

# **MINERIA de DATOS**

## **Clasificador Naïve Bayes**

Edgar Acuna

Departamento de Matematicas

Universidad de Puerto Rico en Mayaguez

# Construccion de un clasificador No Bayesiano estimando $f(x/C)$

- Asumir que se quiere predecir la variable  $Y$  que asume  $G$  valores distintos y que estos valores son  $v_1, v_2, \dots, v_G$ .
- Asumir que hay  $m$  atributos de entrada llamados  $X_1, X_2, \dots, X_m$
- Dividir el conjunto de datos en  $G$  subconjuntos de datos llamados  $DS_1, DS_2, \dots, DS_G$ .
- Definir  $DS_i =$  Registros en los cuales  $Y=v_i$
- En cada grupo  $DS_i$ , usamos estimacion de densidad para estimar la distribucion de las variables de entrada entre los registros. Es decir, la funcion de probabilidad conjunta por clase  $P(X_1, X_2, \dots, X_m \mid Y=v_i)$
- Idea 1. Suponga que se quiere predecir la clase  $Y$  a la cual pertenece el vector  $(X_1 = u_1, X_2 = u_2, \dots, X_m = u_m)$ . Se puede tomar como la clase  $Y=v_i$  a aquella para la cual  $P(X_1, X_2, \dots, X_m \mid Y=v_i)$  sea la mayor posible. Esto es,

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)$$

# Construccion de un clasificador Bayesiano estimando $f(x/c)$

- Asumir que se quiere predecir la variable  $Y$  que asume  $G$  valores distintos y que estos valores son  $v_1, v_2, \dots, v_G$ .
- Asumir que hay  $m$  atributos de entrada llamados  $X_1, X_2, \dots, X_m$
- Dividir el conjunto de datos en  $G$  subconjuntos de datos llamados  $DS_1, DS_2, \dots, DS_G$ .
- Definir  $DS_i =$  Registros en los cuales  $Y=v_i$
- En cada grupo  $DS_i$ , usamos estimacion de densidad para estimar la distribucion de las variables de entrada entre los registros  $Y=v_i$ . Es decir, la funcion de probabilidad conjunta por clase  $P(X_1, X_2, \dots, X_m \mid Y=v_i)$
- Idea 2: Para predecir la clase a la cual pertenece el nuevo vector de entradas  $(X_1 = u_1, X_2 = u_2, \dots, X_m = u_m)$  es mejor hallar la clase  $Y=v_i$  para la cual la probabilidad posterior  $P(Y=v_i \mid X_1, X_2, \dots, X_m)$  sea la mayor posible.

# Terminologia

- MLE (Estimador Maximo Verosimil):

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)$$

- MAP (Estimador Maximo a Posteriori):

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(Y = v \mid X_1 = u_1 \cdots X_m = u_m)$$

# Obteniendo la probabilidad posterior

$$\begin{aligned} & P(Y = v \mid X_1 = u_1 \cdots X_m = u_m) \\ = & \frac{P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)P(Y = v)}{P(X_1 = u_1 \cdots X_m = u_m)} \\ = & \frac{P(X_1 = u_1 \cdots X_m = u_m \mid Y = v)P(Y = v)}{\sum_{j=1}^G P(X_1 = u_1 \cdots X_m = u_m \mid Y = v_j)P(Y = v_j)} \end{aligned}$$

$$P(X_1 = u_1, \dots, X_n = u_n) = \sum_{v=1}^G P(X_1 = u_1, \dots, X_n = u_n, Y = v)$$

$$\sum_{v=1}^G P(X_1 = u_1, \dots, X_n = u_n \mid Y = v)P(Y = v)$$

# Estimacion de un Clasificador Bayesiano[1]

1. Estimar la distribucion de las predictoras en cada clase. Es decir, estimar  $P(X_1, X_2, \dots, X_m \mid Y=v_i)$ .
2. Estimar  $P(Y=v_i)$ . Como la fraccion de registros con  $Y=v_i$ .
3. Para hacer una nueva prediccion usar:

$$\begin{aligned} Y^{\text{predict}} &= \operatorname{argmax}_v P(Y = v \mid X_1 = u_1 \cdots X_m = u_m) \\ &= \operatorname{argmax}_v P(X_1 = u_1 \cdots X_m = u_m \mid Y = v) P(Y = v) \end{aligned}$$

# Estimacion de un clasificador Bayesiano[2]

1. Estimar la distribucion de las predictoras en cada clase. Es decir, estimar  $P(X_1, X_2, \dots, X_m / Y=v_i)$ .
3. Estimar  $P(Y=v_i)$ . Como la fraccion de registros con  $Y=v_i$ .
4. Para una nueva prediccion:

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(Y = v)$$
$$= \underset{v}{\operatorname{argmax}} P(X_1 = u_1 \cdots X_m = u_m / Y=v)$$

Podemos usar nuestro favorito estimador de densidad.

Tenemos dos opciones:

- Estimador de densidad conjunta (kernel, k-nn)
- Estinador Naïve de densidad

# Clasificador Naïve Bayes[1]

En el caso del clasificador naive se supone que las variables predictoras son independientes en cada una de las clases . Esto es,

$$P(X_1 = u_1 \cdots X_m = u_m | Y = v) = P(X_1 = u_1 / Y = v) \dots P(X_m = u_m / Y = v).$$

Luego,

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(X_1 = u_1 \cdots X_m = u_m | Y = v) P(Y = v)$$

Se convierte en:

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} P(Y = v) \prod_{j=1}^m P(X_j = u_j | Y = v)$$



# Clasificador Naïve Bayes[2]

Si hay muchos atributos de entrada este producto puede producir underflow, así que es mejor usar logaritmos.

$$Y^{\text{predict}} = \underset{v}{\operatorname{argmax}} \left( \log P(Y = v) + \sum_{j=1}^m \log P(X_j = u_j | Y = v) \right)$$

Si  $X_j$  es discreta entonces

$P(X_j = u_j | Y = v) = (\text{\#de records con } X_j = u_j \text{ en la clase } v) / \text{\# de records en la clase } v$ . Es decir, la frecuencia relativa.

# Clasificador Naïve Bayes[3]

El clasificador Naïve Bayes puede ser aplicado tambien cuando hay predictoras continuas, hay dos alternativas

- a) Aplicar previamente un metodo de discretizacion tal como: Usando intervalos de igual ancho, usando intervalos con igual frecuencia, ChiMerge, 1R, Discretizacion usando el metodo de la entropia. Todos ellos estan disponible en la libreria dprep (ver disc.mentr, disc.ew, disc.ef y chiMerge). Algunas veces discretizacion solo en dos intervalos es usada.
- b) Asumiendo una distribucion para cada predictora, por lo general esta se asume Gausiana, con media y varianza estimada de los datos. En forma mas general se puede estimar la distribucion usando un metodo tal como el kernel (muy pesado computacionalmente).

# Clasificador Naïve Bayes [4]

Así en que caso que  $X_j$  sea continua,

$$P[X_j = a_j / C_i] = \frac{1}{s_j \sqrt{2\pi}} \exp\left[-\frac{(a_j - \bar{x}_j)^2}{2s_j^2}\right]$$

Donde,  $\bar{x}_j$  y  $s_j$  son la media y la varianza de los valores de la variable  $X_j$  en la clase  $C_i$ .

La librería e1071 de R contiene una función **naiveBayes** que calcula el clasificador naïve Bayes, tanto para datos discretos como continuos.

Naïve Bayes también está disponible en Weka, Rapidminer, y en la librería scikit-learn de Python.

# Ejemplo 1.(atributos discretos solamente)

X1	X2	X3	Y
0	0	1	0
0	1	0	0
1	1	0	0
0	0	1	1
1	1	1	1
0	0	1	1
1	1	0	1

# Ejemplo 1. (Cont.)

$$P(Y = 0) = 3/7$$

$$P(Y = 1) = 4/7$$

A que clase sera asignada el registro (X1=0,X2=0,X3=1)?

$$P(X_1 = 0, X_2 = 0, X_3 = 1 / Y = 0) = P(X_1 = 0 / Y = 0)P(X_2 = 0 / Y = 0)$$

$$P(X_3 = 1 / Y = 0) = (2/3)(1/3)(1/3) = 2/27$$

$$P(X_1 = 0, X_2 = 0, X_3 = 1 / Y = 1) = P(X_1 = 0 / Y = 1)P(X_2 = 0 / Y = 1)$$

$$P(X_3 = 1 / Y = 1) = (2/4)(2/4)(3/4) = 3/16$$

Como  $(3/7)(2/27) < (4/7)(3/16)$  entonces (X1=0,X2=0,x3=1) sera asignado a la clase 1. Si el objeto esta asignado a la clase 0 entonces el NB comete un error.

## Ejemplo 2. (atributos discretos y continuos)

X1	X2	X3	X4	Y
0	0	1	3.15	0
0	1	0	8.17	0
1	1	0	5.72	0
0	0	1	7.16	1
1	1	1	9.32	1
0	0	1	12.81	1
1	1	0	15.48	1

## Ejemplo 2. (cont.)

#Metodo 1. Discretizando la columna 4

```
dnaiveeje2=disc.ew(naiveeje2,c(4:5))
```

```
dnaiveeje2
```

```
col1 col2 col3 col4 col5
```

```
[1,] 0 0 1 1 0
```

```
[2,] 0 1 0 1 0
```

```
[3,] 1 1 0 1 0
```

```
[4,] 0 0 1 1 1
```

```
[5,] 1 1 1 2 1
```

```
[6,] 0 0 1 2 1
```

```
[7,] 1 1 0 2 1
```

#Metodo 2. Sin discretizar la columna 4

Media y desviacion estandar de la col4 en cada clase

```
mean(naiveeje2[naiveeje2[,5]==0,4])
```

```
1] 5.68
```

```
> mean(naiveeje2[naiveeje2[,5]==1,4])
```

```
➤[1] 11.1925
```

## Ejemplo 2. (cont.)

```
> sd(naiveeje2[naiveeje2[,5]==0,4])
```

```
[1] 2.510239
```

```
> sd(naiveeje2[naiveeje2[,5]==1,4])
```

```
[1] 3.686293
```

```
> # a que clase sera asignado el vector(0,0,1,4.25)?
```

Hay que calcular

$$P[X_1=0/Y=0]P[X_2=0/Y=0]P[X_3=1/Y=0]f(X_4=4.25/Y=0)P[Y=0]$$

y compararla con

$$P[X_1=0/Y=1]P[X_2=0/Y=1]P[X_3=1/Y=1]f(X_4=4.25/Y=1)P[Y=1]$$

```
> (2/27)*pnorm(4.25,5.68,2.5102)*3/7
```

```
[1] 0.009030122
```

```
> (3/16)*pnorm(4.25,11.1925,3.6862)*4/7
```

```
[1] 0.003195506
```

Luego el vector sera asignado a la clase 0.

```
>#usando naivebayes de e1071
```

```
naivebayes21=as.data.frame(naiveeje2)
```

```
a=naiveBayes(col5~.,data=naivebayes21)
```

```
a
```



## Ejemplo 2. (cont.)

```
>.....  
>A-priori probabilities:  
>Y  
>    0    1  
>0.4285714 0.5714286  
  
>Conditional probabilities:  
> col1  
> Y    [,1]    [,2]  
> 0 0.3333333 0.5773503  
> 1 0.5000000 0.5773503  
>.....  
>col4  
>Y    [,1]    [,2]  
> 0 5.6800 2.510239  
> 1 11.1925 3.686293
```

## Ejemplo 2. (cont.)

```
pred=predict(a,naivebayes21[,-5],type="raw")
pred1=max.col(pred)
pred1
[1] 1 1 1 2 2 2 2
#matriz de confusion
table(pred1,naivebayes21[,5])
Pred1  0 1
      1 3 0
      2 0 4
Error=0
```

# Naive Bayes para Bupa

## Sin discretizar

```
> a=naiveBayes(V7~.,data=bupa)
> pred=predict(a,bupa[,-7],type="raw")
> pred1=max.col(pred)
> table(pred1,bupa[,7])
pred1  1  2
      1 112 119
      2  33  81
> error=152/345
[1] 0.4405797
```

## Discretizando con el metodo de la entropia

```
> dbupa=disc.mentr(bupa,1:7)
#convirtiendo cada variable en un factor de lo contrario no considerada que
#ha sido discretizada
>for (i in 1:7)
+dbupa[,i]=as.factor(dbupa[,i])
>b=naiveBayes(V7~.,data=dbupa)
>pred=predict(b,dbupa[,-7])
>error=sum(pred!=bupa[,7])/dim(dbupa)[1]
>error
[1] 0.3681159
```

# Naïve Bayes para Bupa (cont.)

## Discretizando por el metodo ChiMerge

```
chibupa=chiMerge(bupa,1:6)
for (i in 1:7)
  chibupa[,i]=as.factor(chibupa[,i])
b=naiveBayes(V7~.,data=chibupa)
pred=predict(b,chibupa[,-7])
error=sum(pred!=chibupa[,7])/dim(chibupa)[1]
error
[1] 0.1420290
```

## Discretizando usando intervalos de igual ancho

```
dbupa=disc.ew(bupa,1:6)
#convirtiendo cada variable en un factor
for (i in 1:7)
  dbupa[,i]=as.factor(dbupa[,i])
b=naiveBayes(V7~.,data=dbupa)
pred=predict(b,dbupa[,-7])
error=sum(pred!=dbupa[,7])/dim(dbupa)[1]
error
[1] 0.2608696
```

# Naïve Bayes en Rapidminer

The screenshot displays the RapidMiner software interface with a workflow designed for Naïve Bayes classification. The main process area contains the following steps:

- Retrieve bupa**: Retrieves data from the 'bupa' repository.
- Set Role**: Configures the data roles. The 'target role' is set to 'label'.
- Naive Bayes**: The core classification model.
- Apply Model**: Applies the Naive Bayes model to the data.
- Performance**: Evaluates the model's performance.

The left sidebar shows the 'Operators' panel with a search for 'perf' and a list of performance-related operators. The 'Repositories' panel lists various datasets, including 'bupa'. The right sidebar shows the 'Parameters' panel for the 'Set Role' operator, with a 'Compatibility level' indicator.

# Naïve Bayes para Diabetes

## Sin Descretizar

```
> a=naiveBayes(V9~.,data=diabetes)
> pred=predict(a,diabetes[,-9],type="raw")
> pred1=max.col(pred)
> table(pred1,diabetes[,9])
```

```
pred1  1  2
      1 421 104
      2  79 164
```

```
> error=(79+104)/768
[1] 0.2382813
```

## Discretizando por el metodo de la entropia

```
> ddiabetes=disc.mentr(diabetes,1:9)
> for (i in 1:9)
+ ddiabetes[,i]=as.factor(ddiabetes[,i])
> b=naiveBayes(V9~.,data=ddiabetes)
> pred=predict(b,ddiabetes[,-9])
> error=sum(pred!=ddiabetes[,9])/dim(ddiabetes)[1]
> error
[1] 0.2161458
```

# Naïve Bayes para Diabetes (cont.)

## Discretizando por el metodo ChiMerge

```
ddiabetes=chiMerge(diabetes,1:8)
for (i in 1:9)
+ ddiabetes[,i]=as.factor(ddiabetes[,i])
b=naiveBayes(V9~.,data=ddiabetes)
pred=predict(b,ddiabetes[,-9])
error=sum(pred!=ddiabetes[,9])/dim(ddiabetes)[1]
> error
[1] 0.09895833
```

## Discretizando por el metodo de intervalos de igual ancho

```
ddiab=disc.ew(diabetes,1:8)
ddiabetes=disc.ew(diabetes,1:8)
for (i in 1:9)
+ ddiabetes[,i]=as.factor(ddiabetes[,i])
b=naiveBayes(V9~.,data=ddiabetes)
pred=predict(b,ddiabetes[,-9])
error=sum(pred!=ddiabetes[,9])/dim(ddiabetes)[1]
> error
[1] 0.2083333
```

# Error por Validacion cruzada- Naïve Bayes

```
library (dprep)  
data(diabetes)  
crossval(diabetes,method="naiveBayes",repet=1)
```

The error estimation in each repetition are:

```
[1] 0.2513021
```

The mean error estimation by cross-validation using  
all the repetititons is:

```
[1] 0.2513021
```



# Validacion Cruzada en Rapidminer

The screenshot displays the RapidMiner Studio Basic 6.5.001 interface. The main workspace is divided into two panels: "Training" and "Testing".

**Training Panel:** A "Naive Bayes" model node is connected to a "Performance" node. The "Performance" node is configured with "mod" (model) and "thr" (threshold) as inputs.

**Testing Panel:** An "Apply Model" node is connected to the "Performance" node. The "Apply Model" node is configured with "mod" (model) and "thr" (threshold) as inputs. The "Performance" node is also connected to a "Write as Text" node, which is configured with "inp" (input) and "inp" (input) as inputs.

**Left Panel:** The "Operators" pane shows a list of operators categorized by "Utility (3)", "Files (3)", "Export (12)", "Data (3)", "Models (2)", and "Attributes (2)". The "Repositories" pane shows a list of data sources, including "arbol1", "breastw1", "bupa", "bw", "census", "censusna", "crimen", "eje1disc", "ionosphere", "magic", "vehicle", and "diabetes".

**Right Panel:** The "Parameters" pane shows the "Performance (Performance (Classi...))" node. The "main criteri..." dropdown is set to "first". The "accuracy" and "classification error" checkboxes are checked. The "kappa", "weighted mean recall", "weighted mean precis", "spearman rho", and "kendall tau" checkboxes are unchecked.

**Bottom Panel:** The "Help" pane shows the "Performance (Classi...)" node's synopsis.

# The Auto-mpg dataset

Donor: Quinlan, R. (1993)

Number of Instances: 398 minus 6 missing=392 (training: 196 test: 196):

Number of Attributes: 9 including the class attribute  
7. Attribute Information:

1. mpg: continuous (discretizado bad $\leq$ 25, good $>$ 25)
2. cylinders: multi-valued discrete
3. displacement: continuous (discretizado low $\leq$ 200, high $>$ 200)
4. horsepower: continuous ((discretizado low $\leq$ 90, high $>$ 90)
5. weight: continuous (discretizado low $\leq$ 3000, high $>$ 3000)
6. acceleration: continuous (discretizado low $\leq$ 15, high $>$ 15)
7. model year: multi-valued discrete (discretizado 70-74, 75-77, 78-82)
8. origin: multi-valued discrete
9. car name: string (unique for each instance)

Note: horsepower has 6 missing values

# The auto-mpg dataset

18.0 8 307.0 130.0 3504. 12.0 70 1 "chevrolet chevelle  
malibu"

15.0 8 350.0 165.0 3693. 11.5 70 1 "buick skylark 320"

18.0 8 318.0 150.0 3436. 11.0 70 1 "plymouth satellite"

16.0 8 304.0 150.0 3433. 12.0 70 1 "amc rebel sst"

17.0 8 302.0 140.0 3449. 10.5 70 1 "ford torino"

.....

27.0 4 140.0 86.00 2790. 15.6 82 1 "ford mustang gl"

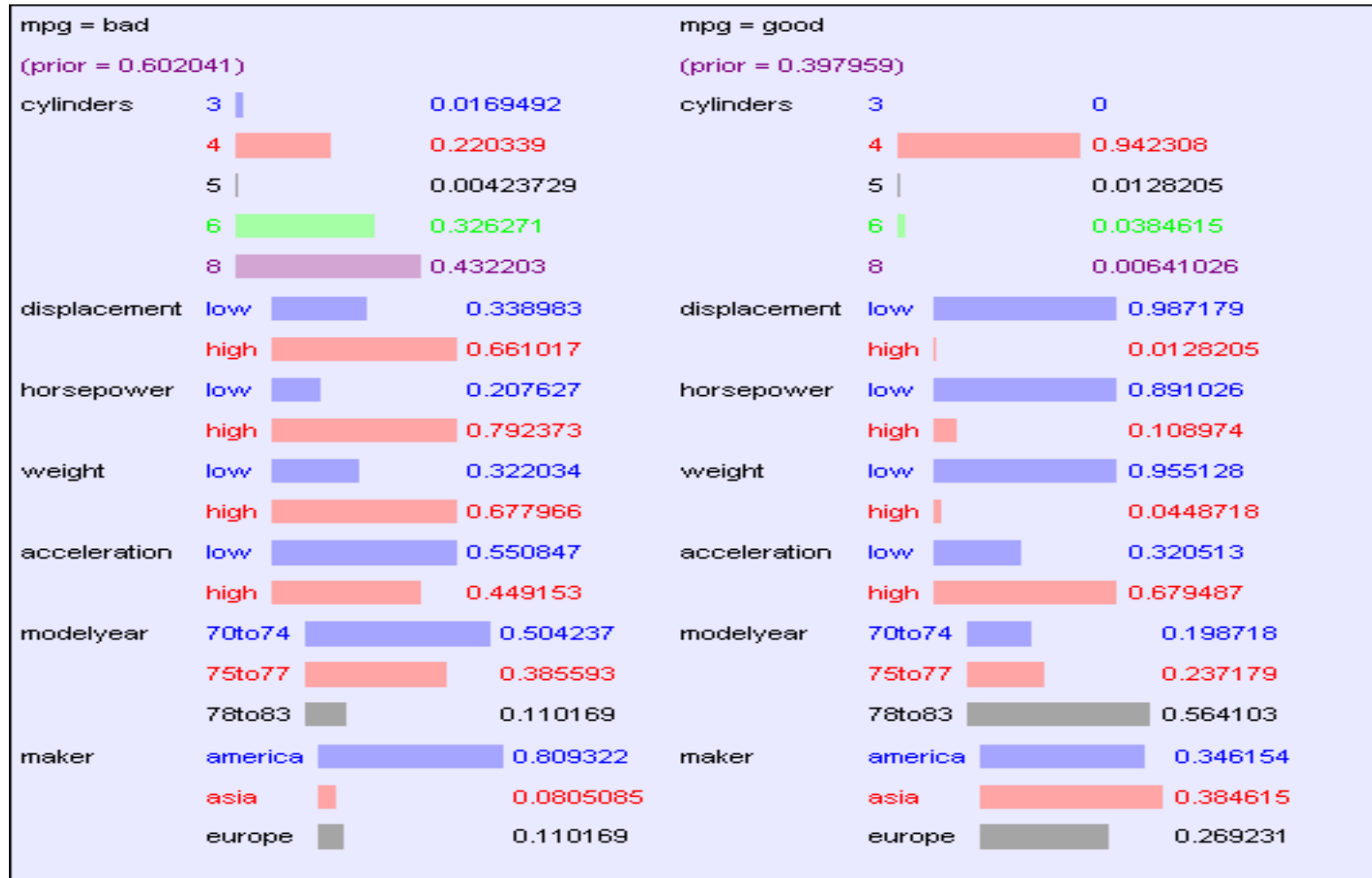
44.0 4 97.00 52.00 2130. 24.6 82 2 "vw pickup"

32.0 4 135.0 84.00 2295. 11.6 82 1 "dodge rampage"

28.0 4 120.0 79.00 2625. 18.6 82 1 "ford ranger"

31.0 4 119.0 82.00 2720. 19.4 82 1 "chevy s-10"

# Resultados del clasificador NB para “MPG”: 392 records



# Resultados del clasificador NB para: “mpg”

```
> #sin discretizar  
> b=naiveBayes(mpg~.,data=autompg)  
> pred=predict(b,autompg[,-1],type="raw")  
> pred1=max.col(pred)  
> table(pred1,autompg[,1])
```

```
pred1  1  2
```

```
  1 180  8
```

```
  2  56 148
```

```
> error=64/392
```

```
[1] 0.1632
```

```
> #Discretizando manualmente
```

```
> b=naiveBayes(mpg~.,data=autompg2)
```

```
> pred=predict(b,autompg2[,-1])
```

```
> table(pred,autompg2[,1])
```

```
pred  1  2
```

```
  1 182  7
```

```
  2  54 149
```

```
> 61/392
```

```
[1] 0.1556122
```

# Clasificadores Naive Bayes (cont.)

- Las probabilidades cero afectan al clasificador Naïve Bayes (ocurre si un valor de una variable en el conjunto de prueba no aparece en el conjunto de entrenamiento). Para resolver el problema se usa la correccion de Laplace  $(1+\text{conteo})/(k+n)$ .
- El proceso de discretizacion tambien parecer afectar el rendimiento del clasificador.
- Naïve Bayes es bastante barato. No tiene problemas para trabajar con 10,000 atributos.
- Naive Bayes puede aplicarse a conjunto de datos con distinto tipos de variables.
- Naïve Bayes es un caso particular de Redes bayesianas