

Árboles de Decisión

1. Introducción

- Los árboles de decisión son muy utilizados en la práctica
- A pesar de no siempre ser el método más preciso
- Esto debido a que son intuitivos y fáciles de explicar
- De hecho, se usan mucho en sector financiero y bancario
- La aplicación que veremos corresponde a dicho sector
- Con la crisis de 2007-2008 los controles a los créditos se volvieron más estrictos
- Los sistemas de préstamo se volvieron más sofisticados
- Se empezó a usar machine learning para predecir el riesgo de default
- En el sector financiero los árboles de decisión son populares porque suelen ser precisos y fáciles de comunicar
- Los gobiernos son cuidadosos de vigilar las decisiones crediticias de los bancos
- Por lo tanto, los ejecutivos deben ser capaces de explicar sus decisiones
- También es útil para explicarle a los clientes por qué les pueden haber negado un crédito
- Desarrollaremos un modelo simple de aprobación automatizada de créditos usando el algoritmo C5.0

2. Datos

- La idea del modelo es identificar los factores que predicen una alta probabilidad de que un cliente haga default
- Usaremos datos con información de préstamos pasados, identificando si dichos préstamos tuvieron default o no
- También contamos con un buen conjunto de información sobre los prestatarios
- Trabajaremos con el archivo `credit.csv`, que contiene información de 1000 préstamos en Alemania con su respectiva indicación de si el crédito se fue a default o no
- Para comenzar, no olviden determinar el directorio de trabajo

```
credit <- read.csv("credit.csv")
```

- Nótese que en este caso no usamos el parámetro `stringsAsFactors=FALSE`
- La razón es que en este ejemplo sí nos conviene transformar las variables de texto en factores
- Démosle un vistazo a la base:

```
str(credit)
```

```
## 'data.frame': 1000 obs. of 21 variables:
## $ checking_balance : Factor w/ 4 levels "1 - 200 DM", "< 0 DM", ...: 2 1 4 2 2 4 4 1 4 1 ...
## $ months_loan_duration: int 6 48 12 42 24 36 24 36 12 30 ...
## $ credit_history : Factor w/ 5 levels "critical", "delayed", ...: 1 5 1 5 2 5 5 5 1 ...
## $ purpose : Factor w/ 10 levels "business", "car (new)", ...: 8 8 5 6 2 5 6 3 8 2 ...
## $ amount : int 1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
## $ savings_balance : Factor w/ 5 levels "101 - 500 DM", ...: 5 3 3 3 3 5 2 3 4 3 ...
## $ employment_length : Factor w/ 5 levels "0 - 1 yrs", "1 - 4 yrs", ...: 4 2 3 3 2 2 4 2 3 5 ...
## $ installment_rate : int 4 2 2 2 3 2 3 2 2 4 ...
## $ personal_status : Factor w/ 4 levels "divorced male", ...: 4 2 4 4 4 4 4 4 1 3 ...
## $ other_debtors : Factor w/ 3 levels "co-applicant", ...: 3 3 3 2 3 3 3 3 3 3 ...
## $ residence_history : int 4 2 3 4 4 4 4 2 4 2 ...
## $ property : Factor w/ 4 levels "building society savings", ...: 3 3 3 1 4 4 1 2 3 2 ...
## $ age : int 67 22 49 45 53 35 53 35 61 28 ...
## $ installment_plan : Factor w/ 3 levels "bank", "none", ...: 2 2 2 2 2 2 2 2 2 2 ...
## $ housing : Factor w/ 3 levels "for free", "own", ...: 2 2 2 1 1 1 2 3 2 2 ...
## $ existing_credits : int 2 1 1 1 2 1 1 1 1 2 ...
## $ job : Factor w/ 4 levels "mangement self-employed", ...: 2 2 4 2 2 4 2 1 4 1 ...
## $ dependents : int 1 1 2 2 2 2 1 1 1 1 ...
## $ telephone : Factor w/ 2 levels "none", "yes": 2 1 1 1 1 2 1 2 1 1 ...
## $ foreign_worker : Factor w/ 2 levels "no", "yes": 2 2 2 2 2 2 2 2 2 2 ...
## $ default : int 1 2 1 1 2 1 1 1 1 2 ...
```

- default es el outcome que pronosticaremos. El problema es que es una variable **integer**, pero la necesitaremos como factor (no se transformó porque no era un string)

```
credit$default <- factor(credit$default)
```

- Tenemos 1000 observaciones (créditos) y 21 variables
- Hay tanto características del crédito como del prestatario
- Por ejemplo, el valor del préstamo:

```
summary(credit$amount)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      250   1366   2320   3271   3972   18420
```

- O la duración:

```
summary(credit$months_loan_duration)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      4.0   12.0   18.0   20.9   24.0   72.0
```

- El balance en cuenta corriente:

```
table(credit$checking_balance)
```

```
##
## 1 - 200 DM    < 0 DM    > 200 DM    unknown
##      269      274      63      394
```

- Los datos están en marcos alemanes (DM)
- O el balance en cuenta de ahorros

```
table(credit$savings_balance)
```

```
##
```

```
## 101 - 500 DM 501 - 1000 DM      < 100 DM      > 1000 DM      unknown
##           103           63           603           48           183
```

- Muy probablemente estas variables son buenos predictores de si alguien hace default o no

3. Bases de entrenamiento y prueba

- Anteriormente hemos partido fácilmente las bases entre entrenamiento y prueba
- Esto porque los datos ya venían en un orden aleatorio
- Pero, qué hacer si vienen en un orden predeterminado?
- Por ejemplo, podrían estar ordenados según el valor del crédito
- Para ello tomamos una muestra aleatoria de los préstamos para entrenamiento y el resto para prueba
- Usaremos el 90% de los datos para entrenamiento y el restante 10% para prueba
- Para poder replicar los resultados más adelante, debemos fijar una semilla con la función `set.seed()`:

```
set.seed(123)
```

- Ahora generemos un vector de 900 números aleatorios, de la secuencia de números enteros de 1 a 1000:

```
train_sample <- sample(1000, 900)
```

- Miremos el vector:

```
str(train_sample)
```

```
## int [1:900] 288 788 409 881 937 46 525 887 548 453 ...
```

- Tenemos un vector con 900 enteros aleatorios
- Lo que haremos es quedarnos con las filas de `credit` que corresponden a estos 900 números. Esta será nuestra base de entrenamiento
- Naturalmente, las filas que no corresponden a estos 900 números serán la base de prueba

```
credit_train <- credit[train_sample, ]
credit_test  <- credit[-train_sample, ]
```

- Nótese que `credit_train` es una base con 900 observaciones y 21 variables, mientras que `credit_test` es una base con 100 observaciones y 21 variables

```
str(credit_train)
```

```
## 'data.frame': 900 obs. of 21 variables:
## $ checking_balance : Factor w/ 4 levels "1 - 200 DM", "< 0 DM", ...: 1 4 4 4 3 4 1 1 4 4 ...
## $ months_loan_duration: int 48 48 24 24 9 11 18 24 24 12 ...
## $ credit_history : Factor w/ 5 levels "critical", "delayed", ...: 2 1 5 5 5 1 5 1 5 3 ...
## $ purpose : Factor w/ 10 levels "business", "car (new)", ...: 7 3 8 3 8 2 8 1 8 6 ...
## $ amount : int 7582 2751 3235 7814 745 1393 1113 2825 1552 2759 ...
## $ savings_balance : Factor w/ 5 levels "101 - 500 DM", ...: 1 5 2 3 3 3 3 5 3 3 ...
## $ employment_length : Factor w/ 5 levels "0 - 1 yrs", "1 - 4 yrs", ...: 5 4 4 3 2 1 2 3 3 4 ...
## $ installment_rate : int 2 4 3 3 3 4 4 4 3 2 ...
## $ personal_status : Factor w/ 4 levels "divorced male", ...: 4 4 1 4 2 2 2 4 4 4 ...
## $ other_debtors : Factor w/ 3 levels "co-applicant", ...: 3 3 3 3 3 3 2 3 3 3 ...
## $ residence_history : int 4 3 2 3 2 4 4 3 1 4 ...
## $ property : Factor w/ 4 levels "building society savings", ...: 4 2 2 2 3 2 3 4 2 1 ...
```

```
## $ age : int 31 38 26 38 28 35 26 34 32 34 ...
## $ installment_plan : Factor w/ 3 levels "bank","none",...: 2 2 2 2 2 2 2 1 2 ...
## $ housing : Factor w/ 3 levels "for free","own",...: 1 2 2 2 2 2 2 2 2 ...
## $ existing_credits : int 1 2 1 1 1 2 1 2 1 2 ...
## $ job : Factor w/ 4 levels "mangement self-employed",...: 1 2 1 1 4 1 4 2 2 2 ...
## $ dependents : int 1 2 1 1 1 1 2 2 2 1 ...
## $ telephone : Factor w/ 2 levels "none","yes": 2 2 2 2 1 1 1 2 1 1 ...
## $ foreign_worker : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
## $ default : Factor w/ 2 levels "1","2": 1 1 1 1 2 1 1 1 1 1 ...
```

```
str(credit_test)
```

```
## 'data.frame': 100 obs. of 21 variables:
## $ checking_balance : Factor w/ 4 levels "1 - 200 DM","< 0 DM",...: 4 3 2 1 4 1 4 4 1 2 ...
## $ months_loan_duration: int 6 12 12 54 24 9 12 18 45 24 ...
## $ credit_history : Factor w/ 5 levels "critical","delayed",...: 3 4 1 3 5 1 5 4 5 5 ...
## $ purpose : Factor w/ 10 levels "business","car (new)",...: 8 8 3 1 2 6 3 2 8 6 ...
## $ amount : int 426 409 1526 15945 1469 1919 2445 6458 3031 3021 ...
## $ savings_balance : Factor w/ 5 levels "101 - 500 DM",...: 3 4 3 3 1 3 5 3 1 3 ...
## $ employment_length : Factor w/ 5 levels "0 - 1 yrs","1 - 4 yrs",...: 4 2 4 1 4 3 1 4 2 2 ...
## $ installment_rate : int 4 3 4 3 4 4 2 2 4 2 ...
## $ personal_status : Factor w/ 4 levels "divorced male",...: 3 2 4 4 3 4 3 4 4 1 ...
## $ other_debtors : Factor w/ 3 levels "co-applicant",...: 3 3 3 3 3 3 3 3 2 3 ...
## $ residence_history : int 4 3 4 4 4 3 4 4 4 2 ...
## $ property : Factor w/ 4 levels "building society savings",...: 2 3 4 4 3 2 2 4 1 3 ...
## $ age : int 39 42 66 58 41 35 26 39 21 24 ...
## $ installment_plan : Factor w/ 3 levels "bank","none",...: 2 2 2 2 2 2 2 1 2 2 ...
## $ housing : Factor w/ 3 levels "for free","own",...: 2 3 1 3 3 3 3 2 3 3 ...
## $ existing_credits : int 1 2 2 1 1 1 1 2 1 1 ...
## $ job : Factor w/ 4 levels "mangement self-employed",...: 4 2 1 2 4 2 2 1 2 4 ...
## $ dependents : int 1 1 1 1 1 1 1 2 1 1 ...
## $ telephone : Factor w/ 2 levels "none","yes": 1 1 1 2 1 2 2 2 1 1 ...
## $ foreign_worker : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
## $ default : Factor w/ 2 levels "1","2": 1 1 1 2 1 1 1 2 2 1 ...
```

- Podemos verificar que la proporción de default es similar en ambas bases:

```
prop.table(table(credit_train$default))
```

```
##
##      1      2
## 0.7033333 0.2966667
```

```
prop.table(table(credit_test$default))
```

```
##
##      1      2
## 0.67 0.33
```

- Que de hecho va en concordancia con la proporción de defaults en toda la base:

```
prop.table(table(credit$default))
```

```
##
##      1      2
## 0.7 0.3
```

- 3 de cada 10 créditos terminan en default

3. Entrenamiento del modelo

- Para entrenar el modelo y predecir si un préstamo termina en default o no, usaremos el paquete C50 en R

```
library(C50)
```

- El paquete ofrece dos funciones: una para entrenar el modelo y otra para predecir
- El entrenamiento lo hacemos con la función `c5.0`, que tiene la siguiente sintaxis:

```
m <- C5.0(train, class, trials, costs)
```

- Donde `train` es la base de entrenamiento
- `class` es el vector de clases de los objetos
- `trials` es un número opcional de *boosting iterations*. Sobre esto hablaremos más adelante. El default es `trials=1`
- `costs` es una matriz opcional que especifica los costos relativos de diferentes tipos de errores
- En nuestra base `credit_train`, la columna 21 corresponde a la variable `default`, que es la que queremos predecir
- Entonces para entrenar el modelo, usamos el data frame sin dicha columna
- Y de hecho el vector asociado a dicha columna es el vector `class` en el algoritmo. Creémoslo:

```
credit_model <- C5.0(credit_train[-21], credit_train$default)
```

- Hemos creado un árbol de decisión con base en el algoritmo C5.0
- Podemos ver algunas de sus características:

```
credit_model
```

```
##
## Call:
## C5.0.default(x = credit_train[-21], y = credit_train$default)
##
## Classification Tree
## Number of samples: 900
## Number of predictors: 20
##
## Tree size: 54
##
## Non-standard options: attempt to group attributes
```

- Es un árbol basado en 900 ejemplo y 20 características. En total tiene 57 decisiones
- Podemos ver las decisiones con la función `summary()`

```
summary(credit_model)
```

```
##
## Call:
## C5.0.default(x = credit_train[-21], y = credit_train$default)
##
##
## C5.0 [Release 2.07 GPL Edition]      Tue Mar  7 13:06:52 2017
## -----
##
```

```

## Class specified by attribute `outcome'
##
## Read 900 cases (21 attributes) from undefined.data
##
## Decision tree:
##
## checking_balance in {> 200 DM,unknown}: 1 (412/50)
## checking_balance in {1 - 200 DM,< 0 DM}:
## :...other_debtors = guarantor:
## :   ...months_loan_duration > 36: 2 (4/1)
## :   :   months_loan_duration <= 36:
## :   :   :...installment_plan in {none,stores}: 1 (24)
## :   :   :   installment_plan = bank:
## :   :   :   :...purpose = car (new): 2 (3)
## :   :   :   :   purpose in {business,car (used),domestic appliances,education,
## :   :   :   :   :   furniture,others,radio/tv,repairs,
## :   :   :   :   :   retraining}: 1 (7/1)
## other_debtors in {co-applicant,none}:
## :...credit_history = critical: 1 (102/30)
## :   credit_history = fully repaid: 2 (27/6)
## :   credit_history = fully repaid this bank:
## :   :...other_debtors = co-applicant: 1 (2)
## :   :   other_debtors = none: 2 (26/8)
## :   credit_history in {delayed,repaid}:
## :   :...savings_balance in {501 - 1000 DM,> 1000 DM}: 1 (19/3)
## :   :   savings_balance = 101 - 500 DM:
## :   :   :...other_debtors = co-applicant: 2 (3)
## :   :   :   other_debtors = none:
## :   :   :   :...personal_status in {divorced male,
## :   :   :   :   :   married male}: 2 (6/1)
## :   :   :   :   personal_status = female:
## :   :   :   :   :...installment_rate <= 3: 1 (4/1)
## :   :   :   :   :   installment_rate > 3: 2 (4)
## :   :   :   :   personal_status = single male:
## :   :   :   :   :...age <= 41: 1 (15/2)
## :   :   :   :   :   age > 41: 2 (2)
## :   savings_balance = unknown:
## :   :...credit_history = delayed: 1 (8)
## :   :   credit_history = repaid:
## :   :   :...foreign_worker = no: 1 (2)
## :   :   :   foreign_worker = yes:
## :   :   :   :...checking_balance = < 0 DM:
## :   :   :   :   :...telephone = none: 2 (11/2)
## :   :   :   :   :   telephone = yes:
## :   :   :   :   :   :...amount <= 5045: 1 (5/1)
## :   :   :   :   :   :   amount > 5045: 2 (2)
## :   :   :   :   checking_balance = 1 - 200 DM:
## :   :   :   :   :...residence_history > 3: 1 (9)
## :   :   :   :   :   residence_history <= 3: [S1]
## savings_balance = < 100 DM:
## :...months_loan_duration > 39:
## :   :...residence_history <= 1: 1 (2)
## :   :   residence_history > 1: 2 (19/1)
## :   months_loan_duration <= 39:

```

```

##           :...purpose in {car (new),retraining}: 2 (47/16)
##           purpose in {domestic appliances,others}: 1 (3)
##           purpose = car (used):
##           :...amount <= 8086: 1 (9/1)
##           :   amount > 8086: 2 (5)
##           purpose = education:
##           :...checking_balance = 1 - 200 DM: 1 (2)
##           :   checking_balance = < 0 DM: 2 (5)
##           purpose = repairs:
##           :...residence_history <= 3: 2 (4/1)
##           :   residence_history > 3: 1 (3)
##           purpose = business:
##           :...credit_history = delayed: 2 (2)
##           :   credit_history = repaid:
##           :     :...age <= 34: 1 (5)
##           :       age > 34: 2 (2)
##           purpose = radio/tv:
##           :...employment_length in {0 - 1 yrs,
##           :   :                               unemployed}: 2 (14/5)
##           :   employment_length = 4 - 7 yrs: 1 (3)
##           :   employment_length = > 7 yrs:
##           :     :...amount <= 932: 2 (2)
##           :       :   amount > 932: 1 (7)
##           :   employment_length = 1 - 4 yrs:
##           :     :...months_loan_duration <= 15: 1 (6)
##           :       :   months_loan_duration > 15:
##           :         :...amount <= 3275: 2 (7)
##           :           amount > 3275: 1 (2)
##           purpose = furniture:
##           :...residence_history <= 1: 1 (8/1)
##           :   residence_history > 1:
##           :     :...installment_plan in {bank,stores}: 1 (3/1)
##           :       installment_plan = none:
##           :         :...telephone = yes: 2 (7/1)
##           :           telephone = none:
##           :             :...months_loan_duration > 27: 2 (3)
##           :               months_loan_duration <= 27: [S2]
##
## SubTree [S1]
##
## property in {building society savings,unknown/none}: 2 (4)
## property = other: 1 (6)
## property = real estate:
## :...job = skilled employee: 2 (2)
##   job in {mangement self-employed,unemployed non-resident,
##   unskilled resident}: 1 (2)
##
## SubTree [S2]
##
## checking_balance = 1 - 200 DM: 2 (5/2)
## checking_balance = < 0 DM:
## :...property in {building society savings,real estate,unknown/none}: 1 (8)
##   property = other:
##   :...installment_rate <= 1: 1 (2)

```

```

##      installment_rate > 1: 2 (4)
##
##
## Evaluation on training data (900 cases):
##
##      Decision Tree
##      -----
##      Size      Errors
##
##      54  135(15.0%)  <<
##
##
##      (a)  (b)  <-classified as
##      ----  ----
##      589   44   (a): class 1
##      91   176   (b): class 2
##
##
## Attribute usage:
##
## 100.00% checking_balance
##  54.22% other_debtors
##  50.00% credit_history
##  32.56% savings_balance
##  25.22% months_loan_duration
##  19.78% purpose
##  10.11% residence_history
##   7.33% installment_plan
##   5.22% telephone
##   4.78% foreign_worker
##   4.56% employment_length
##   4.33% amount
##   3.44% personal_status
##   3.11% property
##   2.67% age
##   1.56% installment_rate
##   0.44% job
##
##
## Time: 0.0 secs

```

- La primera parte de este output muestra las ramas del árbol y las predicciones asociadas
- En la segunda parte tenemos una evaluación del modelo, sobre los datos de entrenamiento
- La precisión es del 85%. Tenemos 135/900 errores
- Hay un total de 44 falsos positivos (no's clasificados como si's). Un tasa de 4.8%
- Y un total de 91 falsos negativos (si's clasificados como no's). Una tasa de 10%
- ¿Qué es más grave en este caso? Probablemente los falsos negativos, porque son préstamos que terminan en default pero el algoritmo predice que no. Es grave para los bancos esto
- Sin embargo estas no son las tasas que importan
- Los árboles tienden a sobre-estimar los datos de entrenamiento

- Lo que vale es qué tan bien pronostica los datos de prueba

4. Evaluación del desempeño del modelo

- Debemos ahora predecir la clase para cada ejemplo en la base de prueba
- Recordemos la lógica: vemos en que nodo terminal cae cada ejemplo
- Y le asignamos la categoría de la mayoría de casos de ese nodo
- Para hacer esto, usamos la función `predict()`, del paquete `C5.0`, que tiene la siguiente sintaxis

```
p <- predict(m, test, type)
```

- Donde `m` es el objeto resultante de la función `C5.0`
- `test` es la base de prueba
- `type` es el tipo de predicción que queremos: la clase o la probabilidad. El default es `type=class`
- Implementemos para el modelo que ya tenemos:

```
credit_pred <- predict(credit_model, credit_test)
```

- Y hacemos nuestras acostumbradas tablas con `gmodels` para evaluar el desempeño

```
library(gmodels)
```

```
CrossTable(credit_test$default, credit_pred, prop.chisq=FALSE, prop.c=FALSE, prop.r=FALSE, dnn=c("default", "prediccion"))
```

```
##
##
##      Cell Contents
## |-----|
## |                      N |
## |      N / Table Total |
## |-----|
##
##
## Total Observations in Table:  100
##
##
##      | default pronosticado
## default real |          1 |          2 | Row Total |
## -----|-----|-----|-----|
##          1 |          60 |          7 |          67 |
##          |          0.600 |          0.070 |          |
## -----|-----|-----|-----|
##          2 |          19 |          14 |          33 |
##          |          0.190 |          0.140 |          |
## -----|-----|-----|-----|
## Column Total |          79 |          21 |          100 |
## -----|-----|-----|-----|
##
##
```

- El modelo pronostica incorrectamente en 26 casos. Una precisión total del 74%
- Una tasa de falsos positivos de 7%

- Y de falsos negativos de 19%
- Bastante alto!
- Profundizaremos en cómo mejorar este desempeño tan poco satisfactorio