Untitled

BRAYAN ALEXIS VARGAS PEREDO

2023-08-01

NAIVE BAYES

Cargando los datos:

```
setwd("C:/Users/Brayan/Downloads/9no SEMESTRE - 2023-1/6. MINERIA DE DATOS/TERCER PARCIAL/Naive Bayes")

#-----------------------------------------------------------------#
#   Naive Bayes                                                   #
#-----------------------------------------------------------------#

#########################################################################
# # Ejemplo: Diabetes                                             # #
#########################################################################
diabetes = read.csv("DiabetesTrain.csv")
#diabetes=read.csv(file.choose())
dim(diabetes)
```

```
## [1] 115   4
```

```
head(diabetes)
```

```
##   glucose insulin sspg  class
## 1      97     289  117 normal
## 2     105     319  143 normal
## 3      90     356  199 normal
## 4      90     323  240 normal
## 5      86     381  157 normal
## 6     100     350  221 normal
```

```
diabetes$class = factor(diabetes$class)
```

Sin discretizar:

```
# Sin discretizar
library(e1071)

a<-naiveBayes(class ~ .,data = diabetes)
a
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
```

```
##
## A-priori probabilities:
## Y
## chemical   normal    overt
## 0.226087 0.573913 0.200000
##
## Conditional probabilities:
##           glucose
## Y               [,1]      [,2]
##   chemical  99.46154  8.805593
##   normal    91.98485  8.164637
##   overt    207.17391 71.837982
##
##           insulin
## Y               [,1]      [,2]
##   chemical  504.1154  60.05819
##   normal    351.2121  37.69861
##   overt    1002.9565 315.85288
##
##           sspg
## Y              [,1]      [,2]
##   chemical 291.7692 177.65479
##   normal   169.0152  65.48952
##   overt    112.6087 106.57253
```

```
pred = predict(a,diabetes[,-4],type="raw")
pred1 = factor(max.col(pred), labels = levels(diabetes$class))
```

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
confusionMatrix(pred1,diabetes[,4])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical       20      3     3
##   normal          6     63     0
##   overt           0      0    20
##
## Overall Statistics
##
##                Accuracy : 0.8957
##                  95% CI : (0.8248, 0.9449)
##     No Information Rate : 0.5739
##     P-Value [Acc > NIR] : 3.778e-14
##
##                   Kappa : 0.8169
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
```

```
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   0.7692        0.9545       0.8696
## Specificity                   0.9326        0.8776       1.0000
## Pos Pred Value                0.7692        0.9130       1.0000
## Neg Pred Value                0.9326        0.9348       0.9684
## Prevalence                    0.2261        0.5739       0.2000
## Detection Rate                0.1739        0.5478       0.1739
## Detection Prevalence          0.2261        0.6000       0.1739
## Balanced Accuracy             0.8509        0.9160       0.9348
```

```
# utilizando otra libreria
library(naivebayes)
```

```
## naivebayes 0.9.7 loaded
```

```
a<-naive_bayes(class ~ .,data = diabetes)
a
```

```
##
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.formula(formula = class ~ ., data = diabetes)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##
## chemical   normal    overt
## 0.226087 0.573913 0.200000
##
## ---------------------------------------------------------------------------------
##
##  Tables:
##
## ---------------------------------------------------------------------------------
##  ::: glucose (Gaussian)
## ---------------------------------------------------------------------------------
##
## glucose   chemical     normal      overt
##    mean  99.461538  91.984848 207.173913
##    sd     8.805593   8.164637  71.837982
##
## ---------------------------------------------------------------------------------
##  ::: insulin (Gaussian)
## ---------------------------------------------------------------------------------
##
## insulin   chemical     normal      overt
##    mean  504.11538  351.21212 1002.95652
##    sd     60.05819   37.69861  315.85288
##
```

```
## ------------------------------------------------------------------------------
##  ::: sspg (Gaussian)
## ------------------------------------------------------------------------------
##
## sspg    chemical    normal     overt
##   mean 291.76923 169.01515 112.60870
##   sd   177.65479  65.48952 106.57253
##
## ------------------------------------------------------------------------------
```

```
pred=predict(a,diabetes[,-4])
confusionMatrix(pred,diabetes[,4])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical       20      3     3
##   normal          6     63     0
##   overt           0      0    20
##
## Overall Statistics
##
##                Accuracy : 0.8957
##                  95% CI : (0.8248, 0.9449)
##     No Information Rate : 0.5739
##     P-Value [Acc > NIR] : 3.778e-14
##
##                   Kappa : 0.8169
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   0.7692        0.9545       0.8696
## Specificity                   0.9326        0.8776       1.0000
## Pos Pred Value                0.7692        0.9130       1.0000
## Neg Pred Value                0.9326        0.9348       0.9684
## Prevalence                    0.2261        0.5739       0.2000
## Detection Rate                0.1739        0.5478       0.1739
## Detection Prevalence          0.2261        0.6000       0.1739
## Balanced Accuracy             0.8509        0.9160       0.9348
```

```
predict(a,diabetes[,-4],type = "prob")
```

```
##             chemical        normal        overt
##   [1,]  5.295056e-04  9.990214e-01 4.491287e-04
##   [2,]  2.339643e-03  9.971217e-01 5.386942e-04
##   [3,]  2.275912e-03  9.976460e-01 7.807859e-05
##   [4,]  1.180417e-03  9.987288e-01 9.074437e-05
##   [5,]  4.361057e-03  9.954717e-01 1.672017e-04
##   [6,]  6.386586e-03  9.934653e-01 1.481366e-04
##   [7,]  2.183812e-04  9.996164e-01 1.652600e-04
##   [8,]  1.109668e-02  9.886942e-01 2.091171e-04
```

```
##   [9,]  5.561492e-04  9.991428e-01 3.010012e-04
##  [10,]  2.860365e-03  9.970620e-01 7.768089e-05
##  [11,]  2.737109e-04  9.995964e-01 1.298984e-04
##  [12,]  9.429950e-05  9.990956e-01 8.101451e-04
##  [13,]  4.685303e-03  9.952171e-01 9.757051e-05
##  [14,]  7.145331e-03  9.924505e-01 4.041340e-04
##  [15,]  2.577827e-03  9.973178e-01 1.043346e-04
##  [16,]  5.755875e-03  9.941035e-01 1.405755e-04
##  [17,]  1.940953e-04  9.994795e-01 3.264158e-04
##  [18,]  1.206339e-03  9.986771e-01 1.165314e-04
##  [19,]  5.334159e-03  9.945073e-01 1.585289e-04
##  [20,]  2.698566e-04  9.995533e-01 1.768468e-04
##  [21,]  7.047837e-04  9.992159e-01 7.928524e-05
##  [22,]  5.066611e-02  9.491241e-01 2.098169e-04
##  [23,]  6.040562e-03  9.937900e-01 1.694724e-04
##  [24,]  4.388028e-03  9.954543e-01 1.577203e-04
##  [25,]  4.236783e-04  9.994793e-01 9.706961e-05
##  [26,]  1.851161e-03  9.980173e-01 1.315720e-04
##  [27,]  5.266643e-04  9.969041e-01 2.569231e-03
##  [28,]  2.002639e-03  9.978337e-01 1.636298e-04
##  [29,]  4.942068e-04  9.990224e-01 4.834124e-04
##  [30,]  5.673222e-03  9.942287e-01 9.806927e-05
##  [31,]  2.769180e-03  9.970957e-01 1.351083e-04
##  [32,]  1.196466e-02  9.877739e-01 2.614334e-04
##  [33,]  9.187795e-03  9.906305e-01 1.816555e-04
##  [34,]  6.866383e-04  9.990540e-01 2.593738e-04
##  [35,]  8.178601e-02  9.165555e-01 1.658472e-03
##  [36,]  1.860864e-02  9.810572e-01 3.341775e-04
##  [37,]  1.232439e-02  9.875490e-01 1.266344e-04
##  [38,]  4.917400e-04  9.990449e-01 4.633993e-04
##  [39,]  9.000279e-05  9.982301e-01 1.679900e-03
##  [40,]  1.080090e-03  9.986026e-01 3.173065e-04
##  [41,]  3.708905e-03  9.961129e-01 1.782156e-04
##  [42,]  2.054200e-04  9.995771e-01 2.174545e-04
##  [43,]  7.433120e-04  9.991504e-01 1.063132e-04
##  [44,]  7.221347e-04  9.991566e-01 1.212746e-04
##  [45,]  3.258039e-04  9.995439e-01 1.303026e-04
##  [46,]  1.254320e-03  9.986309e-01 1.147530e-04
##  [47,]  3.679488e-04  9.995214e-01 1.106108e-04
##  [48,]  8.575833e-04  9.988566e-01 2.858096e-04
##  [49,]  1.275039e-03  9.983903e-01 3.346247e-04
##  [50,]  9.521719e-01  4.412351e-02 3.704616e-03
##  [51,]  4.828539e-02  9.510608e-01 6.537688e-04
##  [52,]  5.440347e-01  4.450488e-01 1.091648e-02
##  [53,]  2.667509e-01  7.317704e-01 1.478681e-03
##  [54,]  3.289868e-04  9.994252e-01 2.458509e-04
##  [55,]  2.113070e-01  7.871383e-01 1.554676e-03
##  [56,]  1.067948e-01  8.923053e-01 8.998199e-04
##  [57,]  1.784725e-02  9.818881e-01 2.646238e-04
##  [58,]  3.200394e-02  9.671740e-01 8.220511e-04
##  [59,]  2.527538e-02  9.744571e-01 2.675100e-04
##  [60,]  9.727327e-03  9.899078e-01 3.649193e-04
##  [61,]  7.682488e-03  9.922034e-01 1.141017e-04
##  [62,]  3.744839e-01  6.238293e-01 1.686796e-03
```

```
##  [63,]  7.721793e-01  2.230828e-01 4.737862e-03
##  [64,]  2.003020e-01  7.989477e-01 7.503410e-04
##  [65,]  3.440964e-03  9.964530e-01 1.060147e-04
##  [66,]  2.381242e-02  9.760112e-01 1.763595e-04
##  [67,]  2.275912e-03  9.976460e-01 7.807859e-05
##  [68,]  6.761531e-02  9.312615e-01 1.123204e-03
##  [69,]  9.994407e-01  5.208362e-04 3.847674e-05
##  [70,]  6.424901e-02  9.345319e-01 1.219088e-03
##  [71,]  7.408047e-04  9.989515e-01 3.077439e-04
##  [72,]  1.000000e+00  2.338103e-22 9.485134e-09
##  [73,]  9.786212e-01  5.814184e-06 2.137299e-02
##  [74,]  9.999997e-01  5.502353e-11 3.409691e-07
##  [75,]  9.850467e-01  1.231466e-07 1.495318e-02
##  [76,]  9.999758e-01  1.838167e-09 2.423367e-05
##  [77,]  9.999997e-01  7.331410e-15 3.136628e-07
##  [78,]  9.803726e-01  1.880635e-02 8.210894e-04
##  [79,]  9.932796e-01  2.821303e-09 6.720425e-03
##  [80,]  9.810128e-01  1.852270e-02 4.645416e-04
##  [81,]  9.999985e-01  3.314769e-11 1.488563e-06
##  [82,]  9.987065e-01  4.618651e-06 1.288886e-03
##  [83,]  9.999115e-01  3.207163e-05 5.645985e-05
##  [84,]  9.602919e-01  3.594148e-02 3.766582e-03
##  [85,]  3.592417e-01  6.378324e-01 2.925914e-03
##  [86,]  9.990254e-01  5.759318e-06 9.688743e-04
##  [87,]  9.821413e-01  4.745440e-07 1.785827e-02
##  [88,]  9.777282e-01  2.132689e-02 9.449512e-04
##  [89,]  9.678721e-01  5.272532e-07 3.212741e-02
##  [90,]  1.012043e-01  8.981187e-01 6.769971e-04
##  [91,]  6.472607e-01  1.473760e-12 3.527393e-01
##  [92,]  9.923048e-01  6.191171e-05 7.633263e-03
##  [93,] 6.897663e-173  0.000000e+00 1.000000e+00
##  [94,]  4.976878e-03  2.580640e-21 9.950231e-01
##  [95,] 2.477275e-146 1.702646e-304 1.000000e+00
##  [96,]  7.133953e-60 2.662460e-137 1.000000e+00
##  [97,]  1.166381e-35  1.111443e-88 1.000000e+00
##  [98,] 1.851074e-159 5.163499e-300 1.000000e+00
##  [99,]  5.174439e-21  5.308143e-55 1.000000e+00
## [100,]  2.488687e-42  2.626983e-96 1.000000e+00
## [101,]  3.141734e-04  1.022311e-15 9.996858e-01
## [102,] 7.579262e-131 5.081788e-267 1.000000e+00
## [103,]  2.659830e-95 1.653866e-183 1.000000e+00
## [104,]  3.774942e-77 5.127724e-169 1.000000e+00
## [105,]  1.938547e-12  2.930057e-45 1.000000e+00
## [106,]  9.973924e-01  1.162370e-10 2.607642e-03
## [107,]  1.467368e-50 3.974579e-111 1.000000e+00
## [108,] 2.314463e-209  0.000000e+00 1.000000e+00
## [109,]  6.664180e-01  1.305333e-07 3.335819e-01
## [110,]  2.259447e-03  1.233813e-18 9.977406e-01
## [111,]  8.066183e-01  1.579498e-13 1.933817e-01
## [112,]  2.863917e-06  7.337970e-28 9.999971e-01
## [113,]  7.552413e-34  1.074422e-84 1.000000e+00
## [114,] 8.805182e-203  0.000000e+00 1.000000e+00
## [115,]  4.892801e-52 4.067389e-116 1.000000e+00
```

```r
a<-naive_bayes(class ~ .,data = diabetes,usekernel = TRUE)
a
```

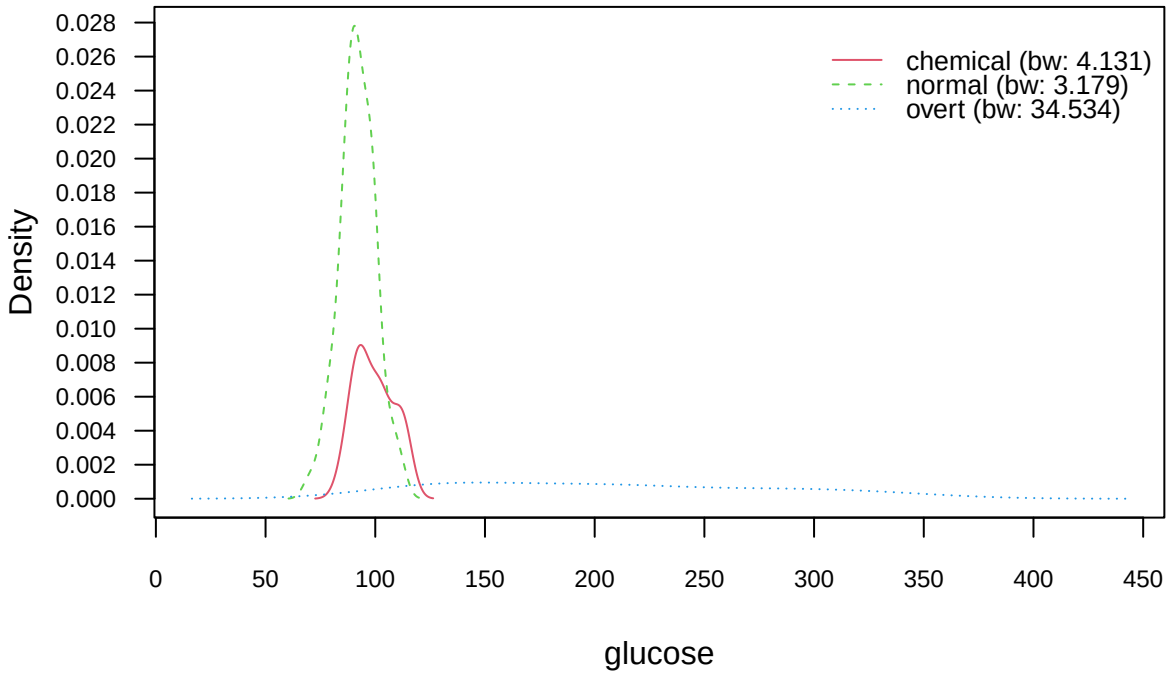
```
##
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.formula(formula = class ~ ., data = diabetes, usekernel = TRUE)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##
## chemical   normal    overt
## 0.226087 0.573913 0.200000
##
## ---------------------------------------------------------------------------------
##
##  Tables:
##
## ---------------------------------------------------------------------------------
##  ::: glucose::chemical (KDE)
## ---------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (26 obs.);   Bandwidth 'bw' = 4.131
##
##        x                y
##  Min.   : 72.61   Min.   :4.949e-05
##  1st Qu.: 86.05   1st Qu.:3.464e-03
##  Median : 99.50   Median :2.127e-02
##  Mean   : 99.50   Mean   :1.857e-02
##  3rd Qu.:112.95   3rd Qu.:3.093e-02
##  Max.   :126.39   Max.   :3.998e-02
##
## ---------------------------------------------------------------------------------
##  ::: glucose::normal (KDE)
## ---------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (66 obs.);   Bandwidth 'bw' = 3.179
##
##        x                y
##  Min.   : 60.46   Min.   :2.145e-05
##  1st Qu.: 75.73   1st Qu.:2.132e-03
##  Median : 91.00   Median :8.905e-03
```

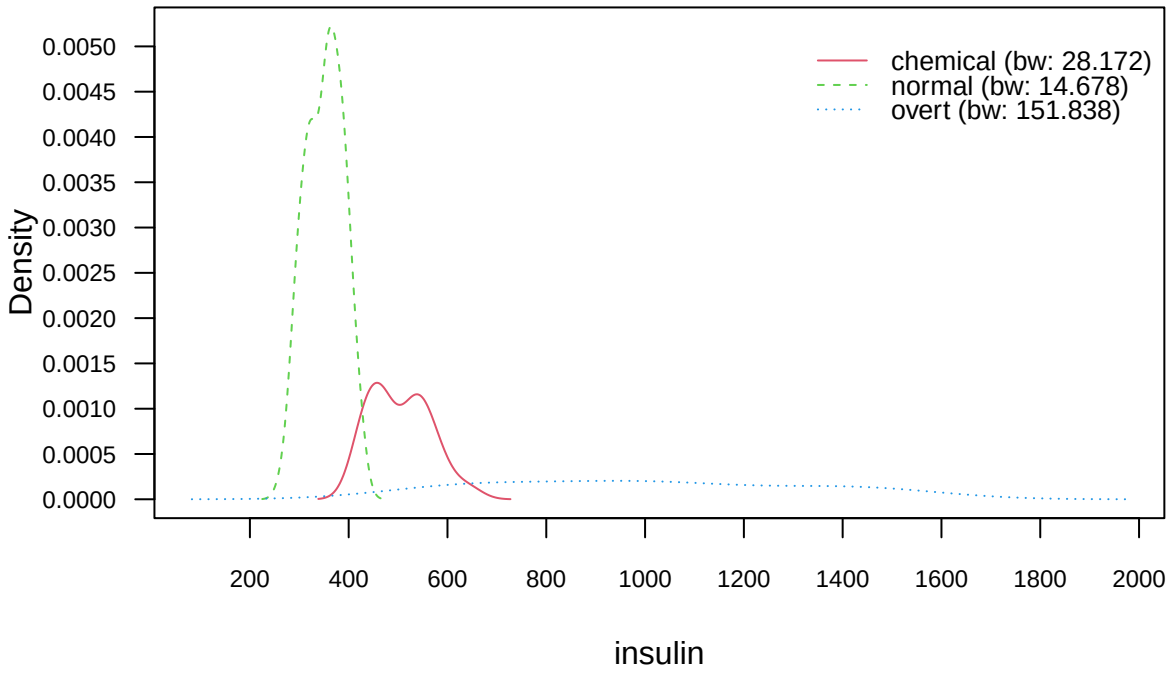
```
##  Mean   : 91.00   Mean   :1.636e-02
##  3rd Qu.:106.27   3rd Qu.:3.114e-02
##  Max.   :121.54   Max.   :4.846e-02
##
## ------------------------------------------------------------------------------
##  ::: glucose::overt (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (23 obs.);   Bandwidth 'bw' = 34.53
##
##        x               y
##  Min.   : 16.4   Min.   :8.447e-06
##  1st Qu.:122.9   1st Qu.:5.362e-04
##  Median :229.5   Median :2.638e-03
##  Mean   :229.5   Mean   :2.343e-03
##  3rd Qu.:336.1   3rd Qu.:3.926e-03
##  Max.   :442.6   Max.   :4.754e-03
##
## ------------------------------------------------------------------------------
##  ::: insulin::chemical (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (26 obs.);   Bandwidth 'bw' = 28.17
##
##        x               y
##  Min.   :338.5   Min.   :6.131e-06
##  1st Qu.:435.7   1st Qu.:4.410e-04
##  Median :533.0   Median :2.180e-03
##  Mean   :533.0   Mean   :2.567e-03
##  3rd Qu.:630.3   3rd Qu.:4.770e-03
##  Max.   :727.5   Max.   :5.687e-03
##
## ------------------------------------------------------------------------------
##  ::: insulin::normal (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (66 obs.);   Bandwidth 'bw' = 14.68
##
##        x               y
##  Min.   :225.0   Min.   :4.725e-06
##  1st Qu.:286.2   1st Qu.:5.085e-04
##  Median :347.5   Median :3.890e-03
##  Mean   :347.5   Mean   :4.076e-03
##  3rd Qu.:408.8   3rd Qu.:7.314e-03
##  Max.   :470.0   Max.   :9.098e-03
```

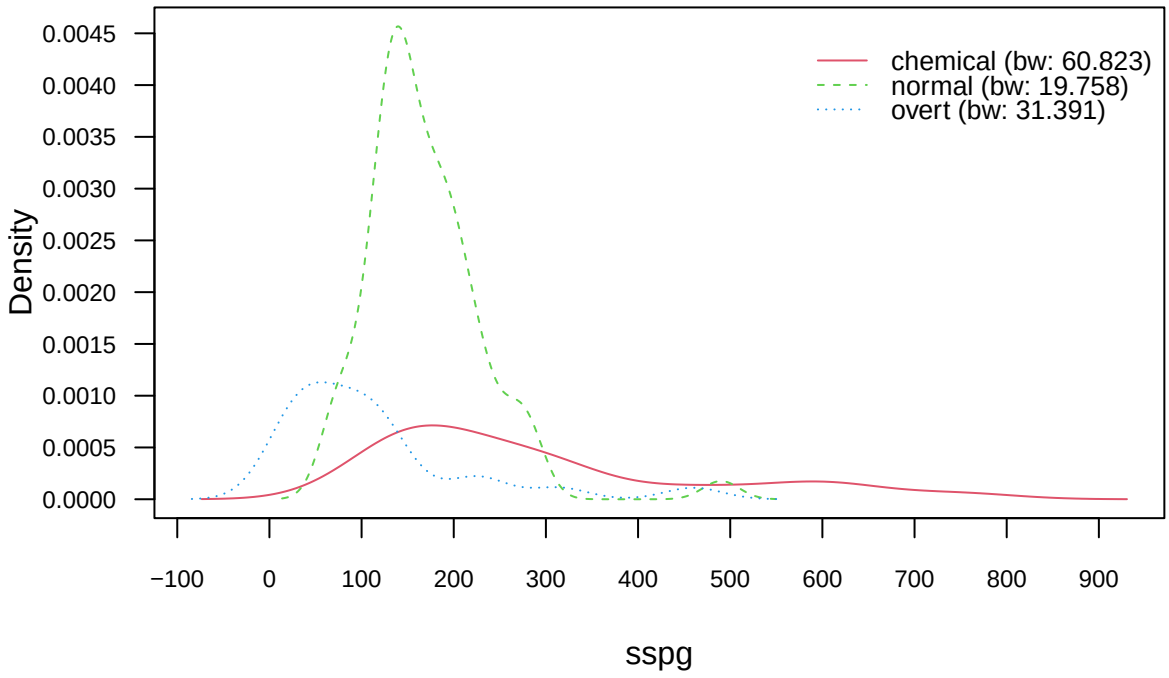
```
##
## ------------------------------------------------------------------------------
##  ::: insulin::overt (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (23 obs.);   Bandwidth 'bw' = 151.8
##
##        x                 y
##  Min.   :  82.48   Min.   :2.495e-06
##  1st Qu.: 555.74   1st Qu.:1.141e-04
##  Median :1029.00   Median :6.280e-04
##  Mean   :1029.00   Mean   :5.276e-04
##  3rd Qu.:1502.26   3rd Qu.:8.772e-04
##  Max.   :1975.52   Max.   :1.019e-03
##
## ------------------------------------------------------------------------------
##  ::: sspg::chemical (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (26 obs.);   Bandwidth 'bw' = 60.82
##
##        x                y
##  Min.   :-73.47   Min.   :2.835e-06
##  1st Qu.:177.52   1st Qu.:2.706e-04
##  Median :428.50   Median :6.522e-04
##  Mean   :428.50   Mean   :9.949e-04
##  3rd Qu.:679.48   3rd Qu.:1.536e-03
##  Max.   :930.47   Max.   :3.155e-03
##
## ------------------------------------------------------------------------------
##  ::: sspg::normal (KDE)
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (66 obs.);   Bandwidth 'bw' = 19.76
##
##        x                y
##  Min.   : 13.73   Min.   :1.000e-09
##  1st Qu.:147.61   1st Qu.:2.739e-05
##  Median :281.50   Median :3.686e-04
##  Mean   :281.50   Mean   :1.865e-03
##  3rd Qu.:415.39   3rd Qu.:2.967e-03
##  Max.   :549.27   Max.   :7.958e-03
##
## ------------------------------------------------------------------------------
##  ::: sspg::overt (KDE)
```

```
## ------------------------------------------------------------------------------
##
## Call:
##  density.default(x = x, na.rm = TRUE)
##
## Data: x (23 obs.);   Bandwidth 'bw' = 31.39
##
##        x                y
##  Min.   :-84.17   Min.   :6.215e-06
##  1st Qu.: 75.41   1st Qu.:2.632e-04
##  Median :235.00   Median :5.806e-04
##  Mean   :235.00   Mean   :1.565e-03
##  3rd Qu.:394.59   3rd Qu.:2.299e-03
##  Max.   :554.17   Max.   :5.651e-03
##
## ------------------------------------------------------------------------------
```

```
plot(a)
```




chemical (bw: 28.172)
normal (bw: 14.678)
overt (bw: 151.838)
Density
0.0050
0.0045
0.0040
0.0035
0.0030
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
200
400
600
800
1000
1200
1400
1600
1800
2000
insulin




```
pred=predict(a,diabetes[,-4])
confusionMatrix(pred,diabetes[,4])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical       22      2     3
##   normal          4     64     0
##   overt           0      0    20
##
## Overall Statistics
##
##                Accuracy : 0.9217
##                  95% CI : (0.8566, 0.9636)
##     No Information Rate : 0.5739
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8634
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   0.8462        0.9697       0.8696
```

```
## Specificity                 0.9438       0.9184       1.0000
## Pos Pred Value              0.8148       0.9412       1.0000
## Neg Pred Value              0.9545       0.9574       0.9684
## Prevalence                  0.2261       0.5739       0.2000
## Detection Rate              0.1913       0.5565       0.1739
## Detection Prevalence        0.2348       0.5913       0.1739
## Balanced Accuracy           0.8950       0.9440       0.9348
```

INTERPRETACION:

- Si una persona se encuentra en la categoria normal, su variacion de Glucosa es aproximadamente entre [50;300], con una media igual a 150 de Glucosa.
- Si una persona se encuentra en la categoria normal, su variacion de Insulina es aproximadamente entre [50;300], con una media igual a 150 de Insulina.
- Si una persona se encuentra en la categoria normal, su variacion de sspg es aproximadamente entre [50;300], con una media igual a 150 de sspg

Discretizando:

```
# DISCRETIZANDO:

# Discretizando por el metodo Chi-Merge
library(discretization)
d_diab=chiM(diabetes,0.01)$Disc.data
for (i in 1:3){
  d_diab[,i]=as.factor(d_diab[,i])}
b<-naive_bayes(class ~ .,data = d_diab)
```

```
## Warning: naive_bayes(): Feature glucose - zero probabilities are present.
## Consider Laplace smoothing.
```

```
## Warning: naive_bayes(): Feature insulin - zero probabilities are present.
## Consider Laplace smoothing.
```

```
## Warning: naive_bayes(): Feature sspg - zero probabilities are present. Consider
## Laplace smoothing.
```

```
b
```

```
##
## ================================== Naive Bayes ==================================
##
##  Call:
## naive_bayes.formula(formula = class ~ ., data = d_diab)
##
## ---------------------------------------------------------------------------------
##
## Laplace smoothing: 0
##
## ---------------------------------------------------------------------------------
##
##  A priori probabilities:
##
## chemical   normal    overt
## 0.226087 0.573913 0.200000
```

```
##
## ------------------------------------------------------------------------------
##
##  Tables:
##
## ------------------------------------------------------------------------------
##  ::: glucose (Categorical)
## ------------------------------------------------------------------------------
##
## glucose   chemical     normal      overt
##       1 0.53846154 0.90909091 0.00000000
##       2 0.46153846 0.09090909 0.00000000
##       3 0.00000000 0.00000000 1.00000000
##
## ------------------------------------------------------------------------------
##  ::: insulin (Categorical)
## ------------------------------------------------------------------------------
##
## insulin   chemical     normal      overt
##       1 0.00000000 0.98484848 0.00000000
##       2 1.00000000 0.01515152 0.17391304
##       3 0.00000000 0.00000000 0.82608696
##
## ------------------------------------------------------------------------------
##  ::: sspg (Categorical)
## ------------------------------------------------------------------------------
##
## sspg   chemical     normal      overt
##    1 0.00000000 0.00000000 0.39130435
##    2 0.15384615 0.36363636 0.43478261
##    3 0.42307692 0.62121212 0.08695652
##    4 0.42307692 0.01515152 0.08695652
##
## ------------------------------------------------------------------------------
```

```
pred=predict(b,d_diab[,-4])
confusionMatrix(pred,d_diab[,4])
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical       26      1     0
##   normal          0     65     0
##   overt           0      0    23
##
## Overall Statistics
##
##                Accuracy : 0.9913
##                  95% CI : (0.9525, 0.9998)
##     No Information Rate : 0.5739
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9851
##
```

```
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   1.0000        0.9848          1.0
## Specificity                   0.9888        1.0000          1.0
## Pos Pred Value                0.9630        1.0000          1.0
## Neg Pred Value                1.0000        0.9800          1.0
## Prevalence                    0.2261        0.5739          0.2
## Detection Rate                0.2261        0.5652          0.2
## Detection Prevalence          0.2348        0.5652          0.2
## Balanced Accuracy             0.9944        0.9924          1.0
```

```
# extras de ejemplos

## Ejemplo con datos categoricos
data(Titanic)
m <- naiveBayes(Survived ~ ., data = Titanic)
m
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.formula(formula = Survived ~ ., data = Titanic)
##
## A-priori probabilities:
## Survived
##       No      Yes
## 0.676965 0.323035
##
## Conditional probabilities:
##         Class
## Survived        1st        2nd        3rd       Crew
##      No  0.08187919 0.11208054 0.35436242 0.45167785
##      Yes 0.28551336 0.16596343 0.25035162 0.29817159
##
##         Sex
## Survived       Male     Female
##      No  0.91543624 0.08456376
##      Yes 0.51617440 0.48382560
##
##         Age
## Survived      Child      Adult
##      No  0.03489933 0.96510067
##      Yes 0.08016878 0.91983122
```

```
predict(m, as.data.frame(Titanic))
```

```
##  [1] Yes No  No  No  Yes Yes Yes Yes No  No  No  No  Yes Yes Yes Yes Yes No  No
## [20] No  Yes Yes Yes Yes No  No  No  No  Yes Yes Yes Yes
## Levels: No Yes
```

```
## Ejemplo con predictores m?tricos:
data(iris)
```

```r
m <- naiveBayes(Species ~ ., data = iris)
## De manera alternativa:
m <- naiveBayes(iris[,-5], iris[,5])
m
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = iris[, -5], y = iris[, 5])
##
## A-priori probabilities:
## iris[, 5]
##     setosa versicolor  virginica
##  0.3333333  0.3333333  0.3333333
##
## Conditional probabilities:
##             Sepal.Length
## iris[, 5]      [,1]      [,2]
##   setosa     5.006 0.3524897
##   versicolor 5.936 0.5161711
##   virginica  6.588 0.6358796
##
##             Sepal.Width
## iris[, 5]      [,1]      [,2]
##   setosa     3.428 0.3790644
##   versicolor 2.770 0.3137983
##   virginica  2.974 0.3224966
##
##             Petal.Length
## iris[, 5]      [,1]      [,2]
##   setosa     1.462 0.1736640
##   versicolor 4.260 0.4699110
##   virginica  5.552 0.5518947
##
##             Petal.Width
## iris[, 5]      [,1]      [,2]
##   setosa     0.246 0.1053856
##   versicolor 1.326 0.1977527
##   virginica  2.026 0.2746501
```

```r
confusionMatrix(predict(m, iris), iris[,5])
```

```
## Confusion Matrix and Statistics
##
##             Reference
## Prediction   setosa versicolor virginica
##   setosa         50          0         0
##   versicolor      0         47         3
##   virginica       0          3        47
##
## Overall Statistics
##
##                Accuracy : 0.96
##                  95% CI : (0.915, 0.9852)
```

```
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.94
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: setosa Class: versicolor Class: virginica
## Sensitivity                 1.0000            0.9400           0.9400
## Specificity                 1.0000            0.9700           0.9700
## Pos Pred Value              1.0000            0.9400           0.9400
## Neg Pred Value              1.0000            0.9700           0.9700
## Prevalence                  0.3333            0.3333           0.3333
## Detection Rate              0.3333            0.3133           0.3133
## Detection Prevalence        0.3333            0.3333           0.3333
## Balanced Accuracy           1.0000            0.9550           0.9550
```

K VECINOS MAS CERCANOS

```
#-----------------------------------------------------------------------#
#  K-NN                                                                 #
#-----------------------------------------------------------------------#

#########################################################################
# # Ejemplo: Diabetes                                                 # #
# #  # train 70%   test 30%
#########################################################################
diabetes.train = read.csv("DiabetesTrain.csv")
diabetes.test = read.csv("DiabetesTest.csv")
#diabetes=read.csv(file.choose())
head(diabetes.train)
```

```
##   glucose insulin sspg  class
## 1      97     289  117 normal
## 2     105     319  143 normal
## 3      90     356  199 normal
## 4      90     323  240 normal
## 5      86     381  157 normal
## 6     100     350  221 normal
```

```
library(class)
b<-knn(train = diabetes.train[,1:3],
       test = diabetes.test[,1:3],
       cl = diabetes.train[,4])
#Estimacion del error por resubstituciOn
confusionMatrix(b,factor(diabetes.test[,4]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical        9      1     0
```

```
##   normal          1      9     0
##   overt           0      0    10
##
## Overall Statistics
##
##                Accuracy : 0.9333
##                  95% CI : (0.7793, 0.9918)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 8.747e-12
##
##                   Kappa : 0.9
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   0.9000        0.9000       1.0000
## Specificity                   0.9500        0.9500       1.0000
## Pos Pred Value                0.9000        0.9000       1.0000
## Neg Pred Value                0.9500        0.9500       1.0000
## Prevalence                    0.3333        0.3333       0.3333
## Detection Rate                0.3000        0.3000       0.3333
## Detection Prevalence          0.3333        0.3333       0.3333
## Balanced Accuracy             0.9250        0.9250       1.0000
```

```r
# con K=3
k_3 <- knn(train = diabetes.train[,1:3],test = diabetes.test[,1:3],cl = diabetes.train[,4], k = 3)
confusionMatrix(k_3,factor(diabetes.test[,4]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical        8      0     0
##   normal          2     10     0
##   overt           0      0    10
##
## Overall Statistics
##
##                Accuracy : 0.9333
##                  95% CI : (0.7793, 0.9918)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 8.747e-12
##
##                   Kappa : 0.9
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   0.8000        1.0000       1.0000
## Specificity                   1.0000        0.9000       1.0000
## Pos Pred Value                1.0000        0.8333       1.0000
```

```
## Neg Pred Value                 0.9091        1.0000       1.0000
## Prevalence                     0.3333        0.3333       0.3333
## Detection Rate                 0.2667        0.3333       0.3333
## Detection Prevalence           0.2667        0.4000       0.3333
## Balanced Accuracy              0.9000        0.9500       1.0000
```

```
mean(k_3 == diabetes[,4])
```

```
## Warning in `==.default`(k_3, diabetes[, 4]): longitud de objeto mayor no es
## múltiplo de la longitud de uno menor
```

```
## Warning in is.na(e1) | is.na(e2): longitud de objeto mayor no es múltiplo de la
## longitud de uno menor
```

```
## [1] 0.3043478
```

```
# con K=7
k_7 <- knn(train = diabetes.train[,1:3],test = diabetes.test[,1:3],cl = diabetes.train[,4], k = 7)
confusionMatrix(k_7,factor(diabetes.test[,4]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical        8      0     0
##   normal          2     10     0
##   overt           0      0    10
##
## Overall Statistics
##
##                Accuracy : 0.9333
##                  95% CI : (0.7793, 0.9918)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 8.747e-12
##
##                   Kappa : 0.9
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   0.8000        1.0000       1.0000
## Specificity                   1.0000        0.9000       1.0000
## Pos Pred Value                1.0000        0.8333       1.0000
## Neg Pred Value                0.9091        1.0000       1.0000
## Prevalence                    0.3333        0.3333       0.3333
## Detection Rate                0.2667        0.3333       0.3333
## Detection Prevalence          0.2667        0.4000       0.3333
## Balanced Accuracy             0.9000        0.9500       1.0000
```

```
mean(k_7 == diabetes[,4])
```

```
## Warning in `==.default`(k_7, diabetes[, 4]): longitud de objeto mayor no es
## múltiplo de la longitud de uno menor

## Warning in `==.default`(k_7, diabetes[, 4]): longitud de objeto mayor no es
```

```
## múltiplo de la longitud de uno menor
```

```
## [1] 0.3043478
```

```
# con K=15
k_15 <- knn(train = diabetes.train[,1:3],test = diabetes.test[,1:3],cl = diabetes.train[,4], k = 15)
confusionMatrix(k_15,factor(diabetes.test[,4]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical        7      0     0
##   normal          3     10     0
##   overt           0      0    10
##
## Overall Statistics
##
##                Accuracy : 0.9
##                  95% CI : (0.7347, 0.9789)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 1.665e-10
##
##                   Kappa : 0.85
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   0.7000        1.0000       1.0000
## Specificity                   1.0000        0.8500       1.0000
## Pos Pred Value                1.0000        0.7692       1.0000
## Neg Pred Value                0.8696        1.0000       1.0000
## Prevalence                    0.3333        0.3333       0.3333
## Detection Rate                0.2333        0.3333       0.3333
## Detection Prevalence          0.2333        0.4333       0.3333
## Balanced Accuracy             0.8500        0.9250       1.0000
```

```
mean(k_15 == diabetes[,4])
```

```
## Warning in `==.default`(k_15, diabetes[, 4]): longitud de objeto mayor no es
## múltiplo de la longitud de uno menor

## Warning in `==.default`(k_15, diabetes[, 4]): longitud de objeto mayor no es
## múltiplo de la longitud de uno menor
```

```
## [1] 0.3130435
```

```
## Usando validación cruzada
set.seed(007)
mean(knn.cv(train = diabetes.train[,1:3],cl = diabetes.train[,4],k=1)==diabetes.train[,4])
```

```
## [1] 0.8869565
```

```
mean(knn.cv(train = diabetes.train[,1:3],cl = diabetes.train[,4],k=3)==diabetes.train[,4])
```

```
## [1] 0.9043478
```

```r
mean(knn.cv(train = diabetes.train[,1:3],cl = diabetes.train[,4],k=5)==diabetes.train[,4])
```

```
## [1] 0.8956522
```

```r
mean(knn.cv(train = diabetes.train[,1:3],cl = diabetes.train[,4],k=7)==diabetes.train[,4])
```

```
## [1] 0.8956522
```

```r
mean(knn.cv(train = diabetes.train[,1:3],cl = diabetes.train[,4],k=9)==diabetes.train[,4])
```

```
## [1] 0.8956522
```

```r
mean(knn.cv(train = diabetes.train[,1:3],cl = diabetes.train[,4],k=11)==diabetes.train[,4])
```

```
## [1] 0.8869565
```

```r
mean(knn.cv(train = diabetes.train[,1:3],cl = diabetes.train[,4],k=13)==diabetes.train[,4])
```

```
## [1] 0.8956522
```

```r
mean(knn.cv(train = diabetes.train[,1:3],cl = diabetes.train[,4],k=15)==diabetes.train[,4])
```

```
## [1] 0.8782609
```

```r
# como se evidencia que k=3 obtiene en promedio un mejor performance

k_3 <- knn(train = diabetes.train[,1:3],test = diabetes.test[,1:3],cl = diabetes.train[,4], k = 3)
confusionMatrix(k_3,factor(diabetes.test[,4]))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction chemical normal overt
##   chemical        8      0     0
##   normal          2     10     0
##   overt           0      0    10
##
## Overall Statistics
##
##                Accuracy : 0.9333
##                  95% CI : (0.7793, 0.9918)
##     No Information Rate : 0.3333
##     P-Value [Acc > NIR] : 8.747e-12
##
##                   Kappa : 0.9
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: chemical Class: normal Class: overt
## Sensitivity                   0.8000        1.0000       1.0000
## Specificity                   1.0000        0.9000       1.0000
## Pos Pred Value                1.0000        0.8333       1.0000
## Neg Pred Value                0.9091        1.0000       1.0000
## Prevalence                    0.3333        0.3333       0.3333
## Detection Rate                0.2667        0.3333       0.3333
## Detection Prevalence          0.2667        0.4000       0.3333
## Balanced Accuracy             0.9000        0.9500       1.0000
```

```r
# Obtener la pro. de votos
K_3_prob <- attr(k_3, "prob")

# Valores predichos
head(k_3)
```

```
## [1] chemical chemical chemical chemical normal   chemical
## Levels: chemical normal overt
```