Análisis discriminante

Eduardo Galvan

2022-11-30

Análisis Discriminante sobre la base de datos de quiebras bancarias

Análisis discriminante en R

```
## Warning: package 'haven' was built under R version 4.2.2
```

```
## Warning: package 'MultBiplotR' was built under R version 4.2.2
```

Es necesario pasar la columna de Quiebra a factor para poder realizar los siguientes pasos

```
bancos$Quiebra = factor(bancos$Quiebra)
levels(bancos$Quiebra) = c('No', 'Si')
X = bancos[,2:10]
rownames(X) = bancos$BANCO
```

Hay 5 bancos en los que el valor de quiebra está ausente, asi que los tenemos que eliminar para realizar el análisis

```
X=X[1:60,]
quiebra = bancos[1:60, 11]
```

El primer paso es realizar un MANOVA para observar si existen diferencias significativas entre los 2 grupos:

```
analisis_manova = manova(as.matrix(X)~ quiebra)
summary(analisis_manova)
```

```
##           Df  Pillai approx F num Df den Df   Pr(>F)
## quiebra    1 0.57964   7.6606      9     50 5.78e-07 ***
## Residuals 58
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Observamos que $p < 0.05 $ por lo que rechazamos la H_0 y concluimos que existen diferencias significativas entre ambos grupos, mas allá de eso podemos comprobar que variables son las que presentan diferencias significativas entre ambos grupos:

```
summary.aov(analisis_manova)
```

```
##  Response R1Liqui1 :
##             Df  Sum Sq  Mean Sq F value Pr(>F)
## quiebra      1 0.01880 0.018797  1.6068   0.21
## Residuals   58 0.67851 0.011699
##
##  Response R2Liqui2 :
##             Df  Sum Sq   Mean Sq F value Pr(>F)
## quiebra      1 0.00058 0.0005824   0.072 0.7894
## Residuals   58 0.46936 0.0080924
##
##  Response R3Liqui3 :
##             Df  Sum Sq  Mean Sq F value Pr(>F)
## quiebra      1 0.01834 0.018341  1.3543 0.2493
## Residuals   58 0.78548 0.013543
##
##  Response R4Autof :
##             Df    Sum Sq    Mean Sq F value  Pr(>F)
## quiebra      1 0.0010948 0.00109481  5.9683 0.01763 *
## Residuals   58 0.0106394 0.00018344
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response R5RenEc :
##             Df     Sum Sq    Mean Sq F value    Pr(>F)
## quiebra      1 0.00035007 0.00035007  35.994 1.366e-07 ***
## Residuals   58 0.00056410 0.00000973
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response R6RenFin :
##             Df  Sum Sq  Mean Sq F value    Pr(>F)
## quiebra      1 0.21799 0.217990   46.03 6.738e-09 ***
## Residuals   58 0.27468 0.004736
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response R7Apal :
##             Df     Sum Sq    Mean Sq F value    Pr(>F)
## quiebra      1 0.00038709 0.00038709  34.971 1.891e-07 ***
## Residuals   58 0.00064200 0.00001107
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response R8CosVen :
##             Df  Sum Sq  Mean Sq F value    Pr(>F)
## quiebra      1 0.10361 0.103605  29.493 1.157e-06 ***
## Residuals   58 0.20375 0.003513
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##  Response R9Cash :
```

```
##             Df     Sum Sq    Mean Sq F value    Pr(>F)
## quiebra      1 0.00086769 0.00086769  52.694 1.076e-09 ***
## Residuals   58 0.00095507 0.00001647
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

ahora podemos podemos proceder a realizar el análisis discriminante:

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:MultBiplotR':
##
##     ginv
```

```
LDA=lda(X, quiebra)
LDA
```

```
## Call:
## lda(X, quiebra)
##
## Prior probabilities of groups:
##  No  Si
## 0.6 0.4
##
## Group means:
##     R1Liqui1  R2Liqui2  R3Liqui3    R4Autof     R5RenEc  R6RenFin      R7Apal
## No 0.4004583 0.2715722 0.4199139 0.02520278 0.006863889 0.1633500 0.007230556
## Si 0.3643292 0.2652125 0.3842250 0.01648333 0.001933333 0.0403125 0.002045833
##     R8CosVen      R9Cash
## No 0.8363278 0.012633333
## Si 0.9211500 0.004870833
##
## Coefficients of linear discriminants:
##                   LD1
## R1Liqui1   -64.077615
## R2Liqui2     4.863948
## R3Liqui3    55.935045
## R4Autof     -8.872928
## R5RenEc   3952.771619
## R6RenFin   -12.041570
## R7Apal   -3562.632076
## R8CosVen     1.568016
## R9Cash    -179.706699
```

```
Prediccion=predict(LDA, X)$class
ct2 <- table(quiebra, Prediccion)
ct2
```

```
##        Prediccion
## quiebra No Si
##      No 32  4
##      Si  1 23
```

```
ngrupo=table(quiebra)
ngrupo
```

```
## quiebra
## No Si
## 36 24
```

```
propgrupo=100*diag(ct2)/ngrupo
propgrupo
```

```
## quiebra
##       No       Si
## 88.88889 95.83333
```

```
ptoptotal=100*sum(diag(ct2))/sum(ngrupo)
ptoptotal
```

```
## [1] 91.66667
```

Observamos que el modelo es capaz de predecir correctamente el 91,6% de los casos, por lo que es un buen modelo. El análisis discriminante es uno de los mejores modelos para realizar este tipo de predicciones, pero para comprobar la eficacia de otros modelos y queriendo comprobar los resultados del artículo de Cinca, Cinca, C. S., & del Brío, B. M. (1993), me propongo realizar el análisis predictivo con otros modelos.

En el artículo citada arriba se utiliza un perceptrón multicapa de 9 X 10 X 1 y uno de 9 X 6 X 1, con los que obtienen resultados similares.

Debido a la antiguedad del artículo se utilizaron pocas neuronas y mucho tiempo de calculo, pero con los avances de hoy es algo que podemos realizar de manera mucho mas sencilla, con menos tiempo y muchas mas neuronas.

Debido a que he sido incapaz de ejecutar código de python en E markdown, pondré el codigo sin jecutar y mostraré los resultados a parte.

Red Neuronal profunda (o perceptrón multicapa)

Después de preparar los datos se creó un modelo con dos capas intermedias de la siguiente manera:

```
def create_baseline():
  model = Sequential()
  model.add(Dense(120,activation = 'relu'))
  model.add(Dense(60, activation = 'relu'))
  model.add(Dense(30, activation = 'relu'))
  model.add(Dense(1 , activation = 'sigmoid'))
  model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
  return model
```

Un resumen del modelo:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 120)               1200

 dense_1 (Dense)             (None, 60)                7260

 dense_2 (Dense)             (None, 30)                1830

 dense_3 (Dense)             (None, 1)                 31

=================================================================
Total params: 10,321
Trainable params: 10,321
Non-trainable params: 0
_________________________________________________________________
```

El siguiente paso es entrenar el modelo con los datos:

```
# Entrenamos el modelo
estimator = KerasClassifier(create_baseline, epochs=100, batch_size=32, verbose=0)
```

Para observar como se comporta el modelo a lo largo de las iteraciones podemos contruir gráficos sobre la pérdida y la precision del mismo a lo largo del tiempo. Se obtienen los siguientes gráficos:

```
plt.plot(loss.history['loss'])
plt.title('pérdida del modelo (entropía cruzada binaria)')
plt.ylabel('pérdida')
plt.xlabel('Iteración')
```

```
plt.plot(loss.history['accuracy'])
plt.title('Precisión del modelo')
plt.ylabel('precisión')
plt.xlabel('Iteración')
```

para comprobar mas precisamente el número de predicciones correctas utilizaremos un Iterador de validación cruzada con estratificación basada en etiquetas de clase, debido a que este método es el más óptimo para datos que tienen poco volumen y un imbalance en las clases, como es el caso.

Utilizaremos el Stratified k-fold, que funciona cogiendo muestras iguales con aproximadamente el mismo numero de casos positivos y negativos.

```
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(estimator, X, Y, cv=kfold)
print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

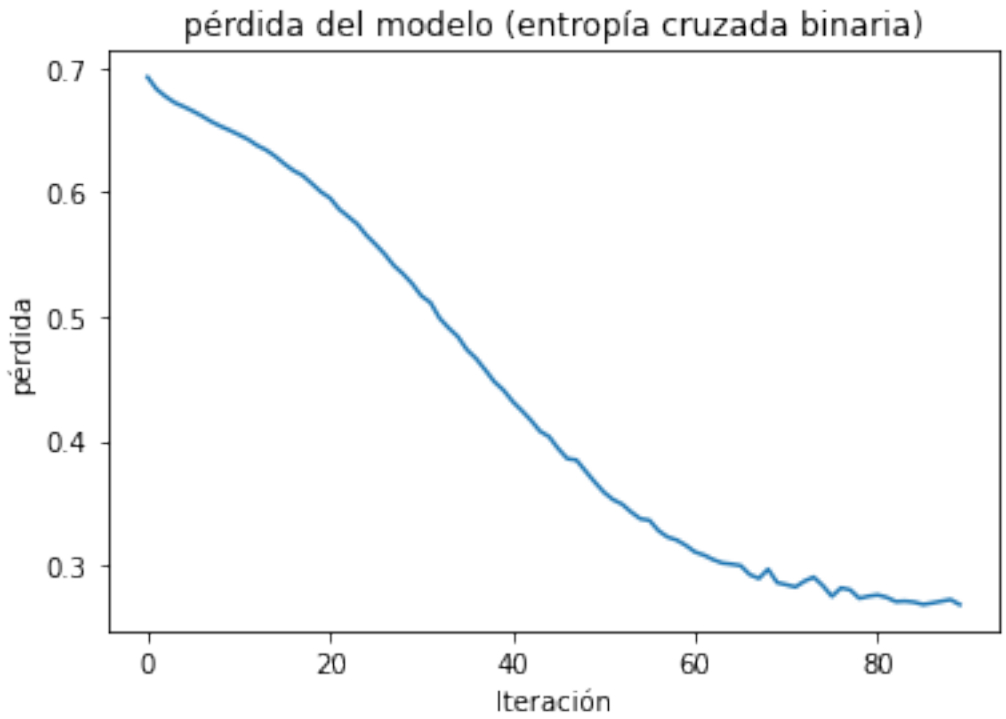


Figure 1: Pérdida del modelo a lo largo de las iteraciones

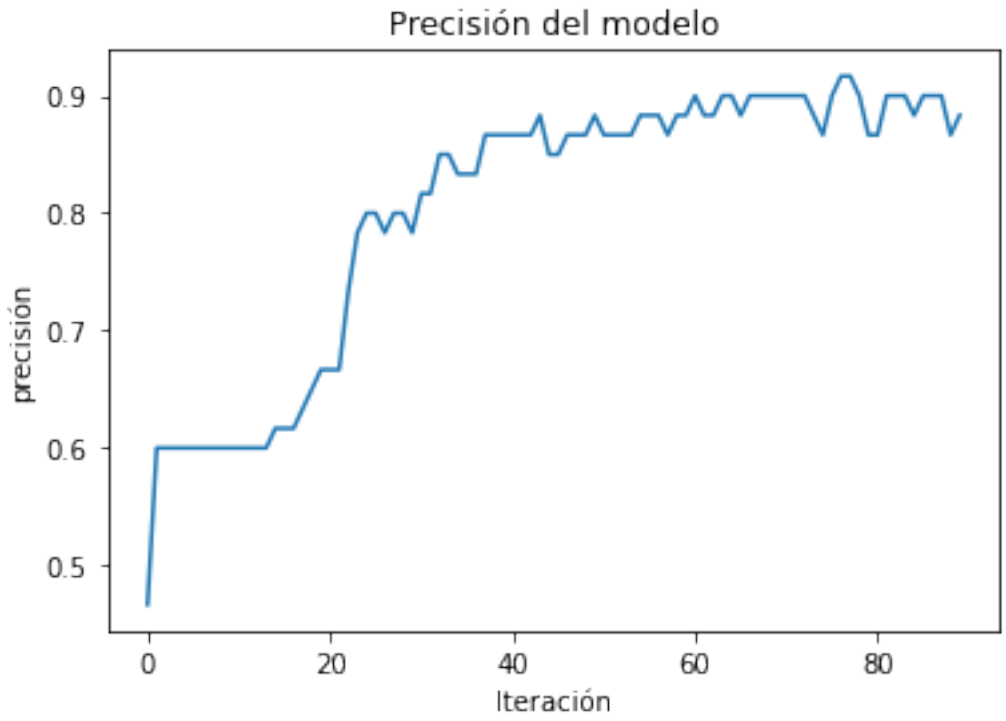


Figure 2: Precisión del modelo a lo largo de las iteraciones

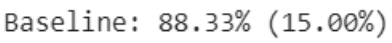


Figure 3: Media de la precisión y la desviación tipica

El output de este es la código nos devuelve la media de la precisión en las distintas iteraciones junto con la desviación tipica que acompaña a esta media, y se obtiene lo siguiente:

Observamos que el modelo tiene un 88,3% de precisión, lo que es menor que la precisión obtenida en el artículo en el que se usan muchas menos neuronas y capas, con técnicas mas anticuadas. Esto podría explicarse por varios motivos:

1. todos los modelos de redes neuronales tienen una naturalze estocástica, es decir, el mismo modelo con el mismo set de datos muy probablemente de resultados diferentes cada vez.
2. En segundo lugar en el artículo utilizn como medida de precisión unicamente el porcentaje de aciertos, en este caso estamos utilizando la validación cruzada y una media de este porcentaje de aciertos entre todas las iteraciones.
3. En toercer lugar, ese 94% de precision que citan en el artículo podría ser el mejor resultado que obtuvieron y no una media de todos ellos.

Regresión logística

Por útlimo utilizaremos un modelo de regresión logística en R para predecir los resultados y compraremos con los dos anteriores.

```
modelo <- glm(Quiebra ~ R1Liqui1 + R2Liqui2+ R3Liqui3 +R4Autof + R5RenEc + R6RenFin + R7Apal + R8CosVen
summary(modelo)
```

```
##
## Call:
## glm(formula = Quiebra ~ R1Liqui1 + R2Liqui2 + R3Liqui3 + R4Autof +
##     R5RenEc + R6RenFin + R7Apal + R8CosVen + R9Cash, family = "binomial",
##     data = bancos)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.83480  -0.06831  -0.00118   0.18586   1.05063
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    15.298     33.219   0.461   0.6451
## R1Liqui1     -317.340    238.063  -1.333   0.1825
## R2Liqui2       17.071     22.264   0.767   0.4432
## R3Liqui3      287.505    221.146   1.300   0.1936
## R4Autof       -51.766     67.599  -0.766   0.4438
## R5RenEc       439.809  11919.975   0.037   0.9706
## R6RenFin      -14.283     44.219  -0.323   0.7467
## R7Apal       -291.005  11005.797  -0.026   0.9789
## R8CosVen       -4.184     34.752  -0.120   0.9042
## R9Cash      -1203.486    617.390  -1.949   0.0513 .
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 80.761  on 59  degrees of freedom
## Residual deviance: 21.808  on 50  degrees of freedom
##   (6 observations deleted due to missingness)
## AIC: 41.808
##
## Number of Fisher Scoring iterations: 9
```

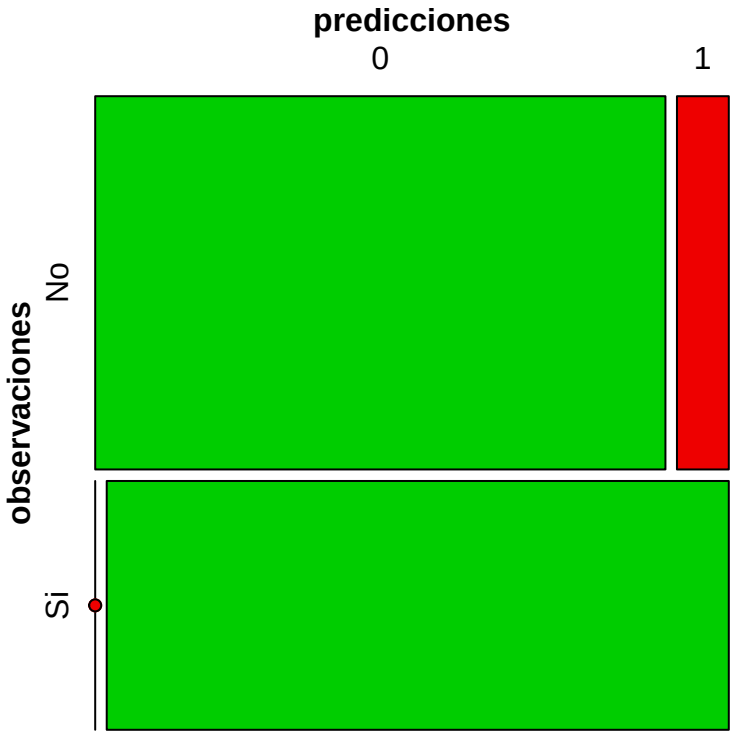
Después de crear el modelo observamos como se ha comportado, observando el nùmero de predicciones correctas:

```
library(vcd)
```

```
## Loading required package: grid
```

```
predicciones <- ifelse(test = modelo$fitted.values > 0.5, yes = 1, no = 0)
matriz_confusion <- table(modelo$model$Quiebra, predicciones,
                          dnn = c("observaciones", "predicciones"))


mosaic(matriz_confusion, shade = T, colorize = T,
       gp = gpar(fill = matrix(c("green3", "red2", "red2", "green3"), 2, 2)))
```



Observando el gráfico parece que el modelo es capaz de predecri correctamente casi la totalidad de los casos, para calcular su precisión:

```
precision_total = 100*sum(diag(matriz_confusion))/sum(matriz_confusion)
cat(precision_total,'%')
```

```
## 95 %
```

El modelo tiene una precisón del 95% que supera la de los modelos anteriores.

Conclusiones

Comparando la precisión de los 3 modelos observamos que el modelo logístico es el que mejor es capaz de predecir si el banco va a quebrar o no con un 95% de precisión, seguido del modelo discriminante con un 91% y finalmente el modelo de la red neuronal profunda con un 88%.

Por tanto podríamos concluir que el modelo logístico es el mejor para este conjunto de datos en concreto.