Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

Máquinas de aprendizaje

Reporte: Evaluación del rendimiento de un modelo, parte I

Docente: Abraham Sánchez López

Alumno	Matrícula
Taisen Romero Bañuelos	202055209

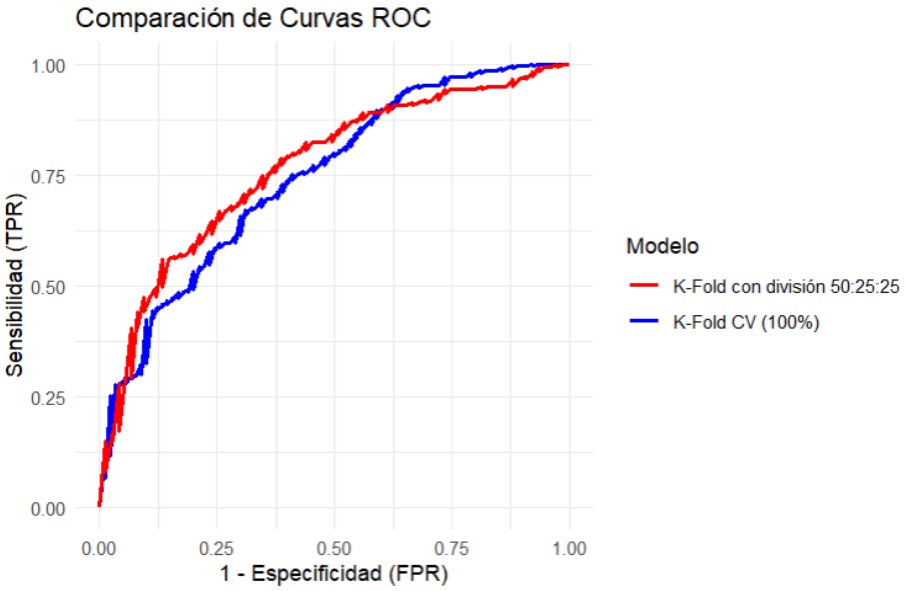
Evaluación de modelos

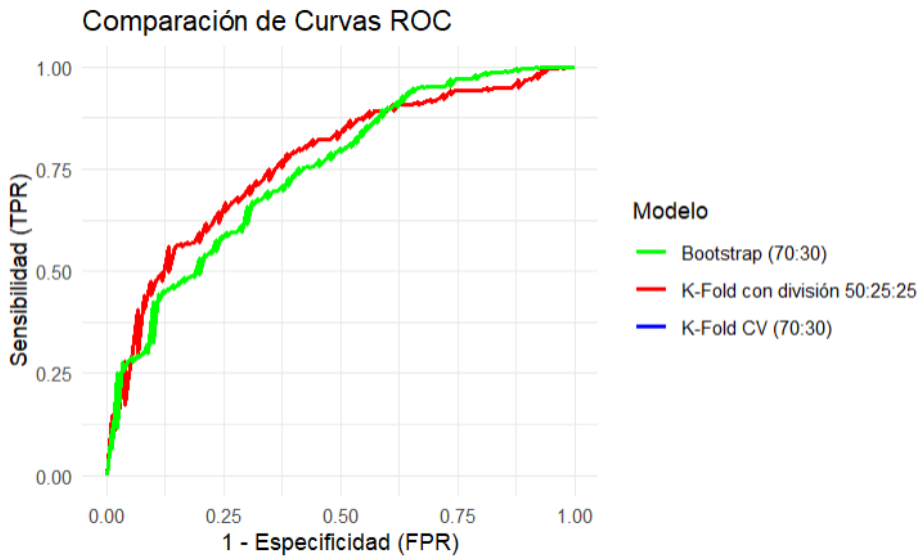
El documento trató de las diferentes formas de evaluar un modelo y también los engaños y las confusiones en que uno puede caer. En mi opinión, lo más importante fue lo relativo a las curvas ROC y al MCC, puesto que son las que se valoran más en la comunidad. Aunque siento que hizo falta explorar más aquellos casos en que un error es más penalizable que otro. Pero sea como sea, me aventuré a aplicar bootstraping con [70:30], k-Folds con [70:30] y k-Folds con [50:25:25] para hacer una comparación de sus curvas ROC.

Lo primero fue convertir las variables categóricas a factor y elegir la variable dependiente para todos nuestros modelos. Usé "default" como variable a predecir, pues nos indica si un cliente incumplió en un pago o no, cosa que seguro le interesará a los bancos.

El proceso fue repetitivo en los tres modelos, ponemos set.seed(), dividimos los datos, configuramos el trControl y evaluamos para generar la curva ROC. Aún con eso, el resultado vale la pena.

Nota: usé 100 muestras de bootstrap para que mi PC no tarde tanto en darme los resultados (aún así tardó un poco, lo que refuerza la idea de que es una metodología inconveniente para grandes conjuntos de datos).





Al parecer no se pueden imprimir 3 curvas a la vez, por lo que usaré la curva roja como punto de comparación. En cuanto a la interpretación de los gráficos podemos decir que entre la línea roja (k-Fold con 50:25:25) y la azul (k-Fold CV 70:30), la roja es mejor en la mayor parte de la curva ROC excepto en el tramo final, donde la línea azul la supera ligeramente.

Si recordamos, en el documento se habla de que es preferible elegir las curvas que sean mejores en los primeros tramos, ya que esto indica una mayor sensibilidad (capacidad para detectar correctamente los casos de default) con menores tasas de falsos positivos. Dado que el objetivo es predecir si un cliente incumplirá con sus pagos (default=yes) es necesario minimizar los falsos negativos, o sea, los casos en los que el modelo clasifica incorrectamente como default=no a un cliente que si va a incumplir. En este sentido, el modelo de la línea roja es mejor ya que es más efectivo detectando a los clientes que de verdad no pagan desde un umbral temprano de clasificación, lo cual sería bueno de cara a una estrategia temprana de mitigación de riesgos.

En cuanto a la línea roja vs la verde (k-Fold - 50:25:25 vs Bootstrap 70:30) se observa que Bootstrap tiene un desempeño inferior, ya que su curva se encuentra por debajo de la roja en la mayor parte del recorrido. Esto indica que Bootstrap no es tan eficaz como la validación cruzada en este caso, probablemente debido a la mayor varianza introducida por el remuestreo con reemplazo.

Finalmente, hablemos sobre las métricas de resumen de los modelos. En términos de sensibilidad, el modelo Bootstrap alcanza 0.8568, lo que significa que detecta mejor los casos de default. Sin embargo, su especificidad es menor (0.41), lo que sugiere que tiende a clasificar más clientes como incumplidos, aumentando los falsos positivos (esto refuerza lo comentado anteriormente sobre la curva ROC). Por otro lado, el

modelo k-Folds 50:25:25 equilibra mejor sensibilidad (0.8428) y especificidad (0.4666), haciéndolo más confiable en escenarios financieros porque minimiza los falsos negativos. Además, la matriz de confusión de este modelo muestra una precisión del 73.2%, lo que confirma que es el modelo más adecuado.

```
> print(model_bootstrap)
C5.0

700 samples
 16 predictor
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Bootstrapped (100 reps)
Summary of sample sizes: 700, 700, 700, 700, 700, 700, ...
Resampling results across tuning parameters:

  model  winnow  trials  ROC        Sens       Spec
  rules  FALSE    1      0.6507101  0.7789888  0.4722486
  rules  FALSE   10      0.7188990  0.8030091  0.4785471
  rules  FALSE   20      0.7313376  0.8159952  0.4749526
  rules   TRUE    1      0.6525022  0.7816424  0.4686234
  rules   TRUE   10      0.7147928  0.8081312  0.4669807
  rules   TRUE   20      0.7260736  0.8154449  0.4722977
  tree   FALSE    1      0.6594185  0.7861444  0.4361988
  tree   FALSE   10      0.7165822  0.8550171  0.4005782
  tree   FALSE   20      0.7274497  0.8568868  0.4100593
  tree    TRUE    1      0.6567897  0.7906482  0.4342970
  tree    TRUE   10      0.7094622  0.8460410  0.3992525
  tree    TRUE   20      0.7212246  0.8494082  0.4013592

ROC was used to select the optimal model using the largest value.
The final values used for the model were trials = 20, model = rules and winnow = FALSE.
```

```
> print(model_kFolds)
C5.0

700 samples
 16 predictor
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 630, 630, 630, 630, 630, 630, ...
Resampling results across tuning parameters:

  model  winnow  trials  ROC        Sens       Spec
  rules  FALSE    1      0.6763362  0.8367347  0.4714286
  rules  FALSE   10      0.7158892  0.8326531  0.4380952
  rules  FALSE   20      0.7249271  0.8489796  0.4380952
  rules   TRUE    1      0.6428086  0.8469388  0.4095238
  rules   TRUE   10      0.7000486  0.8306122  0.4380952
  rules   TRUE   20      0.7104956  0.8367347  0.4571429
  tree   FALSE    1      0.6429543  0.8326531  0.4285714
  tree   FALSE   10      0.7028183  0.8469388  0.3238095
  tree   FALSE   20      0.7081633  0.8571429  0.3714286
  tree    TRUE    1      0.6667638  0.8265306  0.4190476
  tree    TRUE   10      0.7026725  0.8551020  0.4190476
  tree    TRUE   20      0.6958212  0.8530612  0.4142857

ROC was used to select the optimal model using the largest value.
The final values used for the model were trials = 20, model = rules and winnow = FALSE.
```

```
> print(model_kF_retencion)  # Resultados de k-fold CV
C5.0

500 samples
 16 predictor
  2 classes: 'no', 'yes'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 450, 450, 450, 450, 450, 450, ...
Resampling results across tuning parameters:

  model  winnow  trials  ROC        Sens       Spec
  rules  FALSE    1      0.6175238  0.8200000  0.3933333
  rules  FALSE   10      0.7386667  0.8114286  0.5000000
  rules  FALSE   20      0.7467619  0.8171429  0.4733333
  rules   TRUE    1      0.6123810  0.8085714  0.3066667
  rules   TRUE   10      0.7099048  0.8085714  0.4000000
  rules   TRUE   20      0.7231429  0.8028571  0.4533333
  tree   FALSE    1      0.6617143  0.8057143  0.4133333
  tree   FALSE   10      0.7326667  0.8314286  0.4600000
  tree   FALSE   20      0.7465714  0.8428571  0.4666667
  tree    TRUE    1      0.6588571  0.7942857  0.3800000
  tree    TRUE   10      0.7107619  0.8228571  0.4200000
  tree    TRUE   20      0.7160952  0.8142857  0.4333333

ROC was used to select the optimal model using the largest value.
The final values used for the model were trials = 20, model = rules and winnow = FALSE.
```

```
> print(conf_matrix_test)  # Evaluación en prueba
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  140  32
       yes  35  43

               Accuracy : 0.732
                 95% CI : (0.6725, 0.7859)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.1501

                  Kappa : 0.3691

 Mcnemar's Test P-Value : 0.8070

            Sensitivity : 0.8000
            Specificity : 0.5733
         Pos Pred Value : 0.8140
         Neg Pred Value : 0.5513
             Prevalence : 0.7000
         Detection Rate : 0.5600
   Detection Prevalence : 0.6880
      Balanced Accuracy : 0.6867

       'Positive' Class : no
```

```
> print(conf_matrix_valid)  # Evaluación en validación
Confusion Matrix and Statistics

          Reference
Prediction  no yes
       no  139  33
       yes  36  42

               Accuracy : 0.724
                 95% CI : (0.6641, 0.7785)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.2251

                  Kappa : 0.3503

 Mcnemar's Test P-Value : 0.8097

            Sensitivity : 0.7943
            Specificity : 0.5600
         Pos Pred Value : 0.8081
         Neg Pred Value : 0.5385
             Prevalence : 0.7000
         Detection Rate : 0.5560
   Detection Prevalence : 0.6880
      Balanced Accuracy : 0.6771

       'Positive' Class : no
```