Examen Predicción Final

Alvaro de Prada

2/2/2018

Introducción:

Para el desarrollo del presente trabajo se ha proporcionado un dataset con un total de 1239 observaciones, representando cada una de ellas a una empresa. Para cada observación se incluye el valor de diferentes ratios específicos calculados a partir de la contabilidad de una empresa. Estos ratios comparan años anteriores para obtener una cifra. Finalmente, vada observación consta de una columna final llamada 'Manipulater', la cual indica si El objetivo es desarrollar un modelo o modelos de predicción eficientes a la hora de predecir una variable clasificatoria. La variable clasificatioria a prediecir es la llamada "Manipulater" tomará el valor Yes o No en función de si se ha descubierto que la empresa manipulaba la contabilidad y por tanto cometía fraude. Por lo tanto, se tratará averiguar si los ratios aportan información suficiente para poder predecir si una empresa está manipulando o no su contabilidad.

Datos:

En primer lugar procedemos a cargar las librerías que emplearemos a lo largo del trabajo, así como los datos para llevar a cabo un análisis exploratorio:

```
library(readxl)
library(Hmisc)

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula

## Loading required package: ggplot2

##
## Attaching package: 'Hmisc'

## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units

library(caret) # Matriz de confusión

## Warning in as.POSIXlt.POSIXct(Sys.time()): unknown timezone 'default/
## Europe/Madrid'
```

```
##
## Attaching package: 'caret'

## The following object is masked from 'package:survival':
##
##     cluster

library(mlr) # Método Oversampling

## Loading required package: ParamHelpers

## Warning: replacing previous import 'BBmisc::isFALSE' by
## 'backports::isFALSE' when loading 'mlr'

##
## Attaching package: 'mlr'

## The following object is masked from 'package:caret':
##
##     train

## The following object is masked from 'package:Hmisc':
##
##     impute

dataset_orig <- read_excel("DATASET_RAQ_JMLZ(1).xlsx")
dataset_orig <-as.data.frame(dataset_orig[,2:10]) # Eliminamos Company ID
y añadimos una columna.
dataset_orig$Manipulater <- as.factor(ifelse(dataset_orig$Manipulater ==
'Yes', 1, 0)) # Transformamos la variable Manipulater
str(dataset_orig)

## 'data.frame':    1239 obs. of  9 variables:
##  $ DSRI       : num  1.62 1 1 1.49 1 ...
##  $ GMI        : num  1.13 1.61 1.02 1 1.37 ...
##  $ AQI        : num  7.185 1.005 1.241 0.466 0.637 ...
##  $ SGI        : num  0.366 13.081 1.475 0.673 0.861 ...
##  $ DEPI       : num  1.38 0.4 1.17 2 1.45 ...
##  $ SGAI       : num  1.6241 5.1982 0.6477 0.0929 1.7415 ...
##  $ ACCR       : num  -0.1668 0.0605 0.0367 0.2734 0.123 ...
##  $ LEVI       : num  1.161 0.987 1.264 0.681 0.939 ...
##  $ Manipulater: Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 ...
```

En estos primeros pasos del formateo de datos hemos: - Eliminado la primera columna ya que no aporta información adicional. - Transformado la variable Manipulater a factor para poder trabajar con ella.

A continuación vamos a hacer un summary para conocer los principales datos de las variables:

```
summary(dataset_orig)
```

```
##       DSRI              GMI                AQI
##  Min.   : 0.0000   Min.   :-20.8118   Min.   :-32.8856
##  1st Qu.: 0.8908   1st Qu.:  0.9271   1st Qu.:  0.7712
##  Median : 1.0227   Median :  1.0000   Median :  1.0040
##  Mean   : 1.1691   Mean   :  0.9879   Mean   :  0.9978
##  3rd Qu.: 1.1925   3rd Qu.:  1.0580   3rd Qu.:  1.2163
##  Max.   :36.2912   Max.   : 46.4667   Max.   : 52.8867
##       SGI                DEPI              SGAI              ACCR
##  Min.   : 0.02769   Min.   :0.06882   Min.   : 0.0000   Min.   :-
3.14350
##  1st Qu.: 0.97021   1st Qu.:0.93690   1st Qu.: 0.8988   1st Qu.:-
0.07633
##  Median : 1.08896   Median :1.00191   Median : 1.0000   Median :-
0.02924
##  Mean   : 1.12709   Mean   :1.04014   Mean   : 1.1072   Mean   :-
0.03242
##  3rd Qu.: 1.19998   3rd Qu.:1.08136   3rd Qu.: 1.1300   3rd Qu.:
0.02252
##  Max.   :13.08143   Max.   :5.39387   Max.   :49.3018   Max.   :
0.95989
##       LEVI         Manipulater
##  Min.   : 0.0000   0:1200
##  1st Qu.: 0.9232   1:  39
##  Median : 1.0131
##  Mean   : 1.0571
##  3rd Qu.: 1.1156
##  Max.   :13.0586
```

De esta tabla sacamos varias conclusiones: - El dataset no contiene NAs. - La mayoría de los indicadores tienen una media cercana al 1, pero muchos tienen mínimos o máximos muy alejados a la media. Esto será estudiado posteriormente para concluir si existen outliers y que acciones tomar.

- La variable manipulater está muy poco balanceada. Posteriormente se propondrán distintos métodos para balancearla.

Una vez conocemos las variables de las que se compone nuestro dataset, se ha llevado a cabo una breve investigación y en la que se ha descubierto que todas estas variables son empleadas a la hora de calcular el M-Score de Beneish.

Análisis de Outliers:

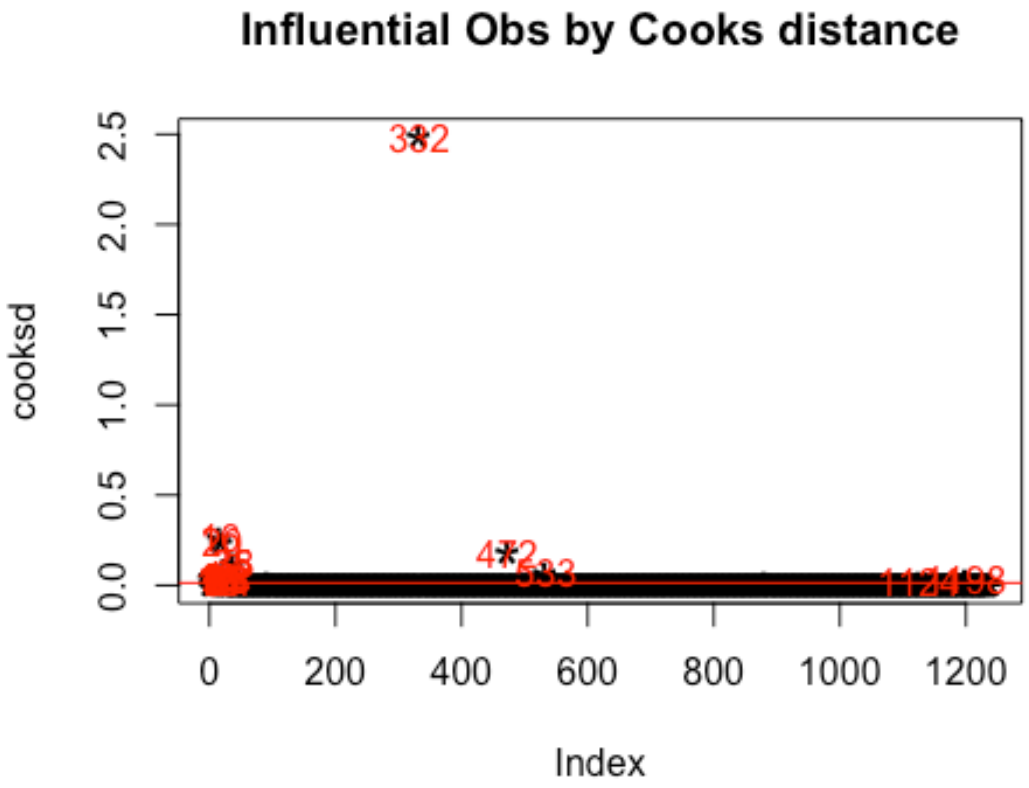
Al no haber encontrado NAs, vamos a comprobar si existen outliers en el dataset que puedan hacer que los resultados de nuestro trabajo posterior se vean perjudicados por datos erróneos. Para ello emplearemos la distancia de Cook:

```
mod <- glm(Manipulater ~ ., data=dataset_orig, family = "binomial")
cooksd <- cooks.distance(mod)
# Influence measures
```

```
# In general use, those observations that have a cook's distance greater
than 4 times the mean may be classified as influential. This is not a
hard boundary.
```

Y la representamos gráficamente. Todos los puntos por encima de la línea roja pueden ser considerados outliers, ya que difieren mucho del resto de observaciones en cuanto a la desviación típica media de las variables.

```
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")  #
plot cook's distance
abline(h = 4*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4*mean(cooksd,
na.rm=T),names(cooksd),""), col="red")  # add labels
```



Vemos que hay varias observaciones que pueden ser outliers, vamos a obtener la tabla con todas las observaciones que identifica como outlier para poder analizarlos:

```
influential <- cooksd>4*mean(cooksd, na.rm=T)  # influential row numbers
outliers1 <- (dataset_orig[influential, ])  # influential observations.
outliers1
```

```
##            DSRI       GMI       AQI       SGI       DEPI       SGAI
## 1    1.62474159 1.1289269 7.1850534 0.3662115 1.38151909 1.62414487
## 4    1.48623853 1.0000000 0.4655348 0.6728395 2.00000000 0.09288991
```

```
## 5    1.00000000  1.3690378  0.6371120  0.8613464 1.45467566 1.74145963
## 6    0.90553202  1.3609149  0.7839949  1.7933237 1.27824407 0.50526004
## 8    7.65997616  0.5801841  1.0357910  1.4854401 0.67883321 0.65365114
## 9    1.34627905  2.4735627  0.7025301  0.9656245 0.98527288 0.59451069
## 10   1.00000000  1.0000000  1.1666382  2.5925926 0.85034014 0.55714286
## 12   1.44838859  0.7956711  4.0016884  1.0673772 1.08554677 3.73449140
## 13   1.27404828  1.0000000  1.0771002  0.9078743 1.24043575 3.73637263
## 16   8.34761476  0.4952399  0.5730163  0.6721113 0.94186047 1.48784878
## 18   1.77824055  0.7422458 -2.1300996  1.9373082 1.00000000 1.80663046
## 20   1.00000000 -1.4166667 16.4266502  0.4166667 0.06882266 2.40000000
## 21   1.00000000  1.0000000 -1.5405442  2.6666667 0.16576087 1.00000000
## 25   2.04000000  1.0000000  0.2915308  0.2192982 1.00000000 1.00000000
## 27   2.23548606  1.0000000  0.4727173  1.1690208 1.03636364 0.90429775
## 33   1.00993383  1.0955838  5.7029044  0.9027431 1.23717437 0.62458785
## 34   1.00000000  1.0000000  0.4083859  1.0000000 1.00000000 1.00000000
## 35   1.00000000  1.0000000  1.0862810  4.5270204 0.53154108 2.24009794
## 38   3.75359653  3.1014341  1.1635660  0.8479576 1.00288512 0.27600736
## 332  0.05948088 -7.0366325  1.4208306 11.2586120 1.00000000 1.00000000
## 472  0.96275277  1.1342677  0.4010216  1.4978367 0.99440018 1.09215570
## 533  3.28201478  3.3516899  0.7063364  1.7416026 1.13377323 1.15527111
## 1124 7.11765758  1.0589964  0.6875815  0.3674066 1.02212809 1.71689310
## 1198 6.08840461  4.5941239  1.0423514  0.1540551 0.87850133 2.03816813
##              ACCR        LEVI Manipulater
## 1    -0.166808698  1.16108165           1
## 4     0.273434125  0.68097501           1
## 5     0.123047699  0.93904722           1
## 6     0.054642384  1.54313707           1
## 8     0.003651969  1.10159131           1
## 9    -0.013484949  1.02845877           1
## 10    0.001028454  0.16302565           1
## 12    0.090457908  0.22209013           1
## 13    0.358456918  0.37934790           1
## 16    0.060810811 13.05855856           1
## 18    0.146384480  1.07953961           1
## 20   -0.004070951  1.00000000           1
## 21    0.035244030  0.01058017           1
## 25    0.717842324  1.00000000           1
## 27    0.510336345  0.53606535           1
## 33    0.007919807  2.55623624           1
## 34    0.387133281  0.35826895           1
## 35    0.087841620  8.56386632           1
## 38   -0.592257932  1.07058503           1
## 332  -0.066855147  2.26830921           0
## 472   0.959887640  0.81297134           0
## 533   0.405920296  1.98453029           0
## 1124  0.043370276  1.50667245           0
## 1198 -0.170060630  1.11699875           0
```

Esta tabla arroja mucha información, la más relevante es la siguiente: - Podemos ver que clasifica a muchas de las observaciones como outliers ya que en alguna de sus

variables tienen valores que se alejan excesivamente de la media. Recordemos que la mayoría de las variables tenían una media cercana al 1. Sin embargo en estas observaciones vemos, por ejmplo en la primera observación, que el valor del ratio AQI es de 7.18, muy alejado de la media. Si vamos observación a observación podremos ver que situaciones similares se repiten en todasa. En total, 24 observaciones. - ¿Debemos considerarla outliers y eliminarlas del dataset? Por un lado si podríamos decir que se trata de outliers. Sin embargo, de las 24 observaciones obtenidas, 19 son de Manipulater=1. Recordemos que en el dataset completo existían un total de 39 Manipulater=1/Yes. Es decir, más de la mitad de las observaciones de tipo Manipulater=1/Yes están compuestas de algún tipo de outlier.

Por lo tanto, la conlcusión que obtenemos es que si bien pueden tratarse de outliers, lo más probable es que el propio hecho de que alguno de los ratios contena un outlier eleva la probabilidad de que se esté manipulando la contabilidad. Por lo tanto, no podemos quitar los outliers del dataset ya que nos están aportando una información muy valiosa.

Beneish M Score:

El M-Score de Beneish fue creado por el profesor Messod Beneish. Se trata de un indicador para detectar la manipulación de ingresos en las empresas. (Messod D. Beneish, 1999)

Su funcionamiento es simple. Le da un peso concreto a cada uno de los ocho indicadores incluidos en el dataset del presente trabajo para crear un nuevo indicador.

M-Score = -4.84 + 0.92*DSRI + 0.528*GMI + 0.404*AQI + 0.892*SGI + 0.115*DEPI – 0.172*SGAI + 4.679*ACCR – 0.327*LEVI

Un M-Score mayor a -1.78 indicaría una elevada probabilidad de manipulación.

Por lo tanto, lo que haremos será crear una nueva columna en nuestro dataset original que incluya el resultado del M-Score para cada observación, para posteriormente poder trabajar con esta columna, a sabiendas de que está correlacionada con el resto.

```
mscore <- -4.84 + 0.92*dataset_orig$DSRI + 0.528*dataset_orig$GMI +
0.404*dataset_orig$AQI + 0.892*dataset_orig$SGI + 0.115*dataset_orig$DEPI
- 0.172*dataset_orig$SGAI + 4.679*dataset_orig$ACCR -
0.327*dataset_orig$LEVI

dataset_orig <- cbind(dataset_orig, mscore)
summary(dataset_orig)
```

```
##       DSRI              GMI                AQI
##  Min.   : 0.0000   Min.   :-20.8118   Min.   :-32.8856
##  1st Qu.: 0.8908   1st Qu.:  0.9271   1st Qu.:  0.7712
##  Median : 1.0227   Median :  1.0000   Median :  1.0040
##  Mean   : 1.1691   Mean   :  0.9879   Mean   :  0.9978
```

```
##  3rd Qu.: 1.1925   3rd Qu.:  1.0580   3rd Qu.:  1.2163
##  Max.   :36.2912   Max.   : 46.4667   Max.   : 52.8867
##       SGI                DEPI              SGAI              ACCR
##  Min.   : 0.02769   Min.   :0.06882   Min.   : 0.0000   Min.   :-
3.14350
##  1st Qu.: 0.97021   1st Qu.:0.93690   1st Qu.: 0.8988   1st Qu.:-
0.07633
##  Median : 1.08896   Median :1.00191   Median : 1.0000   Median :-
0.02924
##  Mean   : 1.12709   Mean   :1.04014   Mean   : 1.1072   Mean   :-
0.03242
##  3rd Qu.: 1.19998   3rd Qu.:1.08136   3rd Qu.: 1.1300   3rd Qu.:
0.02252
##  Max.   :13.08143   Max.   :5.39387   Max.   :49.3018   Max.   :
0.95989
##       LEVI         Manipulater     mscore
##  Min.   : 0.0000   0:1200      Min.   :-17.570
##  1st Qu.: 0.9232   1:  39      1st Qu.: -2.928
##  Median : 1.0131               Median : -2.491
##  Mean   : 1.0571               Mean   : -2.403
##  3rd Qu.: 1.1156               3rd Qu.: -2.044
##  Max.   :13.0586               Max.   : 26.383
```

Con esto damos por concluido el tratamiento previo de los datos.

Predicción: glm

A continuación vamos a llevar a cabo un modelo predictivo sobre el dataset tratado. Previamente, dividiremos el dataset tratado en muestra de training y test, al 70-30.

```
set.seed(200)
sep1 <- sample(nrow(dataset_orig), 0.7*nrow(dataset_orig))
train1 <- dataset_orig[sep1,]
test1 <- dataset_orig[-sep1,]
```

Comprobamos la distribución de la variable a predecir para asegurarnos de que el peso está equilibrado.

```
describe(train1$Manipulater)

## train1$Manipulater
##        n  missing distinct
##      867        0        2
##
## Value          0     1
## Frequency    839    28
## Proportion 0.968 0.032

describe(test1$Manipulater)
```

```
## test1$Manipulater
##        n  missing distinct
##      372        0        2
##
## Value         0    1
## Frequency   361   11
## Proportion 0.97 0.03
```

Vemos que en ambas muestras hay aproximadamente un 3% de observaciones con manipulaciones, por lo que está balanceado.

Llevamos a cabo un primer modelo con todas las variables menos con el m-score, ya que no tendría sentido pues está correlacionada con todas las demás.

```
modelo1 <- glm(Manipulater ~.-mscore  , data=train1, family = "binomial")
summary(modelo1)
```

```
##
## Call:
## glm(formula = Manipulater ~ . - mscore, family = "binomial",
##     data = train1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5329  -0.1993  -0.1627  -0.1318   3.0111
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.68220    1.03521  -6.455 1.08e-10 ***
## DSRI         0.73267    0.19507   3.756 0.000173 ***
## GMI          0.37395    0.28021   1.335 0.182024
## AQI          0.33218    0.07806   4.255 2.09e-05 ***
## SGI          0.57999    0.28311   2.049 0.040497 *
## DEPI         0.32477    0.54205   0.599 0.549075
## SGAI         0.08747    0.34856   0.251 0.801850
## ACCR         5.07847    1.41385   3.592 0.000328 ***
## LEVI         0.02728    0.22309   0.122 0.902673
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 247.32  on 866  degrees of freedom
## Residual deviance: 164.20  on 858  degrees of freedom
## AIC: 182.2
##
## Number of Fisher Scoring iterations: 8
```

Vemos que hay algunas variables más significativas que otras. Ejecutamos la función step sobre el modelo para tratar de averiguar que combinación de variables es la mejor.

```r
step(modelo1)
```

```
## Start:  AIC=182.2
## Manipulater ~ (DSRI + GMI + AQI + SGI + DEPI + SGAI + ACCR +
##     LEVI + mscore) - mscore
##
##        Df Deviance    AIC
## - LEVI  1   164.22 180.22
## - SGAI  1   164.36 180.36
## - DEPI  1   164.50 180.50
## <none>      164.20 182.20
## - SGI   1   170.75 186.75
## - GMI   1   174.39 190.39
## - ACCR  1   176.03 192.03
## - DSRI  1   184.62 200.62
## - AQI   1   187.20 203.20
##
## Step:  AIC=180.21
## Manipulater ~ DSRI + GMI + AQI + SGI + DEPI + SGAI + ACCR
##
##        Df Deviance    AIC
## - SGAI  1   164.36 178.36
## - DEPI  1   164.52 178.52
## <none>      164.22 180.22
## - SGI   1   170.81 184.81
## - GMI   1   174.42 188.42
## - ACCR  1   176.05 190.05
## - AQI   1   187.77 201.77
## - DSRI  1   190.04 204.04
##
## Step:  AIC=178.36
## Manipulater ~ DSRI + GMI + AQI + SGI + DEPI + ACCR
##
##        Df Deviance    AIC
## - DEPI  1   164.63 176.63
## <none>      164.36 178.36
## - SGI   1   172.94 184.94
## - GMI   1   174.52 186.52
## - ACCR  1   176.07 188.07
## - AQI   1   188.27 200.27
## - DSRI  1   212.84 224.84
##
## Step:  AIC=176.63
## Manipulater ~ DSRI + GMI + AQI + SGI + ACCR
##
##        Df Deviance    AIC
## <none>      164.63 176.63
## - SGI   1   172.94 182.94
## - GMI   1   174.77 184.77
## - ACCR  1   176.24 186.24
```

```
## - AQI   1   188.27 198.27
## - DSRI  1   212.88 222.88

##
## Call:  glm(formula = Manipulater ~ DSRI + GMI + AQI + SGI + ACCR,
family = "binomial",
##     data = train1)
##
## Coefficients:
## (Intercept)         DSRI          GMI          AQI          SGI
##     -6.2459       0.7619       0.3726       0.3296       0.5739
##        ACCR
##      4.9898
##
## Degrees of Freedom: 866 Total (i.e. Null);  861 Residual
## Null Deviance:       247.3
## Residual Deviance: 164.6     AIC: 176.6
```

Repetimos el modelo utilizando las variables SGI,GMI, ACCR, AQI y DSRI, pues son las que nos han dado un AIC menor.

```
modelo1 <- glm(Manipulater ~.-mscore -DEPI -SGAI -LEVI  , data=train1,
family = "binomial")
summary(modelo1)

##
## Call:
## glm(formula = Manipulater ~ . - mscore - DEPI - SGAI - LEVI,
##     family = "binomial", data = train1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.5130  -0.1998  -0.1626  -0.1331   2.9424
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.24593    0.71833  -8.695  < 2e-16 ***
## DSRI         0.76188    0.15315   4.975 6.53e-07 ***
## GMI          0.37260    0.27603   1.350 0.177068
## AQI          0.32964    0.07689   4.287 1.81e-05 ***
## SGI          0.57394    0.27306   2.102 0.035564 *
## ACCR         4.98983    1.38579   3.601 0.000317 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 247.32  on 866  degrees of freedom
## Residual deviance: 164.63  on 861  degrees of freedom
## AIC: 176.63
```

```
##
## Number of Fisher Scoring iterations: 7
```

A pesar de que algunas variables parecen no ser significativas, hemos llevado a cabo varias pruebas y hemos comprobado que el AIC es menor con dichas variables icluidas. Llevamos a cabo la predicción sobre la muestra de test con nuestro modelo:

```
prediction1 <- predict(modelo1, test1, type="response")
```

Finalmente comprobamos el accuracy de nuestro modelo con la matriz de confusión:

```
confusionMatrix(test1$Manipulater, round(prediction1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 361   0
##          1  10   1
##
##                Accuracy : 0.9731
##                  95% CI : (0.9511, 0.987)
##     No Information Rate : 0.9973
##     P-Value [Acc > NIR] : 1.000000
##
##                   Kappa : 0.1625
##  Mcnemar's Test P-Value : 0.004427
##
##             Sensitivity : 0.97305
##             Specificity : 1.00000
##          Pos Pred Value : 1.00000
##          Neg Pred Value : 0.09091
##              Prevalence : 0.99731
##          Detection Rate : 0.97043
##    Detection Prevalence : 0.97043
##       Balanced Accuracy : 0.98652
##
##        'Positive' Class : 0
##
```

Obtenemos un 97.31% de precisión.

A continuación vamos a hacer un modelo con la variable m-score creada:

```
modelo2 <- glm(Manipulater ~mscore  + DSRI, data=train1, family =
"binomial")
summary(modelo2)
```

```
##
## Call:
## glm(formula = Manipulater ~ mscore + DSRI, family = "binomial",
##     data = train1)
```

```
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.7926  -0.2012  -0.1668  -0.1386   2.8427
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -2.6528     0.3947  -6.721 1.81e-11 ***
## mscore        0.7580     0.1497   5.062 4.14e-07 ***
## DSRI          0.3108     0.1193   2.604  0.00921 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 247.32  on 866  degrees of freedom
## Residual deviance: 168.82  on 864  degrees of freedom
## AIC: 174.82
##
## Number of Fisher Scoring iterations: 7
```

Tras realizar varias pruebas hemos concluido que la variable m-score junto con DSRI obtiene un AIC menor.

A continuación predecimos con este nuevo modelo que emplea el m-score:

```
prediction2 <- predict(modelo2, test1, type="response")
```

Finalmente comprobamos el accuracy de nuestro modelo con la matriz de confusión:

```
confusionMatrix(test1$Manipulater, round(prediction2))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 360   1
##          1  10   1
##
##                Accuracy : 0.9704
##                  95% CI : (0.9477, 0.9851)
##     No Information Rate : 0.9946
##     P-Value [Acc > NIR] : 1.00000
##
##                   Kappa : 0.1461
##  Mcnemar's Test P-Value : 0.01586
##
##             Sensitivity : 0.97297
##             Specificity : 0.50000
##          Pos Pred Value : 0.99723
##          Neg Pred Value : 0.09091
```

```
##              Prevalence : 0.99462
##          Detection Rate : 0.96774
##    Detection Prevalence : 0.97043
##       Balanced Accuracy : 0.73649
##
##        'Positive' Class : 0
##
```

Obtenemos un 97.04% menor que en el anterior modelo.

Finalmente, aunque no sea una predicción como tal, vamos a poner en práctica la teoría del M-Score que dice que un m-score mayor de -1.78 tiene probabilidad elevada de manipular. Para ello, llevaremos a cabo la prueba sobre el total de datos, ya que no se trata de un modelo que haya sido entrenado y por lo tanto no es necesario utilizar la muestra de test.

```
manip_mscore <- ifelse(dataset_orig$mscore>-1.78,1,0)

confusionMatrix(dataset_orig$Manipulater, manip_mscore)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1029  171
##          1    8   31
##
##                Accuracy : 0.8555
##                  95% CI : (0.8347, 0.8746)
##     No Information Rate : 0.837
##     P-Value [Acc > NIR] : 0.04016
##
##                   Kappa : 0.2159
##  Mcnemar's Test P-Value : < 2e-16
##
##             Sensitivity : 0.9923
##             Specificity : 0.1535
##          Pos Pred Value : 0.8575
##          Neg Pred Value : 0.7949
##              Prevalence : 0.8370
##          Detection Rate : 0.8305
##    Detection Prevalence : 0.9685
##       Balanced Accuracy : 0.5729
##
##        'Positive' Class : 0
##
```

Vemos que el m-score predice peor que los modelos desarrollados.

Sin embargo, lo que observamos tras llevar a cabo los tres modelos anteriores, es que al haber muy pocos datos de empresas que manipuilan, el hecho de concluir que

ninguna empresa manipula obtendría un porcentaje de acierto superior al resto de modelos. Comprobémoslo:

```
nulo <- 0
dataset_orig <- cbind(dataset_orig, nulo)

confusionMatrix(reference= dataset_orig$Manipulater, dataset_orig$nulo)

## Warning in confusionMatrix.default(reference =
dataset_orig$Manipulater, :
## Levels are not in the same order for reference and data. Refactoring
data
## to match.

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1200   39
##          1    0    0
##
##                Accuracy : 0.9685
##                  95% CI : (0.9572, 0.9775)
##     No Information Rate : 0.9685
##     P-Value [Acc > NIR] : 0.5425
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 1.166e-09
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.9685
##          Neg Pred Value :    NaN
##              Prevalence : 0.9685
##          Detection Rate : 0.9685
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

Comprobamos que si dijeramos que ninguna empresa manipula, acertaríamos un 96.85% de las veces. Este porcentaje es ligeramente inferior al de las predicciones, lo cual nos plantea ciertas cuestiones:

¿Están los modelos sesgados por el propio peso de la variable a predecir? ¿Podemos hacer algo para balancear los dataset y que así los modelos tengan una mayor fiabilidad?

Oversampling y Undersamplning:

(source: https://mlr-org.github.io/mlr-tutorial/release/html/over_and_undersampling/index.html)

Las anteriores preguntas nos llevan a encontrar posibles soluciones a un dataset desequilibrado en un problema de clasificación. Estas soluciones son los métododos de basados en muestreo:

- Oversampling: Consiste en generar observaciones adicionales de la clase minoritaria (en nuestro caso las observaciones con Manipulater=Yes), mediante la creación de copias exactas de observaciones minoritarias existentes, mientras que la clase mayoritaria no se manipula, sigue siendo constante.
- Oversampling-Smote: Se trata de una variación del oversampling que sigue la misma metodología, salvo que en lugar de duplicar observaciones minoritarias (que podría ocasionar overfitting), con este método las nuevas observaciones de la clase minoritaria son creadas siguiendo el método de los vecinos más cercanos.
- Undersampling: Esta técnica seria la contraria a las anteriores, ya que consiste en la eliminación de variables mayoritarias para conseguir de esta manera equilibrar las proporciones.

Por lo tanto, el oversampling estandar no lo vamos a emplear ya que el smote nos parece más correcto por su forma de proceder. Con lo cual, a continuación vamos a crear 2 nuevos datasets, uno con cada técnica, para repetir los modelos de predicción anteriores sobre estos nuevos datasets más equilibrados. (Librería mlr)

Utilizando la función makeClassifTask indicamos cual es la variable que está desequilibrada, la variable que queremos predecir:

```
task = makeClassifTask(data = dataset_orig[,-(10:11)], target =
"Manipulater")
```

Implementamos la función de Smoting:

```
task.smote = smote(task, rate = 8, standardize = T) # fijamos standarize
a True para un mejor calculo de la distancia de gower.
```

Implementamos la función de Undersampling:

```
task.under = undersample(task, rate = 1/8)
```

Comprobamos como la variable manipulater se encuentra más equilibrada con los tres métodos:

```
table(getTaskTargets(task.smote))
```

```
##
##    0    1
## 1200  312
```

```
table(getTaskTargets(task.under))
```

```
##
##   0   1
## 150  39
```

Finalmente creamos los tres nuevos datasets:

```
smote_dataset <- task.smote$env$data
under_dataset <- task.under$env$data
```

A continuación repetiremos los pasos realizados en las predicciones del inicio del trabajo para los nuevos datasets. Como los pasos son los mismos, no requieren expicación esta vez.

Modelo con oversampling - Smote:

```
set.seed(200)
sep2 <- sample(nrow(smote_dataset), 0.7*nrow(smote_dataset))
smotetrain <- smote_dataset[sep2,]
smotetest <- smote_dataset[-sep2,]
```

Comprobamos la distribución de la variable a predecir para asegurarnos de que el peso está equilibrado.

```
describe(smotetrain$Manipulater)
```

```
## smotetrain$Manipulater
##        n  missing distinct
##     1058        0        2
##
## Value          0     1
## Frequency    835   223
## Proportion 0.789 0.211
```

```
describe(smotetest$Manipulater)
```

```
## smotetest$Manipulater
##        n  missing distinct
##      454        0        2
##
## Value          0     1
## Frequency    365    89
## Proportion 0.804 0.196
```

Vemos que las observaciones con manipulación han pasado del 3% al 20%.

Tras varias pruebas obtenemos que este modelo resulta eficiente:

```
modelosmote1 <- glm(Manipulater ~ LEVI + GMI + SGI + AQI + ACCR + DSRI  ,
data=smotetrain, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(modelosmote1)
```

```
##
## Call:
## glm(formula = Manipulater ~ LEVI + GMI + SGI + AQI + ACCR + DSRI,
##     family = "binomial", data = smotetrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.7895  -0.4874  -0.3337  -0.1293   2.3197
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -6.04799    0.50931 -11.875  < 2e-16 ***
## LEVI        -0.34434    0.13748  -2.505   0.0123 *
## GMI          0.90508    0.18282   4.951 7.40e-07 ***
## SGI          1.43673    0.23313   6.163 7.14e-10 ***
## AQI          0.38910    0.05014   7.760 8.51e-15 ***
## ACCR         9.30690    1.00776   9.235  < 2e-16 ***
## DSRI         1.28689    0.18909   6.806 1.01e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1089.70  on 1057  degrees of freedom
## Residual deviance:  691.58  on 1051  degrees of freedom
## AIC: 705.58
##
## Number of Fisher Scoring iterations: 7
```

Llevamos a cabo la predicción sobre la muestra de test con nuestro modelo:

```
predictionsmote1 <- predict(modelosmote1, smotetest, type="response")
```

Finalmente comprobamos el accuracy de nuestro modelo con la matriz de confusión:

```
confusionMatrix(smotetest$Manipulater, round(predictionsmote1))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 347  18
##          1  48  41
##
##                Accuracy : 0.8546
##                  95% CI : (0.8188, 0.8858)
##     No Information Rate : 0.87
##     P-Value [Acc > NIR] : 0.8521500
```

```
##
##                  Kappa : 0.4714
##  Mcnemar's Test P-Value : 0.0003575
##
##             Sensitivity : 0.8785
##             Specificity : 0.6949
##          Pos Pred Value : 0.9507
##          Neg Pred Value : 0.4607
##              Prevalence : 0.8700
##          Detection Rate : 0.7643
##    Detection Prevalence : 0.8040
##       Balanced Accuracy : 0.7867
##
##        'Positive' Class : 0
##
```

Obtenemos un 85.46% de precisión.

¿Qué precisión obtendríamos si dijéramos que todos las observaciones son 'Manipulater = No'?

```
nulo <- 0
smote_dataset1 <- cbind(smote_dataset, nulo)

confusionMatrix(reference = smote_dataset1$Manipulater,
smote_dataset1$nulo)
```

```
## Warning in confusionMatrix.default(reference =
## smote_dataset1$Manipulater, : Levels are not in the same order for
## reference and data. Refactoring data to match.

## Confusion Matrix and Statistics
##
##           Reference
## Prediction    0    1
##          0 1200  312
##          1    0    0
##
##                Accuracy : 0.7937
##                  95% CI : (0.7724, 0.8138)
##     No Information Rate : 0.7937
##     P-Value [Acc > NIR] : 0.5152
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7937
##          Neg Pred Value :    NaN
##              Prevalence : 0.7937
```

```
##          Detection Rate : 0.7937
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
##        'Positive' Class : 0
##
```

Obtendríamos sólo un 79.37% de precisión, mucho menor que con el dataset original.

Modelo con undersampling:

```
set.seed(200)
sep3 <- sample(nrow(under_dataset), 0.7*nrow(under_dataset))
undertrain <- under_dataset[sep3,]
undertest <- under_dataset[-sep3,]
```

Comprobamos la distribución de la variable a predecir para asegurarnos de que el peso está equilibrado.

```
describe(undertrain$Manipulater)
```

```
## undertrain$Manipulater
##        n  missing distinct
##      132        0        2
##
## Value          0     1
## Frequency    105    27
## Proportion 0.795 0.205
```

```
describe(undertest$Manipulater)
```

```
## undertest$Manipulater
##        n  missing distinct
##       57        0        2
##
## Value          0     1
## Frequency     45    12
## Proportion 0.789 0.211
```

Vemos que las observaciones con manipulación han pasado del 3% al 20%.

Tras varias pruebas obtenemos que este modelo resulta eficiente:

```
modelounder1 <- glm(Manipulater ~ GMI + AQI + SGI + ACCR + DSRI,
data=undertrain, family = "binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
summary(modelounder1)
```

```
##
## Call:
```

```
## glm(formula = Manipulater ~ GMI + AQI + SGI + ACCR + DSRI, family =
"binomial",
##     data = undertrain)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.6202  -0.3861  -0.2019  -0.0621   2.4817
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -11.9969     2.4890  -4.820 1.44e-06 ***
## GMI           2.3240     0.9118   2.549 0.010808 *
## AQI           0.7480     0.1877   3.984 6.76e-05 ***
## SGI           3.5403     1.0851   3.263 0.001103 **
## ACCR         12.0178     2.9829   4.029 5.60e-05 ***
## DSRI          2.1640     0.6500   3.329 0.000871 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 133.753  on 131  degrees of freedom
## Residual deviance:  68.212  on 126  degrees of freedom
## AIC: 80.212
##
## Number of Fisher Scoring iterations: 8
```

Llevamos a cabo la predicción sobre la muestra de test con nuestro modelo:

```
predictionunder1 <- predict(modelounder1, undertest, type="response")
```

Finalmente comprobamos el accuracy de nuestro modelo con la matriz de confusión:

```
confusionMatrix(undertest$Manipulater, round(predictionunder1))

## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 41  4
##          1  3  9
##
##                Accuracy : 0.8772
##                  95% CI : (0.7632, 0.9492)
##     No Information Rate : 0.7719
##     P-Value [Acc > NIR] : 0.03495
##
##                   Kappa : 0.6415
##  Mcnemar's Test P-Value : 1.00000
##
##             Sensitivity : 0.9318
```

```
##             Specificity : 0.6923
##          Pos Pred Value : 0.9111
##          Neg Pred Value : 0.7500
##              Prevalence : 0.7719
##          Detection Rate : 0.7193
##    Detection Prevalence : 0.7895
##       Balanced Accuracy : 0.8121
##
##        'Positive' Class : 0
##
```

Obtenemos un 87.72% de precisión.

¿Qué precisión obtendríamos si dijéramos que todos las observaciones son 'Manipulater = No'?

```
nulo <- 0
under_dataset1 <- cbind(under_dataset, nulo)

confusionMatrix(reference = under_dataset1$Manipulater,
under_dataset1$nulo)
```

```
## Warning in confusionMatrix.default(reference =
## under_dataset1$Manipulater, : Levels are not in the same order for
## reference and data. Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 150  39
##          1   0   0
##
##                Accuracy : 0.7937
##                  95% CI : (0.7289, 0.849)
##     No Information Rate : 0.7937
##     P-Value [Acc > NIR] : 0.5427
##
##                   Kappa : 0
##  Mcnemar's Test P-Value : 1.166e-09
##
##             Sensitivity : 1.0000
##             Specificity : 0.0000
##          Pos Pred Value : 0.7937
##          Neg Pred Value :    NaN
##              Prevalence : 0.7937
##          Detection Rate : 0.7937
##    Detection Prevalence : 1.0000
##       Balanced Accuracy : 0.5000
##
```

```
##        'Positive' Class : 0
##
```

Obtendríamos sólo un 79.37% de precisión, mucho menor que con el dataset original.

Conclusiones:

En primer lugar, hagamos una recapitulación de las partes elaboradas en este traajo: - Carga de datos y limpieza (si procediera). Análisis exploratorio. - Análisis de outliers. - Explicación M-Score. - Modelos glm con dataset original. Prueba del M-Score. - Técnicas de oversampling y undersampling. - Modelos glm con datasets de oversampling y undersampling.

En primer lugar, hemos comprobado que existen outliers en el dataset, pero tras analizarlos hemos optado por no eliminarlos ya que estos outliers coinciden con la clase minoritaria y por lo tanto puede ser un factor fundamental a la hora de predecir si manipula o no. Es decir, ¿es el hecho de que sea outlier un factor de peso a la hora de que manipule?

En segundo lugar, hemos explorado la teoría del M-Score, comprobándo que aunque es eficiente, obtiene una precisión menor que los modelos que hemos desarrollado en este trabajo.

En tercer lugar hemos desarrollado varios modelos de predicción con el dataset original que obtenían precisiones de más del 95%. Si bien ésta es una cifra admirable, no nos hemos conformado puesto que hemos comprobado que una predicción en la que todos los valores a predecir sean definidos a 0 obtendría la misma precisión. Por este motivo, hemos llegado a la conlusión de que un dataset desequilibrado estaba produciendo overfiting, es decir, el modelo estaba viendo que en el más del 90% de los casos la predicción era 0, y estaba teniendo esto en cuenta a la hora de estimar el modelo, quedando por lo tanto sesgado. Esto se vería más claramente al tratar de predecir un nuevo conjunto, en el que todas las observaciones fueran Manipulater = 1. En este caso lo más probable es que el modelo predijera mucho peor y su precisión se redujera sustancialmente. Por las limitaciones de tiempo, este caso no ha podido ser demostrado.

Por este motivo, hemos llevado a cabo técnicas de muestreo de oversampling en su variante smote, la cual crea nuevas observaciones de la clase minoritaria ('Manipulater = Yes') basándose en técnicas de vecinos más cercanos. También hemos empleado la técnica de undersampling que elimina observaciones mayoritarias. El objetivo de ambas técnicas es el mismo; equilibrar en la medida de lo posible el peso de la variable a predecir en el dataset.

Hemos repetido los modelos de predicción con estos dos nuevos dataset y hemos obtenido, Oversampling-smote: 85.46% Undersampling: 87.72% A primera vista estas precisiones son menores que las de los primeros modelos con el dataset original, pero, ¿cual es la precisión en estos modelos de una predicción que diga que todas las

observaciones son 'Manipulater = 0'? nulo-Oversampling-smote: 79.37% nulo-Undersampling: 79.37%

Por lo tanto podemos comprobar que si bien la precisión baja, los resultados son más realistas ya que están considerando una población más equilibrada. La precisión de una predicción 'nula' es mucho menor que la predicción del modelo.

.