Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Máquinas de aprendizaje

Reporte: Otro ejemplo de reglas de asociación

BUAP

Docente: Abraham Sánchez López

Alumno	Matrícula
Taisen Romero Bañuelos	202055209

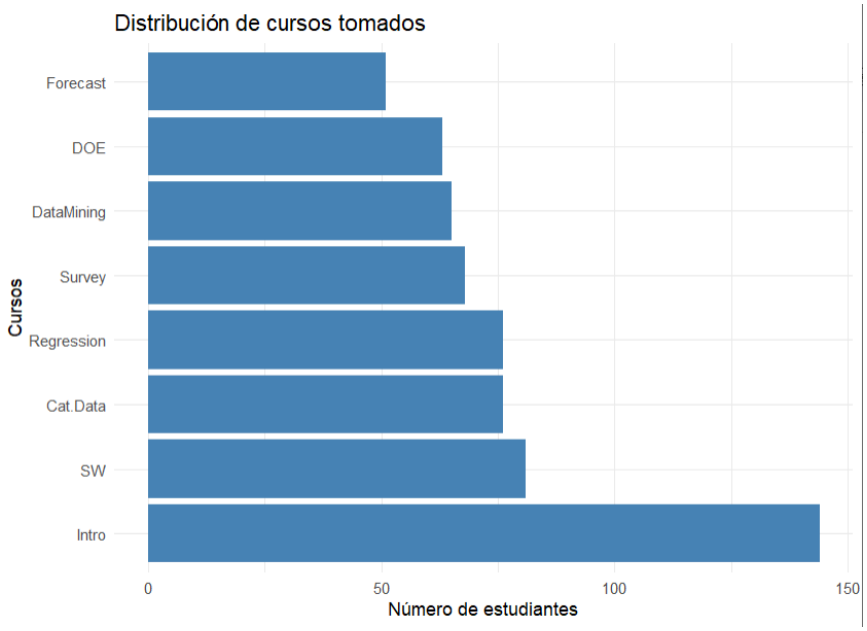
Identificación de combinaciones de cursos

El dataset es el mismo que el que usamos en la actividad 15 (Tarea de reglas de clasificación), así que voy a recuperar los elementos más importantes del EDA.

Los datos son binarios (indican si se compró el curso o no) por lo que un histograma y otros ploteos típicos del EDA será mejor omitirlos.

```
> str(data)
'data.frame':   365 obs. of  8 variables:
 $ Intro     : int  1 0 0 1 1 0 1 0 1 0 ...
 $ DataMining: int  1 0 1 0 1 1 0 0 0 0 ...
 $ Survey    : int  0 1 0 0 0 0 0 0 0 0 ...
 $ Cat.Data  : int  0 0 1 0 0 0 0 1 0 1 ...
 $ Regression: int  0 0 1 0 0 0 0 0 0 0 ...
 $ Forecast  : int  0 0 0 0 0 0 0 1 0 0 ...
 $ DOE       : int  0 0 0 0 0 0 0 1 0 0 ...
 $ SW        : int  0 0 1 0 0 0 0 1 0 0 ...
```

Como no podemos hacer un histograma con la función hist() por el tipo de datos que manejamos hagamos en su lugar un conteo de los cursos tomados para ver su distribución general. Como podemos ver, Intro está al frente, por lo que podemos estar seguro que el principio a priori de las reglas va a darle mucha atención a las combinaciones con Intro.



Podemos empezar a identificar correlaciones entre cursos si hacemos una matriz de correlación. Aquí, al igual que en el plot anterior, podemos empezar a inferir el tipo de reglas que obtendremos con el principio a priori. Si nos damos cuenta, Forecast tiene varias correlaciones positivas fuertes, lo que me hace pensar que veremos reglas con Forecast. Sobre Intro, tenemos correlaciones negativas, lo que me hace cuestionarme

cómo afectará a la creación de reglas, no sé si al ser negativa la correlación afecte negativamente a la creación de las reglas.

```
> cor(data)
                Intro   DataMining       Survey    Cat.Data  Regression    Forecast          DOE           SW
Intro       1.00000000 -0.08269372 -0.06950177 -0.05499603 -0.05499603 -0.01811741 -0.11651358  0.04106152
DataMining -0.08269372  1.00000000 -0.02040822  0.07876161  0.04348801  0.12223245 -0.04205271 -0.00731834
Survey     -0.06950177 -0.02040822  1.00000000  0.15321827 -0.08940502  0.09130416  0.00489739  0.04927086
Cat.Data   -0.05499603  0.07876161  0.15321827  1.00000000  0.06938627  0.10472150  0.06931710 -0.03781377
Regression -0.05499603  0.04348801 -0.08940502  0.06938627  1.00000000  0.06579752 -0.03781377  0.05089424
Forecast   -0.01811741  0.12223245  0.09130416  0.10472150  0.06579752  1.00000000  0.02503559  0.03199019
DOE        -0.11651358 -0.04205271  0.00489739  0.06931710 -0.03781377  0.02503559  1.00000000  0.12246266
SW          0.04106152 -0.00731834  0.04927086  0.09960857  0.05089424  0.03199019  0.12246266  1.00000000
```

También podemos identificar las combinaciones típicas de los cursos si hacemos una matriz de co-ocurrencia, esta matriz será fundamental para que en un futuro corroboremos los resultados obtenidos con las reglas de asociación.

```
> #Matriz de co-ocurrencia (Para estudiar qué cursos aparecen con mayor frecuencia juntos en un mismo estudiante)
> co_occurrence <- t(data) %*% as.matrix(data)
> co_occurrence
           Intro DataMining Survey Cat.Data Regression Forecast DOE SW
Intro        144         20     22       26         26       19  17 35
DataMining    20         65     11       18         16       15   9 14
Survey        22         11     68       23          9       14  12 18
Cat.Data      26         18     23       76         20       16  17 23
Regression    26         16      9       20         76       14  11 20
Forecast      19         15     14       16         14       51  10 13
DOE           17          9     12       17         11       10  63 21
SW            35         14     18       23         20       13  21 81
```

Técnicamente con esta matriz ya podemos ver las reglas que se crearán con las reglas. Aunque la correlación de Intro fue negativa para casi todos los cursos, en promedio tiene más combinaciones con otros cursos que cualquier otro. Esto podría leerse como que “Si no se compra un curso de Intro es menos probable que compren cualquier otro curso”, a excepción de SW que era el único con una correlación positiva con Intro.

Para obtener reglas más legibles cambiaré los valores binarios (1,0) por (Yes, No).

```
> head(data)
  Intro DataMining Survey Cat.Data Regression Forecast DOE  SW
1   Yes        Yes     No       No         No       No  No  No
2    No         No    Yes       No         No       No  No  No
3    No        Yes     No      Yes        Yes       No  No Yes
4   Yes         No     No       No         No       No  No  No
5   Yes        Yes     No       No         No       No  No  No
6    No        Yes     No       No         No       No  No  No
```

Para convertir el data frame en un formato de transacción usé la función as(). De manera similar a as.factor(), sólo que con algunas variaciones.

```
> #Convertir los datos a formato de transacciones
> data_trans <- as(data, "transactions")
> head(data_trans)
transactions in sparse format with
 6 transactions (rows) and
 16 items (columns)
```

Ahora ya podemos pasar a entrenar nuestro modelo de reglas de asociación.

```
> summary(rules)
set of 4169 rules

rule length distribution (lhs + rhs):sizes
   2    3    4    5    6    7    8
 106  489 1083 1231  879  325   56

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.000   4.000   5.000   4.836   6.000   8.000

summary of quality measures:
    support         confidence        coverage            lift            count
 Min.   :0.05205   Min.   :0.6000   Min.   :0.05479   Min.   :0.7252   Min.   : 19.00
 1st Qu.:0.07945   1st Qu.:0.7937   1st Qu.:0.09041   1st Qu.:0.9951   1st Qu.: 29.00
 Median :0.09589   Median :0.8571   Median :0.11233   Median :1.0665   Median : 35.00
 Mean   :0.15887   Mean   :0.8494   Mean   :0.19069   Mean   :1.0889   Mean   : 57.99
 3rd Qu.:0.22192   3rd Qu.:0.9130   3rd Qu.:0.24658   3rd Qu.:1.1489   3rd Qu.: 81.00
 Max.   :0.72329   Max.   :1.0000   Max.   :0.86027   Max.   :4.5647   Max.   :264.00

mining info:
       data ntransactions support confidence                                                                  call
 data_trans           365    0.05        0.6 apriori(data = data_trans, parameter = list(supp = 0.05, conf = 0.6, minlen = 2))
```

Bien, tenemos 4169 reglas, esto podría significar que estamos siendo permisivos con los criterios de generación de reglas, sin embargo, tengamos 100 o 200 reglas, si hacemos un inspect podemos ver las mejores reglas, y supongo que como son las mejores reglas van a aparecer siempre, se generen 100 reglas o 200. Esto lo deduzco en parte porque en la actividad pasada se comparó Eclat con Apriori y se demostró que pese a que son metodologías diferentes las reglas importantes siempre serán las mismas, por así decirlo, “La verdad es la verdad, la diga el Quijote o su porquero”. También podemos notar que la mayoría de las reglas tienen entre 4 y 5 elementos (se nota en la distribución del tamaño de las reglas), esto podría ser una señal de que quizá sí hay que ser más estricto con la generación de reglas ya que es extraño que una persona compre 5 cursos del jalón. Luego, en la confianza se nos dice que mínimo, todas las reglas tienen un 60% de confianza, y también que la regla más fuerte aparece en el 72% de los casos (en support). En promedio, la confianza de las reglas es del 85%, pero creo que habrá que tomar este dato con pinzas, en un momento lo explicaré.

Además, podemos decir que según el soporte mínimo absoluto, cualquier regla debe aparecer en al menos 18 de las 365 transacciones.

```
> rules <- apriori(data_trans,parameter = list(supp = 0.05, conf = 0.6, minlen = 2))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.6    0.1    1 none FALSE            TRUE       5    0.05      2     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 18
```

Si hacemos un filtrado para sólo seleccionar las reglas que tengan un lift > 1.2 y una confianza de al menos el 70% estamos reduciendo el número de reglas a 55.

```
> rules_filtered
set of 555 rules
```

Esto debería bastar para poder quedarnos sólo con aquellas reglas que nos sean de ayuda, sin embargo hay un problema que sólo se ve si mostramos muchas reglas y no sólo las 10 mejores.

```
> #Filtrado
> rules_filtered <- subset(rules, lift > 1.2 & confidence > 0.7)
> inspect(rules_filtered[1:100])
     lhs                                                rhs             support    confidence coverage   lift     count
[1]  {DOE=Yes}                                       => {Intro=No}      0.12602740 0.7301587  0.17260274 1.205918 46
[2]  {DOE=Yes, SW=No}                                => {Intro=No}      0.09863014 0.8571429  0.11506849 1.415643 36
[3]  {Intro=No, DOE=Yes}                             => {Regression=No} 0.12054795 0.9565217  0.12602740 1.208064 44
[4]  {Regression=No, DOE=Yes}                        => {Intro=No}      0.12054795 0.8461538  0.14246575 1.397494 44
[5]  {Cat.Data=No, DOE=Yes}                          => {Intro=No}      0.10136986 0.8043478  0.12602740 1.328448 37
[6]  {Survey=No, DOE=Yes}                            => {Intro=No}      0.10684932 0.7647059  0.13972603 1.262976 39
[7]  {DataMining=No, DOE=Yes}                        => {Intro=No}      0.11232877 0.7592593  0.14794521 1.253980 41
[8]  {Forecast=No, DOE=Yes}                          => {Intro=No}      0.10684932 0.7358491  0.14520548 1.215316 39
[9]  {DataMining=Yes, Regression=No}                 => {Intro=No}      0.10684932 0.7959184  0.13424658 1.314526 39
[10] {DataMining=Yes, Cat.Data=No}                   => {Intro=No}      0.09589041 0.7446809  0.12876712 1.229903 35
[11] {DataMining=Yes, Forecast=No}                   => {Intro=No}      0.10136986 0.7400000  0.13698630 1.222172 37
[12] {Survey=Yes, SW=No}                             => {Intro=No}      0.10958904 0.8000000  0.13698630 1.321267 40
[13] {Survey=Yes, Cat.Data=No}                       => {Intro=No}      0.09041096 0.7333333  0.12328767 1.211161 33
[14] {Survey=Yes, Forecast=No}                       => {Intro=No}      0.10958904 0.7407407  0.14794521 1.223395 40
[15] {Regression=Yes, SW=No}                         => {Intro=No}      0.12054795 0.7857143  0.15342466 1.297673 44
[16] {Cat.Data=No, Regression=Yes}                   => {Intro=No}      0.11506849 0.7500000  0.15342466 1.238688 42
[17] {DataMining=No, Regression=Yes}                 => {Intro=No}      0.12054795 0.7333333  0.16438356 1.211161 44
[18] {Regression=Yes, DOE=No}                        => {Intro=No}      0.13150685 0.7384615  0.17808219 1.219631 48
[19] {Cat.Data=Yes, Regression=No}                   => {Intro=No}      0.11506849 0.7500000  0.15342466 1.238688 42
[20] {Regression=No, Forecast=Yes, SW=No}            => {Intro=No}      0.05479452 0.7407407  0.07397260 1.223395 20
[21] {Survey=No, Regression=No, Forecast=Yes}        => {Intro=No}      0.05205479 0.7600000  0.06849315 1.255204 19
[22] {DataMining=No, Survey=No, Forecast=Yes}        => {Intro=No}      0.05205479 0.7307692  0.07123288 1.206927 19
[23] {Regression=No, DOE=Yes, SW=No}                 => {Intro=No}      0.09315068 0.8947368  0.10410959 1.477733 34
[24] {Cat.Data=No, DOE=Yes, SW=No}                   => {Intro=No}      0.08493151 0.9117647  0.09315068 1.505856 31
[25] {Survey=No, DOE=Yes, SW=No}                     => {Intro=No}      0.08767123 0.8888889  0.09863014 1.468074 32
[26] {DataMining=No, DOE=Yes, SW=No}                 => {Intro=No}      0.09041096 0.9166667  0.09863014 1.513952 33
[27] {Forecast=No, DOE=Yes, SW=No}                   => {Intro=No}      0.08767123 0.8888889  0.09863014 1.468074 32
[28] {Cat.Data=No, Regression=No, DOE=Yes}           => {Intro=No}      0.09589041 0.8974359  0.10684932 1.482191 35
[29] {Intro=No, Survey=No, DOE=Yes}                  => {Regression=No} 0.10410959 0.9743590  0.10684932 1.230592 38
[30] {Survey=No, Regression=No, DOE=Yes}             => {Intro=No}      0.10410959 0.8636364  0.12054795 1.426368 38
[31] {Intro=No, DataMining=No, DOE=Yes}              => {Regression=No} 0.10684932 0.9512195  0.11232877 1.201367 39
[32] {DataMining=No, Regression=No, DOE=Yes}         => {Intro=No}      0.10684932 0.8666667  0.12328767 1.431373 39
[33] {Intro=No, Forecast=No, DOE=Yes}                => {Regression=No} 0.10410959 0.9743590  0.10684932 1.230592 38
[34] {Regression=No, Forecast=No, DOE=Yes}           => {Intro=No}      0.10410959 0.8636364  0.12054795 1.426368 38
[35] {Survey=No, Cat.Data=No, DOE=Yes}               => {Intro=No}      0.09041096 0.8048780  0.11232877 1.329323 33
[36] {DataMining=No, Cat.Data=No, DOE=Yes}           => {Intro=No}      0.09589041 0.8139535  0.11780822 1.344312 35
[37] {Cat.Data=No, Forecast=No, DOE=Yes}             => {Intro=No}      0.09041096 0.8048780  0.11232877 1.329323 33
```

Sólo obtenemos un “predicciones” negativas en el rhs. Es decir, ninguna regla nos dice que si compramos *x* curso es probable que también compre el curso *y*. Obviamente esto no nos ayuda a saber qué agrupaciones de cursos sugerirle a la empresa para incrementar sus ingresos. Pero no es todo, en aquellas reglas de “predicción positiva” sólo tenemos valores “No” en lhs. Es como decir que si el cliente no compra los cursos *A complemento*, entonces es probable que compre *A*.

```
> inspect(sort(rules, by = "lift")[1:100])        #Las 10 reglas con mayor lift
     lhs                                                rhs                 support    confidence coverage   lift     count
[1]  {Intro=No,
      DataMining=No,
      Survey=No,
      Cat.Data=No,
      Regression=No,
      Forecast=No,
      SW=No}                                         => {DOE=Yes}         0.07123288 0.7878788  0.09041096 4.564695 26
[2]  {Intro=No,
      Survey=No,
      Cat.Data=No,
      Regression=No,
      Forecast=No,
      DOE=No,
      SW=No}                                         => {DataMining=Yes}  0.06575342 0.7741935  0.08493151 4.347395 24
[3]  {Intro=No,
      DataMining=No,
      Cat.Data=No,
      Regression=No,
      Forecast=No,
      DOE=No,
      SW=No}                                         => {Survey=Yes}      0.06301370 0.7666667  0.08219178 4.115196 23
[4]  {Intro=No,
      DataMining=No,
      Survey=No,
      Cat.Data=No,
      Forecast=No,
      DOE=No,
      SW=No}                                         => {Regression=Yes}  0.09041096 0.8250000  0.10958904 3.962171 33
[5]  {Intro=No,
      DataMining=No,
      Survey=No,
      Cat.Data=No,
      Regression=No,
      SW=No}                                         => {DOE=Yes}         0.07397260 0.6279070  0.11780822 3.637874 27
```

Si hacemos un filtrado para conservar sólo las reglas con un lhs de “Yes” pero en que todos los cursos de rhs no sean “No” obtenemos que no hay reglas así. Es decir, que las únicas reglas que nos dan rhs=”Yes” son aquellas en que lhs=”No” para todos sus cursos.

```
> rules_yes <- subset(rules, rhs %pin% "=Yes") #Mantener las reglas donde el RHS tenga al menos un curso con "Yes"
> rules_no_redundant <- subset(rules_yes, !all(lhs %pin% "=No")) #Eliminar reglas redundantes (donde todos los de lhs son "No")
> length(rules_no_redundant)
[1] 0
```

Veremos si generando las reglas de nuevo podemos mejorar los resultados. Para ello reduje el soporte a 0.006 y para agilizar las cosas sólo marengo aquellas reglas en que rhs=”Yes”.

```
> rules_v2 <- apriori(data_trans,
+                     parameter = list(support = 0.006, confidence = 0.6, minlen = 2),
+                     appearance = list(rhs = c("Intro=Yes", "DataMining=Yes","Survey=Yes","Cat.Data=Yes",'
E=Yes","SW=Yes")))
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.6    0.1    1 none FALSE            TRUE       5   0.006      2     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 2
```

Como vemos, sólo generamos 30 reglas:

```
> summary(rules_v2)
set of 30 rules
```

Si inspeccionamos dichas reglas nos daremos cuenta de que otra vez, no generamos reglas de calidad. Todas las reglas con rhs=”Yes” implican que todos los cursos de lhs tengan “No”.

```
> inspect(rules_v2)
     lhs                rhs                 support confidence   coverage     lift count
[1]  {DataMining=No,
      Survey=No,
      Cat.Data=No,
      Regression=No,
      SW=No}         => {Intro=Yes}      0.22191781  0.6532258 0.33972603 1.655746    81
[2]  {Survey=No,
      Cat.Data=No,
      Regression=No,
      DOE=No,
      SW=No}         => {Intro=Yes}      0.22739726  0.6640000 0.34246575 1.683056    83
```

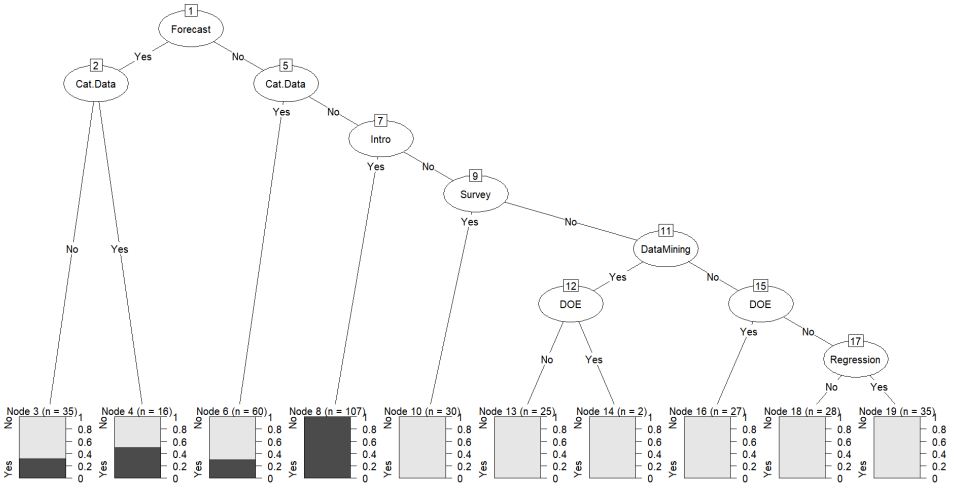
Dato curioso: Si aumento una décima el soporte deja de generar reglas.

En fin, ¿Qué podemos concluir de estos resultados?, básicamente que Apriori puede no ser un modelo bueno para nuestros datos, ya que si recordamos, tenemos pocas observaciones. Pero al menos sobre los resultados que obtuvimos podemos inferir que la mayoría de las transacciones donde no se compra nada (todos los cursos son "No")

terminan con la compra de un curso en particular (como "Intro=Yes" o "DataMining=Yes"), y el algoritmo podría generar reglas que reflejan este patrón, que si bien es redundante pareciera ser que es lo más interesante que captó.

Pero eso no es todo, veamos qué nos dice C.50 al respecto.

```
> evaluate_c50_model(models_c50$Intro, data, "Intro")
Evaluación del árbol para Intro :
        Real
Predicho  No Yes
     No  211  53
     Yes  10  91
Precisión: 82.74 %
-----------------------------------
```



C.50 tiene un rendimiento bueno, por lo que podemos considerarlo para hacer predicciones. Ahora la duda es, ¿Random Forest será mejor?, quien sabe, pero creo que la cantidad de observaciones jugó en mucha contra para las reglas a priori, incluso pese a que este problema es básicamente un problema de la canasta de compras.

En fin, lo bueno que dejaron las reglas a priori fue este gráfico.