     " "               " "
## 3  ( 1 ) " "          " "    " "     " "               " "
## 4  ( 1 ) " "          " "    " "     " "               " "
## 5  ( 1 ) " "          " "    " "     " "               " "
##          NoDocbcCost GenHlth MentHlth PhysHlth DiffWalk Sex Age Education
## 1  ( 1 ) " "         "*"     " "      " "      " "      " " " " " "
## 2  ( 1 ) " "         "*"     " "      " "      " "      " " " " " "
## 3  ( 1 ) " "         "*"     " "      " "      " "      " " " " " "
## 4  ( 1 ) " "         "*"     " "      " "      " "      " " "*" " "
## 5  ( 1 ) " "         "*"     " "      " "      " "      " " "*" " "
##          Income
## 1  ( 1 ) " "
## 2  ( 1 ) " "
## 3  ( 1 ) " "
## 4  ( 1 ) " "
## 5  ( 1 ) " "
```

```
coef(step.model$finalModel, 5)
```

```
## (Intercept)      HighBP    HighChol         BMI     GenHlth         Age
## -0.58663321  0.16583644  0.11341426  0.01235041  0.11299873  0.02862784
```

```
mod <- glm(Diabetes_binary ~ HighBP + HighChol + BMI + GenHlth + Age,
   data = diabets)
summary(mod)
```

```
##
## Call:
## glm(formula = Diabetes_binary ~ HighBP + HighChol + BMI + GenHlth +
##     Age, data = diabets)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.60267  -0.34499   0.04069   0.34305   1.16937
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.5866332  0.0090232  -65.01   <2e-16 ***
## HighBP       0.1658364  0.0037307   44.45   <2e-16 ***
## HighChol     0.1134143  0.0034596   32.78   <2e-16 ***
## BMI          0.0123504  0.0002386   51.76   <2e-16 ***
## GenHlth      0.1129987  0.0015729   71.84   <2e-16 ***
## Age          0.0286278  0.0006100   46.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1780615)
##
##     Null deviance: 17260  on 69056  degrees of freedom
## Residual deviance: 12295  on 69051  degrees of freedom
## AIC: 76817
##
## Number of Fisher Scoring iterations: 2
```

PART 4

We will use the five features that are mentioned in the last part. If we use all of the features or more than five, our efficiency for the trained model will drop. in this part, we use 70 percent of the data for the training and 30 of the data for the test. here we will not use a validation set and we prefer to use cross-validation, more specifically 5fold-cv for it,s time-consuming. then we will try to train the model, tune its hyperparameters, and then test it and get accuracy and precision, and recall from them, then make the confusion matrix for each model based on the test set. We here will use QDA, LDA, Randomforrest, Neural Network, Tree, Linear SVM, Logistic regression, and KNN models. after the test, we will save the accuracy of each model and show and compare them at last and choose one of them. (Here we can see the Tree model had excellent performance besides its simplicity)

```
diabets$Diabetes_binary <- as.factor(diabets$Diabetes_binary)
preProcess <- c("center","scale")
i <- createDataPartition(y = diabets$Diabetes_binary, times = 1, p = 0.7, list = FALSE)
training_set <- diabets[i,]
test_set <- diabets[-i,]
trControl <- trainControl(method = "cv",number = 5)
```

```
model <- train(Diabetes_binary ~ Age+GenHlth+HighChol+HighBP+BMI, method='qda', data = training_set, me
summary(model)
```

```
##             Length Class      Mode
## prior        2     -none-     numeric
## counts       2     -none-     numeric
## means       10     -none-     numeric
## scaling     50     -none-     numeric
## ldet         2     -none-     numeric
## lev          2     -none-     character
## N            1     -none-     numeric
## call         3     -none-     call
## xNames       5     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
## param        0     -none-     list
```

```
test_set$pred <- predict(model, test_set)
test_set$factor_pred <- as.factor(test_set$pred)
test_set$factor_truth <- as.factor(test_set$Diabetes_binary)
precision <- posPredValue(test_set$factor_truth, test_set$factor_pred)
recall <- sensitivity(test_set$factor_truth, test_set$factor_pred)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table
test_set$pred <- predict(model, test_set,probability=TRUE)
#________________________________________________
print(paste("Precision: ",precision))
```

```
## [1] "Precision:  0.688751472320377"
```

```
print(paste("Accuracy: ",accuracy))
```

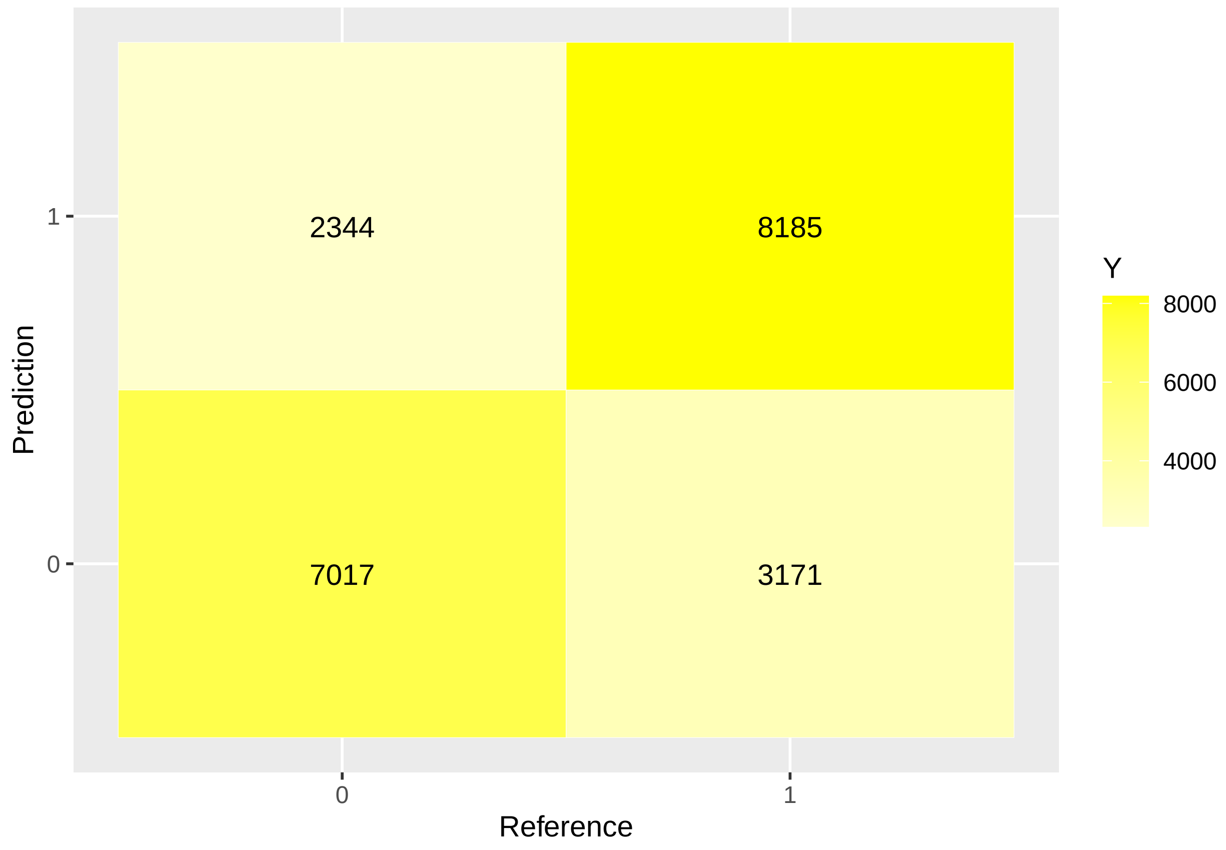
```
## [1] "Accuracy:  0.733793502920307"
```

```
print(paste("Recall: ",recall))
```

```
## [1] "Recall:  0.749599401773315"
```

```
acc_qda <- accuracy
#________________________________________________
Reference <- factor(c(0, 1, 0, 1))
Prediction   <- factor(c(0, 0, 1, 1))
Y  <- array(cmat)
df <- data.frame(Prediction, Reference, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow")
```



```
model <- train(Diabetes_binary ~ Age+GenHlth+HighChol+HighBP+BMI, method='rf', data = training_set, met
summary(model)
```

```
##                 Length Class      Mode
## call                4  -none-     call
## type                1  -none-     character
## predicted       48340  factor     numeric
## err.rate         1500  -none-     numeric
## confusion           6  -none-     numeric
## votes           96680  matrix     numeric
## oob.times       48340  -none-     numeric
## classes             2  -none-     character
## importance          5  -none-     numeric
## importanceSD        0  -none-     NULL
## localImportance     0  -none-     NULL
## proximity           0  -none-     NULL
## ntree               1  -none-     numeric
## mtry                1  -none-     numeric
## forest             14  -none-     list
## y               48340  factor     numeric
## test                0  -none-     NULL
## inbag               0  -none-     NULL
## xNames              5  -none-     character
## problemType         1  -none-     character
## tuneValue           1  data.frame list
## obsLevels           2  -none-     character
## param                0  -none-     list
```

```r
test_set$pred <- predict(model, test_set)
test_set$factor_pred <- as.factor(test_set$pred)
test_set$factor_truth <- as.factor(test_set$Diabetes_binary)
precision <- posPredValue(test_set$factor_truth, test_set$factor_pred)
recall <- sensitivity(test_set$factor_truth, test_set$factor_pred)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table
test_set$pred <- predict(model, test_set,probability=TRUE)
#_________________________________________________
print(paste("Precision: ",precision))
```

```
## [1] "Precision:  0.692775814683942"
```

```r
print(paste("Accuracy: ",accuracy))
```

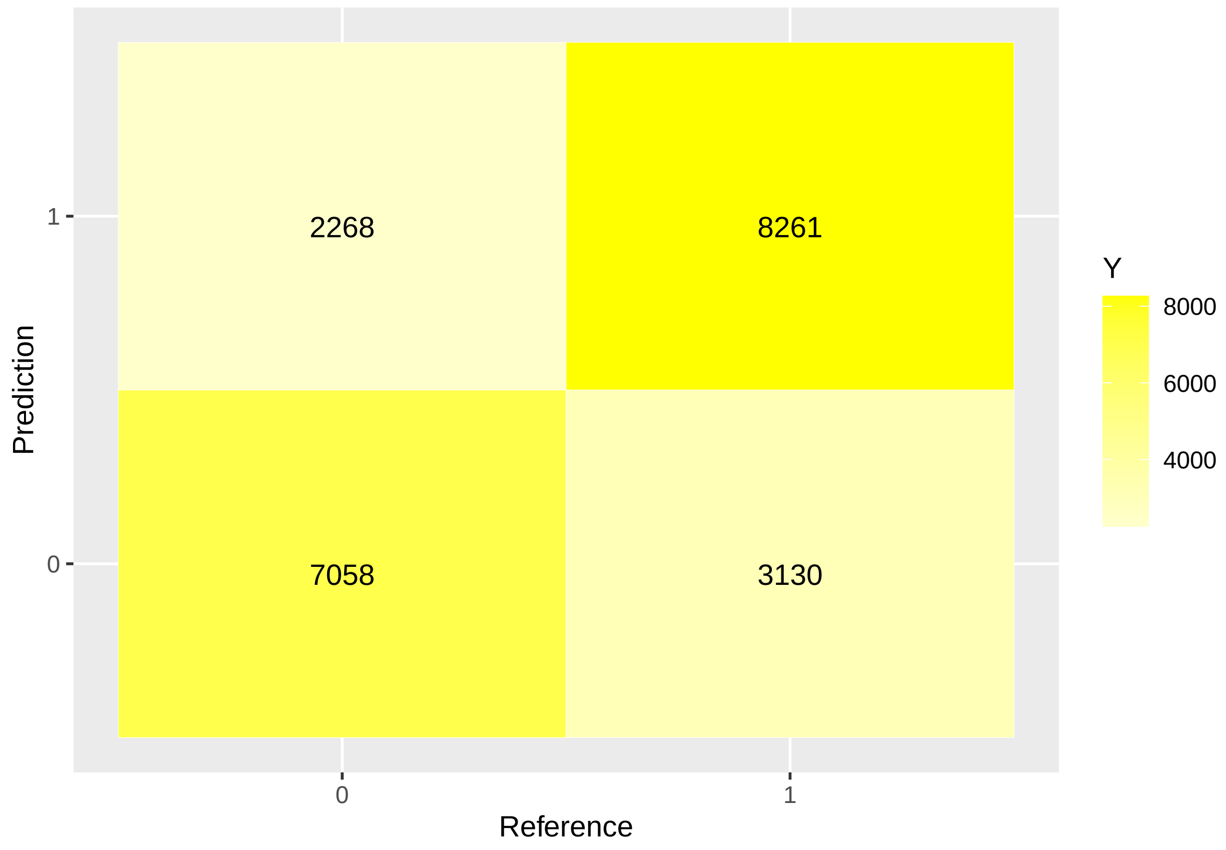
```
## [1] "Accuracy:  0.739441038760438"
```

```r
print(paste("Recall: ",recall))
```

```
## [1] "Recall:  0.756808921295303"
```

```r
acc_rf <- accuracy
#_________________________________________________
Reference <- factor(c(0, 1, 0, 1))
Prediction   <- factor(c(0, 0, 1, 1))
Y  <- array(cmat)
df <- data.frame(Prediction, Reference, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow")
```



1

```
model <- train(Diabetes_binary ~ Age+GenHlth+HighChol+HighBP+BMI, method='lda', data = training_set, me
summary(model)
```

```
##             Length Class      Mode
## prior        2     -none-     numeric
## counts       2     -none-     numeric
## means       10     -none-     numeric
## scaling      5     -none-     numeric
## lev          2     -none-     character
## svd          1     -none-     numeric
## N            1     -none-     numeric
## call         4     -none-     call
## xNames       5     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
## param        1     -none-     list
```

```
test_set$pred <- predict(model, test_set)
test_set$factor_pred <- as.factor(test_set$pred)
test_set$factor_truth <- as.factor(test_set$Diabetes_binary)
precision <- posPredValue(test_set$factor_truth, test_set$factor_pred)
recall <- sensitivity(test_set$factor_truth, test_set$factor_pred)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table
```

```r
test_set$pred <- predict(model, test_set,probability=TRUE)
#_______________________________________________
print(paste("Precision: ",precision))
```

```
## [1] "Precision:  0.703572830781311"
```

```r
print(paste("Accuracy: ",accuracy))
```

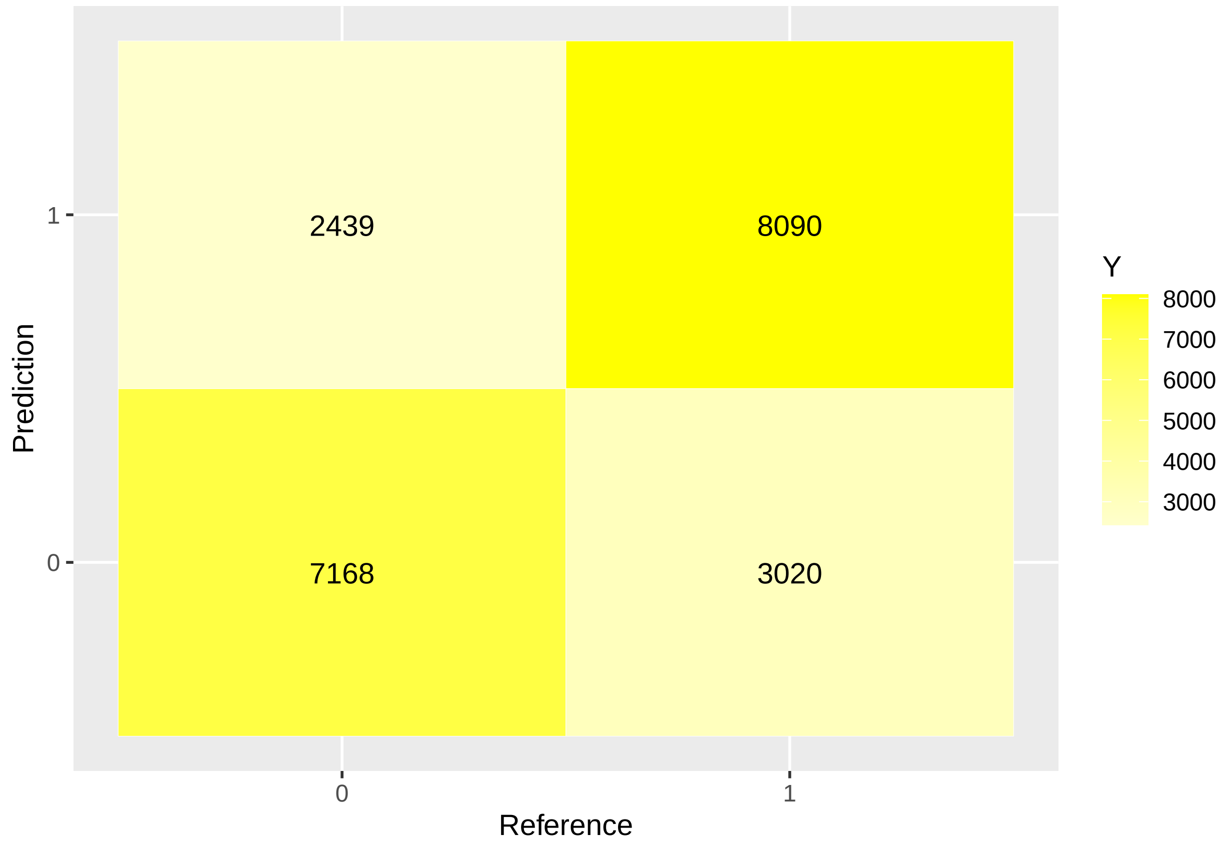
```
## [1] "Accuracy:  0.736496596997635"
```

```r
print(paste("Recall: ",recall))
```

```
## [1] "Recall:  0.746122618923701"
```

```r
acc_lda <- accuracy
#_______________________________________________
Reference <- factor(c(0, 1, 0, 1))
Prediction   <- factor(c(0, 0, 1, 1))
Y  <- array(cmat)
df <- data.frame(Prediction, Reference, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow")
```



```r
model <- train(Diabetes_binary ~ Age+GenHlth+HighChol+HighBP+BMI, method='nnet', data = training_set, m
```

```
## # weights:  8
## initial  value 27475.631598
## iter  10 value 21038.517775
## iter  20 value 20235.862096
## iter  30 value 20198.696705
## iter  40 value 20184.071335
## iter  50 value 20182.863189
## final  value 20182.654174
## converged
## # weights:  22
## initial  value 27256.022350
## iter  10 value 20413.645882
## iter  20 value 20240.068026
## iter  30 value 20068.923616
## iter  40 value 20036.505684
## iter  50 value 20026.392762
## iter  60 value 20023.193424
## iter  70 value 20019.805360
## iter  80 value 20018.188481
## iter  90 value 20017.270463
## final  value 20017.251381
## converged
## # weights:  36
## initial  value 29722.763310
## iter  10 value 20462.328597
## iter  20 value 20256.256665
## iter  30 value 20165.295735
## iter  40 value 20079.760959
## iter  50 value 20028.191547
## iter  60 value 20007.725222
## iter  70 value 19992.919793
## iter  80 value 19989.382511
## iter  90 value 19984.559570
## iter 100 value 19981.601046
## final  value 19981.601046
## stopped after 100 iterations
## # weights:  8
## initial  value 27373.629310
## iter  10 value 20710.032272
## iter  20 value 20265.636178
## iter  30 value 20202.210224
## iter  40 value 20197.636791
## iter  50 value 20197.587775
## final  value 20197.586990
## converged
## # weights:  22
## initial  value 27499.391189
## iter  10 value 20276.846938
## iter  20 value 20170.312025
## iter  30 value 20087.787201
## iter  40 value 20047.325060
## iter  50 value 20042.308233
## iter  60 value 20033.504110
## iter  70 value 20014.760232
## iter  80 value 20011.726617
## iter  90 value 20010.405556
## iter 100 value 20010.352349
## final  value 20010.352349
## stopped after 100 iterations
## # weights:  36
## initial  value 28724.636485
## iter  10 value 20700.445870
## iter  20 value 20297.413495
## iter  30 value 20179.499867
## iter  40 value 20087.236853
## iter  50 value 20047.807320
## iter  60 value 20030.684050
## iter  70 value 20025.494688
## iter  80 value 20018.599464
## iter  90 value 20008.920657
## iter 100 value 19994.688503
## final  value 19994.688503
## stopped after 100 iterations
## # weights:  8
## initial  value 27291.085528
## iter  10 value 21860.071123
## iter  20 value 20734.922364
## iter  30 value 20305.923121
## iter  40 value 20201.862358
## iter  50 value 20184.717046
## iter  60 value 20183.490269
## iter  70 value 20182.666539
## iter  70 value 20182.666466
## iter  70 value 20182.666365
## final  value 20182.666365
## converged
## # weights:  22
## initial  value 28578.189962
## iter  10 value 20507.969706
## iter  20 value 20200.573312
## iter  30 value 20045.318776
## iter  40 value 20025.958363
## iter  50 value 20015.512556
## iter  60 value 20010.447248
## iter  70 value 20007.450291
## iter  80 value 20006.908657
## iter  90 value 20006.837712
## final  value 20006.818498
## converged
## # weights:  36
## initial  value 29568.695334
## iter  10 value 20186.489125
## iter  20 value 20052.673135
## iter  30 value 20036.005993
## iter  40 value 20017.034722
## iter  50 value 19991.647543
## iter  60 value 19977.313284
## iter  70 value 19974.134932
## iter  80 value 19973.556106
## iter  90 value 19971.481662
## iter 100 value 19969.042421
## final  value 19969.042421
## stopped after 100 iterations
## # weights:  8
## initial  value 27394.653725
## iter  10 value 21761.017847
## iter  20 value 20311.957320
## iter  30 value 20256.279602
## iter  40 value 20243.643923
## iter  50 value 20242.719628
## final  value 20242.679526
## converged
## # weights:  22
## initial  value 28232.840671
## iter  10 value 20399.833288
## iter  20 value 20301.551389
## iter  30 value 20178.773073
## iter  40 value 20106.224449
## iter  50 value 20071.319148
## iter  60 value 20063.873747
## iter  70 value 20052.600326
## iter  80 value 20041.130164
## iter  90 value 20040.225703
## iter 100 value 20040.102699
## final  value 20040.102699
## stopped after 100 iterations
## # weights:  36
## initial  value 26755.868286
## iter  10 value 20488.196791
## iter  20 value 20213.583104
## iter  30 value 20150.971938
## iter  40 value 20100.385195
## iter  50 value 20073.259782
## iter  60 value 20048.074387
## iter  70 value 20036.726175
## iter  80 value 20032.979734
## iter  90 value 20029.472485
## iter 100 value 20023.445931
## final  value 20023.445931
## stopped after 100 iterations
## # weights:  8
## initial  value 30758.395337
## iter  10 value 21368.400525
## iter  20 value 20331.436452
## iter  30 value 20261.873534
## iter  40 value 20253.059344
## iter  50 value 20252.341600
## final  value 20252.245213
## converged
## # weights:  22
## initial  value 27743.867033
## iter  10 value 20984.575038
## iter  20 value 20406.804641
## iter  30 value 20212.754940
## iter  40 value 20136.878960
## iter  50 value 20096.748767
## iter  60 value 20089.007221
## iter  70 value 20065.608823
## iter  80 value 20050.725794
## iter  90 value 20045.793333
## iter 100 value 20045.234287
## final  value 20045.234287
## stopped after 100 iterations
## # weights:  36
## initial  value 30654.765893
## iter  10 value 20317.006713
## iter  20 value 20145.284030
## iter  30 value 20103.054377
## iter  40 value 20082.833306
## iter  50 value 20053.000689
## iter  60 value 20031.710273
## iter  70 value 20022.302143
## iter  80 value 20018.973155
## iter  90 value 20014.448105
## iter 100 value 20012.250894
## final  value 20012.250894
## stopped after 100 iterations
## # weights:  8
## initial  value 26799.496453
## iter  10 value 24127.446802
## iter  20 value 20759.276338
## iter  30 value 20303.856845
## iter  40 value 20251.922162
## iter  50 value 20242.932188
## final  value 20242.748199
## converged
## # weights:  22
## initial  value 28436.607384
## iter  10 value 20523.096408
## iter  20 value 20301.926565
## iter  30 value 20107.020540
## iter  40 value 20054.636545
## iter  50 value 20047.887934
## iter  60 value 20044.158656
## iter  70 value 20041.482448
## iter  80 value 20039.988170
## iter  90 value 20039.360312
## iter 100 value 20039.314153
## final  value 20039.314153
## stopped after 100 iterations
## # weights:  36
## initial  value 29259.034977
## iter  10 value 20278.223097
## iter  20 value 20083.996320
## iter  30 value 20058.150023
## iter  40 value 20035.433609
## iter  50 value 20021.269848
## iter  60 value 20016.049555
## iter  70 value 20015.131698
## iter  80 value 20014.979065
## iter  90 value 20014.913785
## iter 100 value 20014.547775
## final  value 20014.547775
## stopped after 100 iterations
## # weights:  8
## initial  value 28728.267010
## iter  10 value 22578.678709
## iter  20 value 20682.057081
## iter  30 value 20333.710878
## iter  40 value 20277.340450
## iter  50 value 20264.469952
## iter  60 value 20263.743216
## iter  70 value 20263.161669
## final  value 20263.139394
## converged
## # weights:  22
## initial  value 26423.876103
## iter  10 value 20742.775700
## iter  20 value 20409.022711
## iter  30 value 20229.655468
## iter  40 value 20195.718463
## iter  50 value 20173.592346
## iter  60 value 20168.998209
## iter  70 value 20161.337393
## iter  80 value 20157.767869
## iter  90 value 20137.503561
## iter 100 value 20132.550851
## final  value 20132.550851
## stopped after 100 iterations
## # weights:  36
## initial  value 26838.286932
## iter  10 value 20177.859347
## iter  20 value 20101.331389
## iter  30 value 20089.092014
## iter  40 value 20075.109714
## iter  50 value 20061.247208
## iter  60 value 20055.357676
## iter  70 value 20052.315916
## iter  80 value 20051.618935
## iter  90 value 20050.731079
## iter 100 value 20048.978713
## final  value 20048.978713
## stopped after 100 iterations
## # weights:  8
## initial  value 27961.866902
## iter  10 value 23074.206347
## iter  20 value 22093.749909
## iter  30 value 21308.441384
## iter  40 value 20376.070580
## iter  50 value 20288.025260
## iter  60 value 20277.350522
## iter  70 value 20276.339863
## iter  70 value 20276.339858
## iter  70 value 20276.339858
## final  value 20276.339858
## converged
## # weights:  22
## initial  value 29436.882889
## iter  10 value 20758.790784
## iter  20 value 20325.369424
## iter  30 value 20259.673648
## iter  40 value 20221.256250
## iter  50 value 20150.371390
## iter  60 value 20113.630055
## iter  70 value 20098.079379
## iter  80 value 20093.608543
## iter  90 value 20091.859864
## iter 100 value 20088.827595
## final  value 20088.827595
## stopped after 100 iterations
## # weights:  36
## initial  value 28417.532675
## iter  10 value 21028.117121
## iter  20 value 20413.832211
## iter  30 value 20284.645720
## iter  40 value 20208.623652
## iter  50 value 20132.537347
## iter  60 value 20113.238736
## iter  70 value 20094.945354
## iter  80 value 20085.812749
## iter  90 value 20079.211312
## iter 100 value 20069.041726
## final  value 20069.041726
## stopped after 100 iterations
## # weights:  8
## initial  value 31172.703825
## iter  10 value 23390.607027
## iter  20 value 21235.224394
## iter  30 value 20530.020290
## iter  40 value 20292.452102
## iter  50 value 20270.385430
## iter  60 value 20263.635020
## final  value 20263.594541
## converged
## # weights:  22
## initial  value 27678.206007
## iter  10 value 22146.753768
## iter  20 value 21363.842619
## iter  30 value 20362.601085
## iter  40 value 20297.310530
## iter  50 value 20268.444091
## iter  60 value 20221.878841
## iter  70 value 20150.550220
## iter  80 value 20108.457002
## iter  90 value 20081.849377
## iter 100 value 20078.119056
## final  value 20078.119056
## stopped after 100 iterations
## # weights:  36
## initial  value 28235.757360
## iter  10 value 20294.768325
## iter  20 value 20137.976456
## iter  30 value 20095.113349
## iter  40 value 20068.819794
## iter  50 value 20055.107659
## iter  60 value 20048.431832
## iter  70 value 20045.591981
## iter  80 value 20044.647591
## iter  90 value 20043.357423
## iter 100 value 20038.291689
## final  value 20038.291689
## stopped after 100 iterations
## # weights:  8
## initial  value 26417.574526
## iter  10 value 21809.702085
## iter  20 value 20420.971676
## iter  30 value 20281.392885
## iter  40 value 20256.374884
## iter  50 value 20249.234152
## final  value 20248.871649
## converged
## # weights:  22
## initial  value 34543.455264
## iter  10 value 21993.013484
## iter  20 value 20962.354892
## iter  30 value 20354.831110
## iter  40 value 20251.706825
## iter  50 value 20234.883512
## iter  60 value 20222.114777
## iter  70 value 20198.401029
## iter  80 value 20153.977169
## iter  90 value 20136.056172
## iter 100 value 20128.621255
## final  value 20128.621255
## stopped after 100 iterations
## # weights:  36
## initial  value 28392.001925
## iter  10 value 21155.592459
## iter  20 value 20421.006957
## iter  30 value 20259.170983
## iter  40 value 20188.793626
## iter  50 value 20146.624953
## iter  60 value 20107.015138
## iter  70 value 20076.019654
## iter  80 value 20062.697590
## iter  90 value 20058.221915
## iter 100 value 20048.315827
## final  value 20048.315827
## stopped after 100 iterations
## # weights:  8
## initial  value 26964.483820
## iter  10 value 21687.606848
## iter  20 value 20672.544592
## iter  30 value 20341.787518
## iter  40 value 20267.854686
## iter  50 value 20259.078029
## final  value 20259.052882
## converged
## # weights:  22
## initial  value 27200.366980
## iter  10 value 20480.359266
## iter  20 value 20301.753263
## iter  30 value 20128.106457
## iter  40 value 20094.465618
## iter  50 value 20087.673461
## iter  60 value 20082.768602
## iter  70 value 20078.742362
## iter  80 value 20075.129692
## iter  90 value 20071.566508
## iter 100 value 20070.527336
## final  value 20070.527336
## stopped after 100 iterations
## # weights:  36
## initial  value 26851.054653
## iter  10 value 20243.155717
## iter  20 value 20128.105465
## iter  30 value 20093.649094
## iter  40 value 20066.261179
## iter  50 value 20046.389658
## iter  60 value 20038.163040
## iter  70 value 20035.242229
## iter  80 value 20034.815981
## iter  90 value 20034.022969
## iter 100 value 20033.376720
## final  value 20033.376720
## stopped after 100 iterations
## # weights:  8
## initial  value 29000.770221
## iter  10 value 20612.791439
## iter  20 value 20299.245532
## iter  30 value 20256.861059
## iter  40 value 20250.219237
## iter  50 value 20248.683785
## final  value 20248.619786
## converged
## # weights:  22
## initial  value 28967.406723
## iter  10 value 20504.782459
## iter  20 value 20286.757548
## iter  30 value 20186.310331
## iter  40 value 20135.619936
## iter  50 value 20108.050960
## iter  60 value 20097.344965
## iter  70 value 20078.557222
## iter  80 value 20074.306124
## iter  90 value 20072.974830
## iter 100 value 20072.852174
## final  value 20072.852174
## stopped after 100 iterations
## # weights:  36
## initial  value 27260.178636
## iter  10 value 20399.362113
## iter  20 value 20185.258552
## iter  30 value 20140.727802
## iter  40 value 20110.054317
## iter  50 value 20095.787769
## iter  60 value 20069.422029
## iter  70 value 20058.624788
## iter  80 value 20052.152169
## iter  90 value 20047.358189
## iter 100 value 20042.351225
## final  value 20042.351225
## stopped after 100 iterations
## # weights:  8
## initial  value 27768.247847
## iter  10 value 20481.578592
## iter  20 value 20197.048021
## iter  30 value 20180.175991
## iter  40 value 20178.683957
## final  value 20178.667094
## converged
## # weights:  22
## initial  value 26481.298699
## iter  10 value 21045.872157
## iter  20 value 20526.573100
## iter  30 value 20246.022348
## iter  40 value 20143.940796
## iter  50 value 20108.480705
## iter  60 value 20085.647820
## iter  70 value 20008.718201
## iter  80 value 20001.815950
## iter  90 value 20000.773770
## iter 100 value 20000.706448
## final  value 20000.706448
## stopped after 100 iterations
## # weights:  36
## initial  value 31520.750516
## iter  10 value 20292.736041
## iter  20 value 20067.481801
## iter  30 value 20018.950754
## iter  40 value 19984.779088
## iter  50 value 19958.421041
## iter  60 value 19953.126761
## iter  70 value 19951.890368
## iter  80 value 19951.383593
## iter  90 value 19950.739109
## iter 100 value 19950.342561
## final  value 19950.342561
## stopped after 100 iterations
## # weights:  8
## initial  value 28288.768356
## iter  10 value 21497.100313
## iter  20 value 20570.040488
## iter  30 value 20232.328811
## iter  40 value 20197.167477
## iter  50 value 20190.473690
## final  value 20190.473428
## converged
## # weights:  22
## initial  value 28198.760369
## iter  10 value 21239.549769
## iter  20 value 20911.372536
## iter  30 value 20217.577771
## iter  40 value 20044.881982
## iter  50 value 20016.006910
## iter  60 value 20006.452217
## iter  70 value 19995.729392
## iter  80 value 19992.617283
## iter  90 value 19992.190565
## final  value 19992.180839
## converged
## # weights:  36
## initial  value 26960.023341
## iter  10 value 21030.877584
## iter  20 value 20479.285342
## iter  30 value 20324.006976
## iter  40 value 20189.442506
## iter  50 value 20065.547737
## iter  60 value 20040.515303
## iter  70 value 20010.568595
## iter  80 value 19992.952141
## iter  90 value 19982.331133
## iter 100 value 19974.365465
## final  value 19974.365465
## stopped after 100 iterations
## # weights:  8
## initial  value 27135.907439
## iter  10 value 22643.305938
## iter  20 value 21561.796336
## iter  30 value 21238.666530
## iter  40 value 20401.085043
## iter  50 value 20203.230040
## iter  60 value 20184.650192
## iter  70 value 20178.572394
## final  value 20178.542531
## converged
## # weights:  22
## initial  value 26476.658834
## iter  10 value 22172.890074
## iter  20 value 21323.448278
## iter  30 value 20364.833195
## iter  40 value 20051.178872
## iter  50 value 20022.072650
## iter  60 value 20009.699106
## iter  70 value 19991.989419
## iter  80 value 19986.910416
## iter  90 value 19986.425504
## final  value 19986.419835
## converged
## # weights:  36
## initial  value 29175.567742
## iter  10 value 20209.096423
## iter  20 value 20039.281980
## iter  30 value 20015.427048
## iter  40 value 19984.719593
## iter  50 value 19974.605185
## iter  60 value 19968.525536
## iter  70 value 19964.452072
## iter  80 value 19962.976152
## iter  90 value 19960.281971
## iter 100 value 19958.482026
## final  value 19958.482026
## stopped after 100 iterations
## # weights:  36
## initial  value 33172.352904
## iter  10 value 25218.647485
## iter  20 value 25114.728115
## iter  30 value 25088.228096
## iter  40 value 25059.191615
## iter  50 value 25035.173589
## iter  60 value 25029.659170
## iter  70 value 25024.883167
## iter  80 value 25020.378806
## iter  90 value 25017.091219
## iter 100 value 25010.702772
## final  value 25010.702772
## stopped after 100 iterations
```

```
summary(model)
```

```
## a 5-5-1 network with 36 weights
## options were - entropy fitting
##  b->h1 i1->h1 i2->h1 i3->h1 i4->h1 i5->h1
##  -1.91  -0.46  -0.43  -1.35  -0.86   0.29
##  b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2
##  -1.24  -1.06  -1.20   0.31   0.14   0.03
##  b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3
##   1.29  -0.16   0.21   0.05   0.15   0.56
##  b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4
##  -6.51  -0.13  -0.96  -5.95  -2.18   0.69
##  b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5
##  -1.38  -4.28   1.07   0.18  -1.41  -2.77
##  b->o h1->o h2->o h3->o h4->o h5->o
## -4.30 -4.68 -1.92  7.33  2.05 -0.34
```

```r
test_set$pred <- predict(model, test_set)
test_set$factor_pred <- as.factor(test_set$pred)
test_set$factor_truth <- as.factor(test_set$Diabetes_binary)
precision <- posPredValue(test_set$factor_truth, test_set$factor_pred)
recall <- sensitivity(test_set$factor_truth, test_set$factor_pred)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table
test_set$pred <- predict(model, test_set,probability=TRUE)
#________________________________________________
print(paste("Precision: ",precision))
```

```
## [1] "Precision:  0.695425991362387"
```

```r
print(paste("Accuracy: ",accuracy))
```

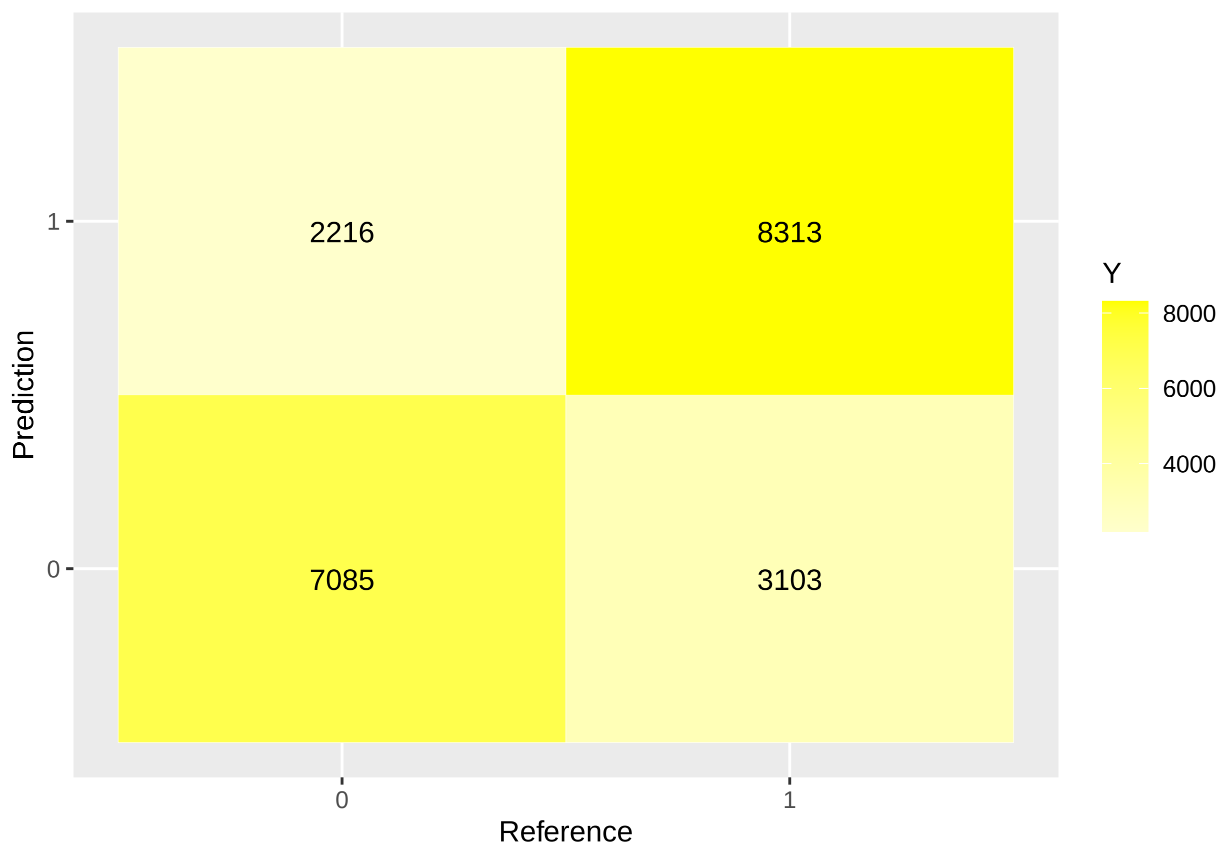
```
## [1] "Accuracy:  0.743254332190954"
```

```r
print(paste("Recall: ",recall))
```

```
## [1] "Recall:  0.761746048811956"
```

```r
acc_nn <- accuracy
#________________________________________________
Reference <- factor(c(0, 1, 0, 1))
Prediction   <- factor(c(0, 0, 1, 1))
Y  <- array(cmat)
df <- data.frame(Prediction, Reference, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow")
```



```r
model <- train(Diabetes_binary ~ Age+GenHlth+HighChol+HighBP+BMI, method='ctree', data = training_set)
summary(model)
```

```
##     Length      Class       Mode 
##          1 BinaryTree         S4
```

```r
test_set$pred <- predict(model, test_set)
test_set$factor_pred <- as.factor(test_set$pred)
test_set$factor_truth <- as.factor(test_set$Diabetes_binary)
precision <- posPredValue(test_set$factor_truth, test_set$factor_pred)
recall <- sensitivity(test_set$factor_truth, test_set$factor_pred)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table
test_set$pred <- predict(model, test_set,probability=TRUE)
#________________________________________________
print(paste("Precision: ",precision))
```

```
## [1] "Precision:  0.677856301531213"
```

```r
print(paste("Accuracy: ",accuracy))
```

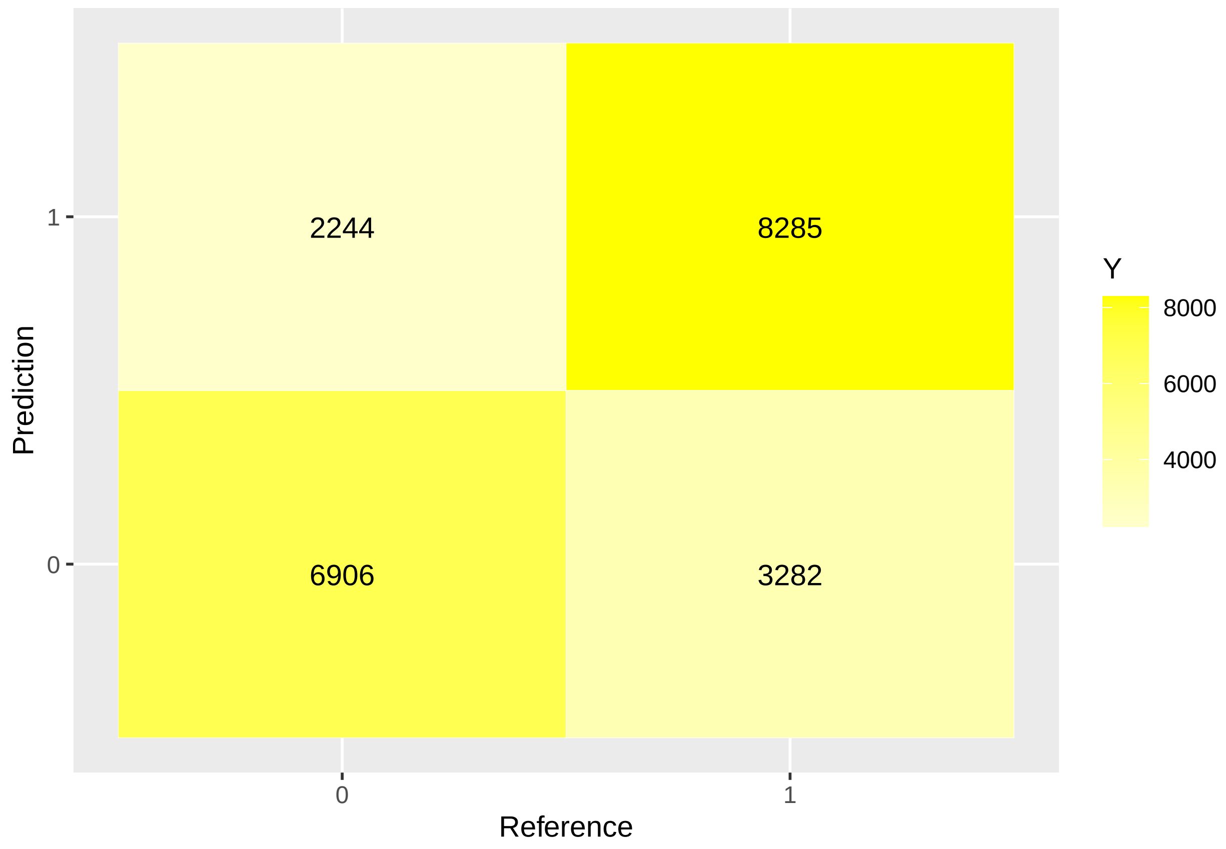
```
## [1] "Accuracy:  0.73326253801226"
```

```r
print(paste("Recall: ",recall))
```

```
## [1] "Recall:  0.754754098360656"
```

```r
acc_tree <- accuracy
#________________________________________________
Reference <- factor(c(0, 1, 0, 1))
Prediction   <- factor(c(0, 0, 1, 1))
Y  <- array(cmat)
df <- data.frame(Prediction, Reference, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow")
```



```r
model <- train(Diabetes_binary ~ Age+GenHlth+HighChol+HighBP+BMI, method='svmLinear', data = training_se
summary(model)
```

```
## Length  Class   Mode
##      1   ksvm     S4
```

```r
test_set$pred <- predict(model, test_set)
test_set$factor_pred <- as.factor(test_set$pred)
test_set$factor_truth <- as.factor(test_set$Diabetes_binary)
precision <- posPredValue(test_set$factor_truth, test_set$factor_pred)
recall <- sensitivity(test_set$factor_truth, test_set$factor_pred)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table
```

```
test_set$pred <- predict(model, test_set,probability=TRUE)
#________________________________________________
print(paste("Precision: ",precision))
```

```
## [1] "Precision:  0.686788378484492"
```

```
print(paste("Accuracy: ",accuracy))
```

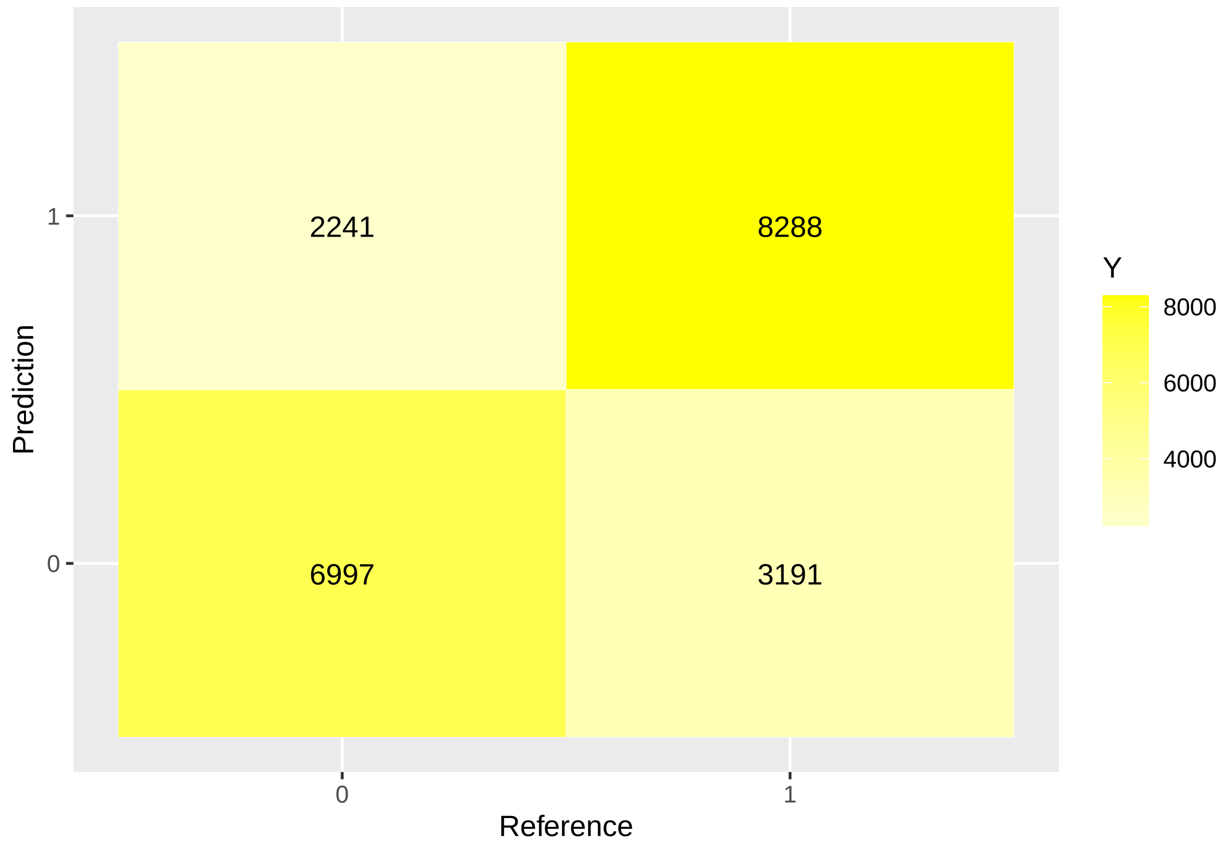
```
## [1] "Accuracy:  0.737799874499204"
```

```
print(paste("Recall: ",recall))
```

```
## [1] "Recall:  0.757415024897164"
```

```
acc_svm <- accuracy
#________________________________________________
Reference <- factor(c(0, 1, 0, 1))
Prediction   <- factor(c(0, 0, 1, 1))
Y  <- array(cmat)
df <- data.frame(Prediction, Reference, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow")
```



```r
model <- train(Diabetes_binary ~ Age+GenHlth+HighChol+HighBP+BMI, method='LogitBoost', data = training_s
summary(model)
```

```
##             Length Class      Mode
## Stump       33     -none-     numeric
## lablist      2     factor     numeric
## xNames       5     -none-     character
## problemType  1     -none-     character
## tuneValue    1     data.frame list
## obsLevels    2     -none-     character
## param        0     -none-     list
```

```r
test_set$pred <- predict(model, test_set)
test_set$factor_pred <- as.factor(test_set$pred)
test_set$factor_truth <- as.factor(test_set$Diabetes_binary)
precision <- posPredValue(test_set$factor_truth, test_set$factor_pred)
recall <- sensitivity(test_set$factor_truth, test_set$factor_pred)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table
test_set$pred <- predict(model, test_set,probability=TRUE)
#________________________________________________
print(paste("Precision: ",precision))
```

```
## [1] "Precision:  0.742343934040047"
```

```r
print(paste("Accuracy: ",accuracy))
```

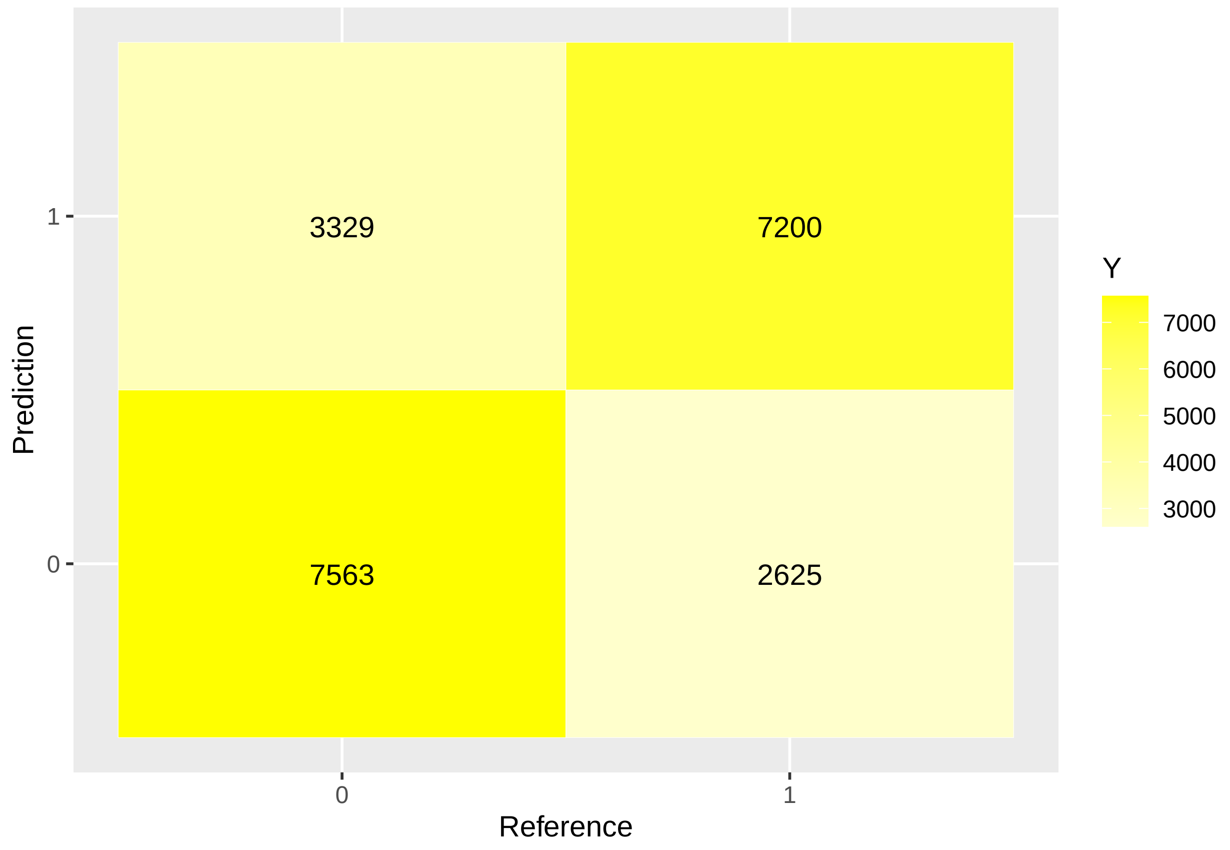
```
## [1] "Accuracy:  0.712603176135541"
```

```r
print(paste("Recall: ",recall))
```

```
## [1] "Recall:  0.694362835108336"
```

```r
acc_lr <- accuracy
#________________________________________________
Reference <- factor(c(0, 1, 0, 1))
Prediction   <- factor(c(0, 0, 1, 1))
Y  <- array(cmat)
df <- data.frame(Prediction, Reference, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow")
```



```r
model <- train(Diabetes_binary ~ Age+GenHlth+HighChol+HighBP+BMI, method='knn', data = training_set, me
summary(model)
```

```
##             Length Class      Mode
## learn       2      -none-     list
## k           1      -none-     numeric
## theDots     0      -none-     list
## xNames      5      -none-     character
## problemType 1      -none-     character
## tuneValue   1      data.frame list
## obsLevels   2      -none-     character
## param       0      -none-     list
```

```r
test_set$pred <- predict(model, test_set)
test_set$factor_pred <- as.factor(test_set$pred)
test_set$factor_truth <- as.factor(test_set$Diabetes_binary)
precision <- posPredValue(test_set$factor_truth, test_set$factor_pred)
recall <- sensitivity(test_set$factor_truth, test_set$factor_pred)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table
test_set$pred <- predict(model, test_set,probability=TRUE)
#________________________________________________
print(paste("Precision: ",precision))
```

```
## [1] "Precision:  0.691794267765999"
```

```r
print(paste("Accuracy: ",accuracy))
```

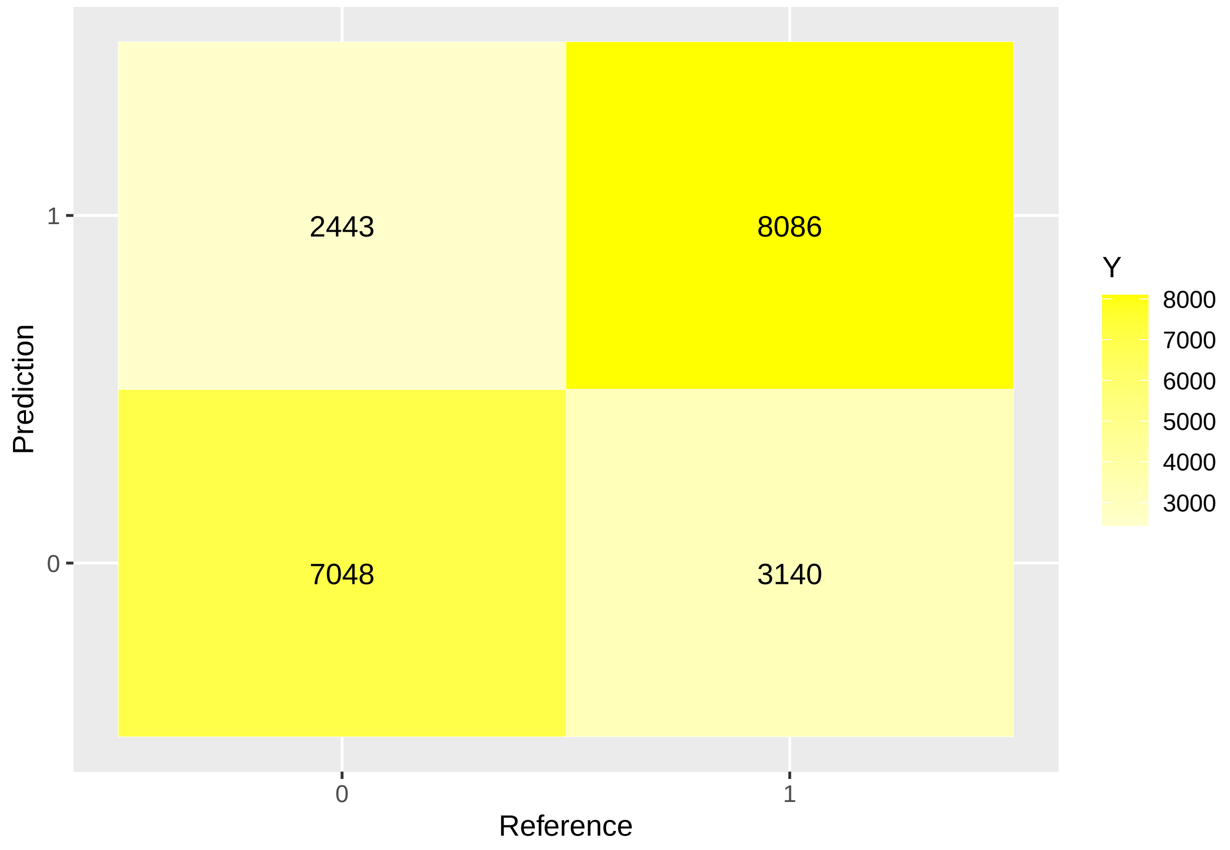
```
## [1] "Accuracy:  0.730511174397838"
```

```r
print(paste("Recall: ",recall))
```

```
## [1] "Recall:  0.742598250974608"
```

```r
acc_knn <- accuracy
#_______________________________________________
Reference <- factor(c(0, 1, 0, 1))
Prediction   <- factor(c(0, 0, 1, 1))
Y  <- array(cmat)
df <- data.frame(Prediction, Reference, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = Reference, y = Prediction)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow")
```



```r
print(paste('Accuracy in QDA:',acc_qda*100,'%'))
```

```
## [1] "Accuracy in QDA: 73.3793502920307 %"
```

```r
print(paste('Accuracy in LDA:',acc_lda*100,'%'))
```

```
## [1] "Accuracy in LDA: 73.6496596997635 %"
```

```
print(paste('Accuracy in Randomforrest:',acc_rf*100,'%'))
```

```
## [1] "Accuracy in Randomforrest: 73.9441038760438 %"
```

```
print(paste('Accuracy in Neural Network:',acc_nn*100,'%'))
```

```
## [1] "Accuracy in Neural Network: 74.3254332190954 %"
```

```
print(paste('Accuracy in CTree:',acc_tree*100,'%'))
```

```
## [1] "Accuracy in CTree: 73.326253801226 %"
```

```
print(paste('Accuracy in Linear SVM:',acc_svm*100,'%'))
```

```
## [1] "Accuracy in Linear SVM: 73.7799874499204 %"
```

```
print(paste('Accuracy in Logreg:',acc_lr*100,'%'))
```

```
## [1] "Accuracy in Logreg: 71.2603176135541 %"
```

```
print(paste('Accuracy in knn:',acc_knn*100,'%'))
```

```
## [1] "Accuracy in knn: 73.0511174397838 %"
```

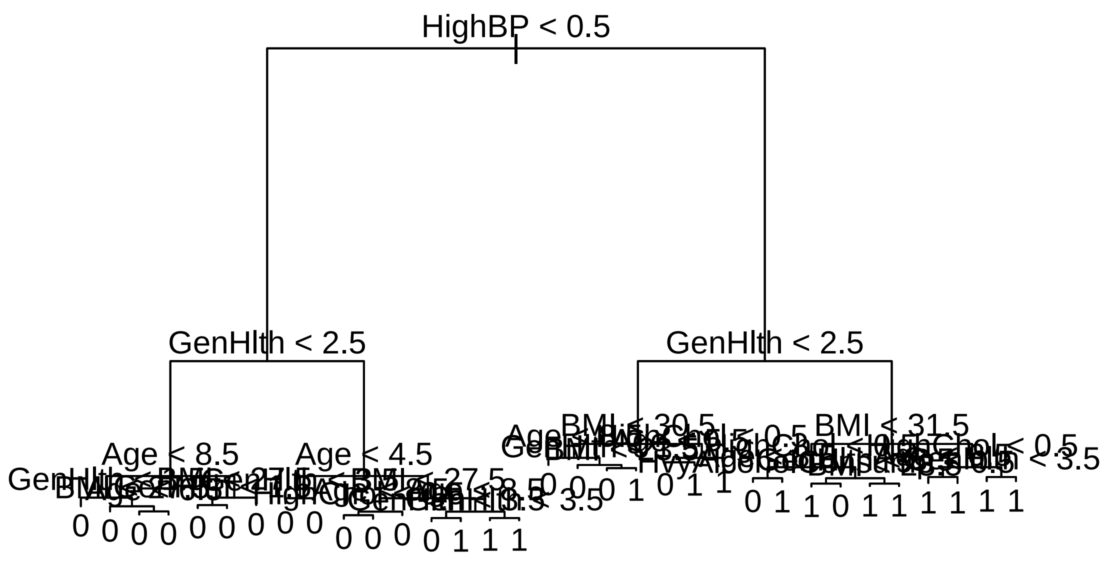
PART 5

Yes, we can do this thing. For this target, we need to use a simple and appropriate inferentially. As we want to ask some questions as less as possible from a person, I prefer to use the Tree model. In the last part, we understood Tree can do nice predictions too. so use a big tree and use cross-validation to get how many nodes and terminals we need for a good model. So we choose 8 and get accuracy, a plot of the Tree, and a confusion matrix for this model. By this tree, we can by asking just some questions, we can predict whether one has diabetes or not, with acceptable accuracy.

```
tree_model <- tree(Diabetes_binary ~ .-output , training_set , mindev=7e-4, minsize=8)
summary(tree_model)
```

```
##
## Classification tree:
## tree(formula = Diabetes_binary ~ . - output, data = training_set,
##     mindev = 7e-04, minsize = 8)
## Variables actually used in tree construction:
## [1] "HighBP"            "GenHlth"           "Age"
## [4] "BMI"               "HighChol"          "HvyAlcoholConsump"
## Number of terminal nodes:  33
## Residual mean deviance:  1.064 = 51400 / 48310
## Misclassification error rate: 0.2685 = 12978 / 48340
```

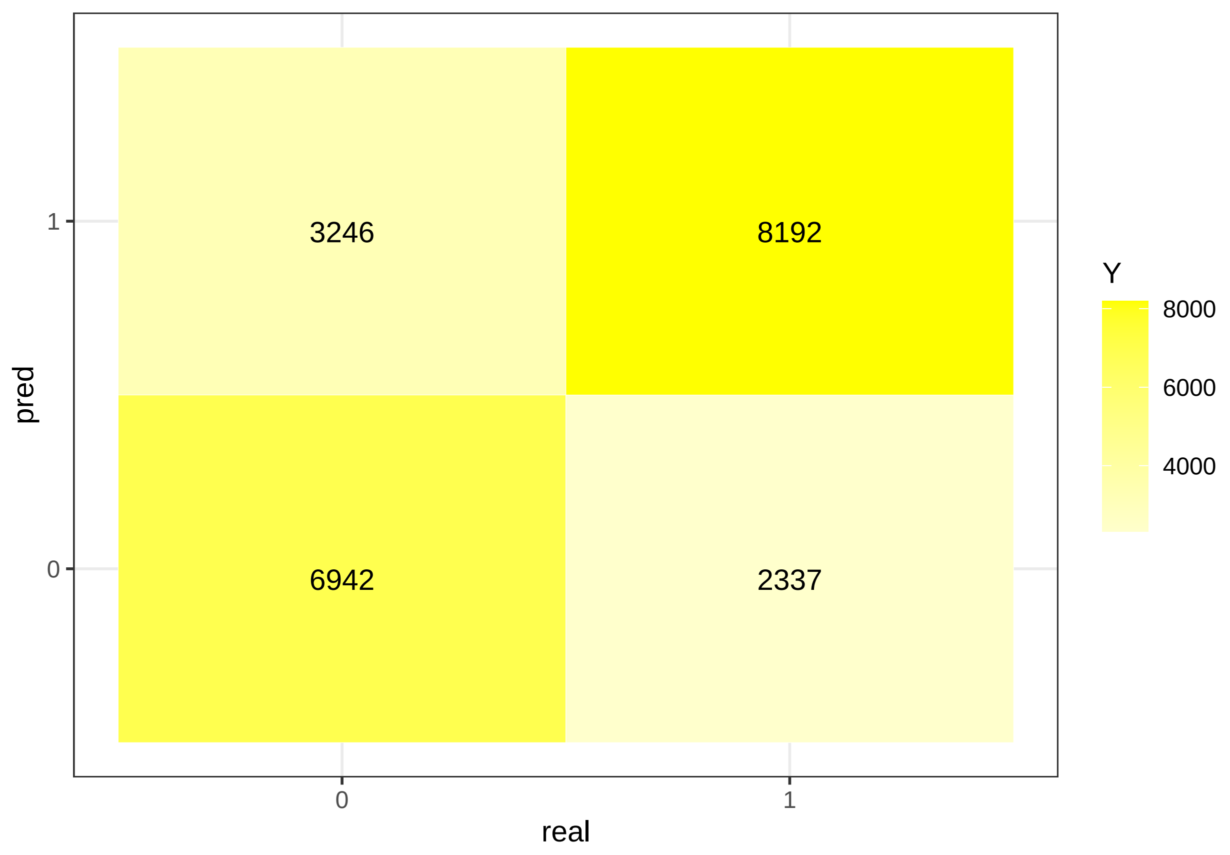
```
plot(tree_model)
text(tree_model, pretty = 0)
```



```r
test_set$pred <- predict(tree_model, test_set,type="class")
ErrorPrune<-mean(test_set$pred!=test_set$Diabetes_binary)
cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table

real <- factor(c(0, 0, 1, 1))
pred   <- factor(c(0, 1, 0, 1))
Y  <- array(cmat)
df <- data.frame(pred, real, Y)

ggplot(data =  df, mapping = aes(x = real, y = pred)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow") +
  theme_bw()
```



```r
print(paste('Error rate :',ErrorPrune))
```

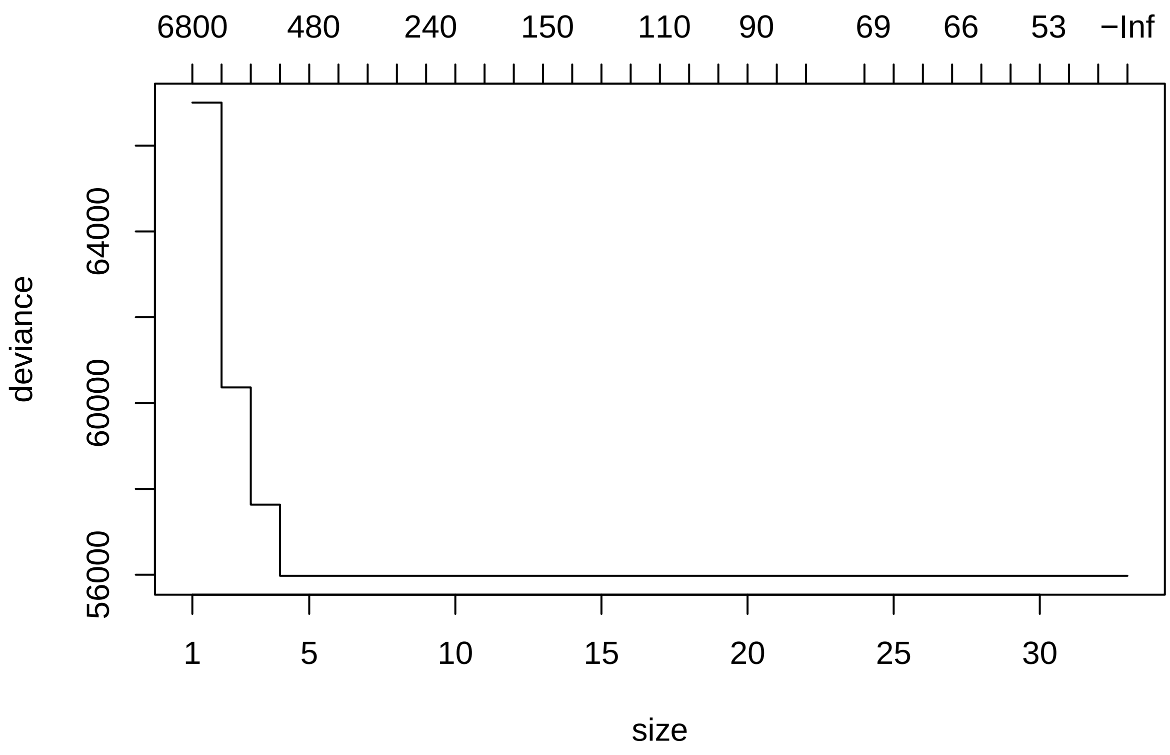
```
## [1] "Error rate : 0.269488825602162"
```

```r
cv.tree<-cv.tree(tree_model, FUN=prune.tree)
cv.tree
```

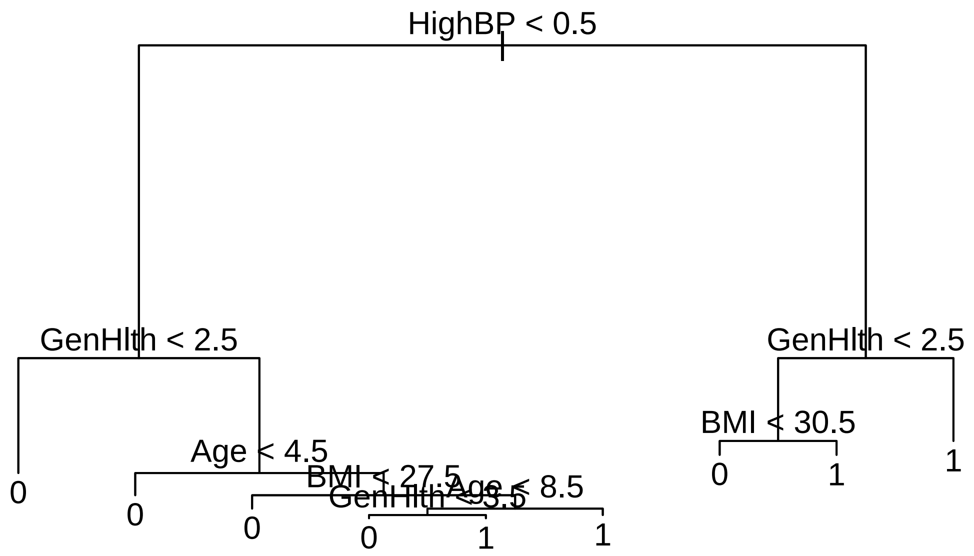
```
## $size
##  [1] 33 32 31 30 29 28 27 26 25 24 22 21 20 19 18 17 16 15 14 13 12 11 10  9  8
## [26]  7  6  5  4  3  2  1
##
## $dev
##  [1] 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35
##  [9] 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35
## [17] 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35 55975.35
## [25] 55975.35 55975.35 55975.35 55975.35 55975.35 57633.17 60364.83 67002.35
##
## $k
##  [1]       -Inf   48.01600   48.50432   53.19121   62.48372   65.39000
##  [7]   65.59553   67.79974   68.82130   69.10782   77.24479   80.21836
## [13]   89.66217   95.64933  110.37263  111.01990  115.88648  129.18833
## [19]  138.87016  154.00557  161.24952  186.67753  212.97755  238.36911
## [25]  289.57154  300.56292  466.27489  479.05378  502.23111 1789.48871
## [31] 2483.36772 6759.14441
##
## $method
## [1] "deviance"
```

```
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
plot(cv.tree)
```



```
prune.tree<-prune.misclass(tree_model,best = 8)
plot(prune.tree);text(prune.tree)
```

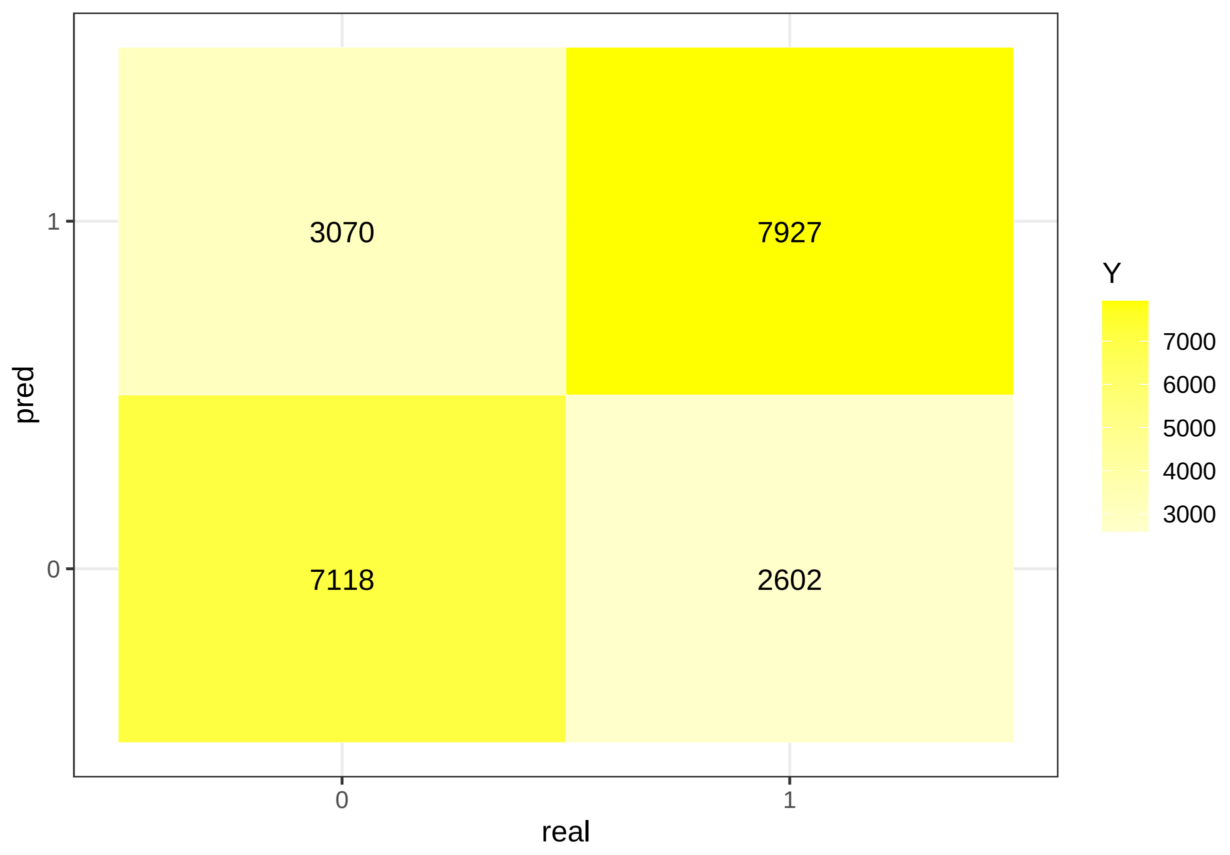


```
test_set$pred <- predict(prune.tree, test_set,type="class")
ErrorPrune<-mean(test_set$pred!=test_set$Diabetes_binary)

cm <- confusionMatrix(test_set$pred, test_set$Diabetes_binary)
accuracy <- cm$overall[1]
cmat <- cm$table

real <- factor(c(0, 0, 1, 1))
pred   <- factor(c(0, 1, 0, 1))
Y  <- array(cmat)
df <- data.frame(pred, real, Y)

library(ggplot2)
ggplot(data =  df, mapping = aes(x = real, y = pred)) +
  geom_tile(aes(fill = Y), colour = "white") +
  geom_text(aes(label = sprintf("%1.0f", Y)), vjust = 1) +
  scale_fill_gradient(low = "#ffffcc", high = "yellow") +
  theme_bw()
```



```
print(paste('New Error rate :',ErrorPrune))
```

```
## [1] "New Error rate : 0.273784814403363"
```

```
print(paste('New accuracy :',((1-ErrorPrune)*100),'%'))
```

```
## [1] "New accuracy : 72.621518559637 %"
```