



MÁSTER UNIVERSITARIO OFICIAL

UNIVERSIDAD INTERNACIONAL MENÉNDEZ PELAYO

Práctica: Entrega Final

Ciencia de Datos y Aprendizaje Automático

Raúl Fauste

100013220@alumnos.uimp.es

January 11, 2026

1 Introducción y descripción del problema

La fuga de clientes constituye uno de los mayores desafíos para las instituciones financieras, ya que adquirir un nuevo cliente suele resultar mucho más costoso que retener a uno existente. Además, cuando un cliente decide cerrar su cuenta, la entidad pierde de manera irreversible la oportunidad de desplegar campañas de fidelización o de ofrecer productos alternativos que podrían haber evitado su salida. Este fenómeno no solo afecta los ingresos directos del banco, sino también su estabilidad a largo plazo.

En este contexto, la capacidad de predecir con antelación qué clientes tienen mayor probabilidad de abandonar la entidad se convierte en una herramienta esencial permitiendo focalizar esfuerzos y optimizar recursos hacia los clientes necesarios.

Para ello, el trabajo consiste en, dado un *dataset* formado por 10000 instancias y 14 variables que se centran en la descripción de multitud de aspectos de cada uno de los clientes de un banco, elaborar un modelo de aprendizaje automático que obtenga una estimación lo más precisa de la variable Exited. La elaboración del modelo ha de seguir las buenas prácticas a lo largo de la asignatura de forma que toda la metodología adquiere gran relevancia.

El conjunto de datos de entrenamiento consta de 8000 instancias. Por otra parte, el conjunto de datos de prueba, está formado por 2000 instancias y se utilizará para realizar la predicción. Todo el *dataset* está disponible en la plataforma Kaggle, plataforma popular entre científicos de datos que compiten por ver quién obtiene las mejores predicciones.

Para la evaluación del trabajo, será necesario subir a Kaggle los resultados de forma que se pueda comparar el trabajo realizado con el de otros científicos de datos.

2 Comprensión y preprocessado de datos

En esta sección se realizará una exploración de los datos para tratar de prepararlos de la mejor forma posible de cara al modelado. El primer paso consiste en ir a Kaggle y descargar las instancias. Una vez hecho esto podemos almacenarlas en variables para poder trabajar con ellas.

Además, para la posterior validación de los modelos entrenados, se divide el conjunto de intancias de entrenamiento en un ratio 80/20, 80% de instancias para entrenamiento y 20% para validación.

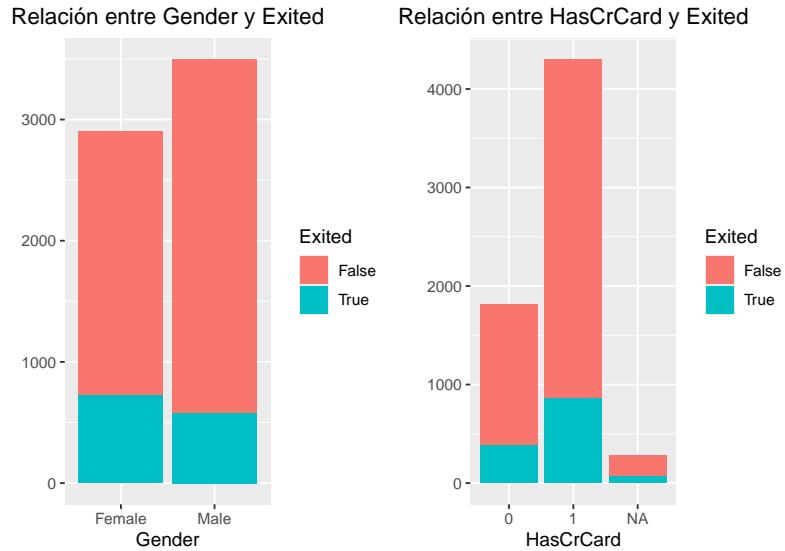
A la hora de leer las instancias en *RStudio* existe un problema con los apóstrofes. Por tanto, se debe incluir el parámetro *quote* = “ “, para ignorarlos, en la función *read.table()* para poder leer la totalidad de instancias de forma correcta.

```
## [1] "Número de instancias de entrenamiento 6400"  
  
## [1] "Número de instancias de validacion 1600"  
  
## [1] "Número de instancias de test 2000"
```

2.1 Exploración y visualización de datos

En esta subsección, se trata de conocer y visualizar las variables tanto categóricas como numéricas, de forma que se puedan intuir patrones y distribuciones.

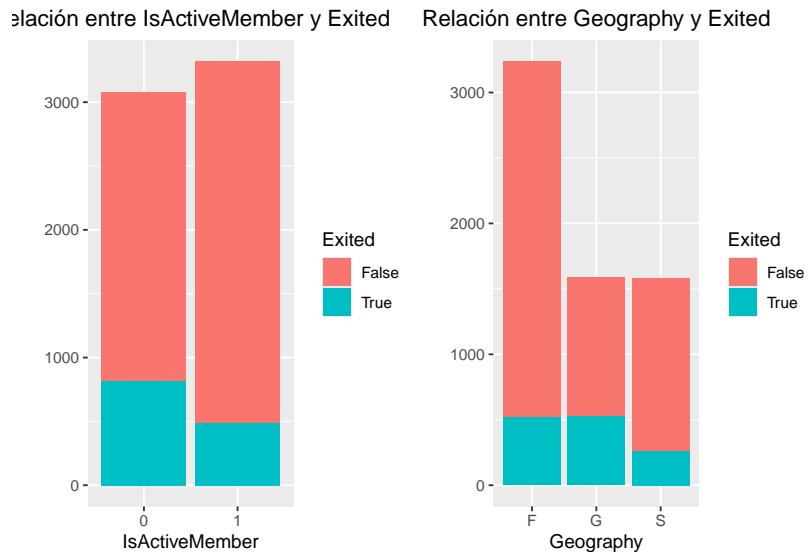
Para comenzar, se analizan las variables categóricas junto con su repercusión a la clase objetivo. Para ello, se eligen gráficos de barras de forma que se pueda ver si el valor de un atributo concreto afecta directamente al abandono.



```
## [1] "Valores faltantes del atributo Gender: 0"
```

```
## [1] "Valores faltantes del atributo HasCrCard: 286"
```

Del atributo *Gender* a primera vista no se puede sacar ninguna conclusión en claro. Sin embargo, del atributo *HasCrCard* se observa que hay una pequeña cantidad de instancias que no tienen su valor asociado. Esto implica que habrá que tomar alguna decisión al respecto con cómo actuar con dichas instancias.



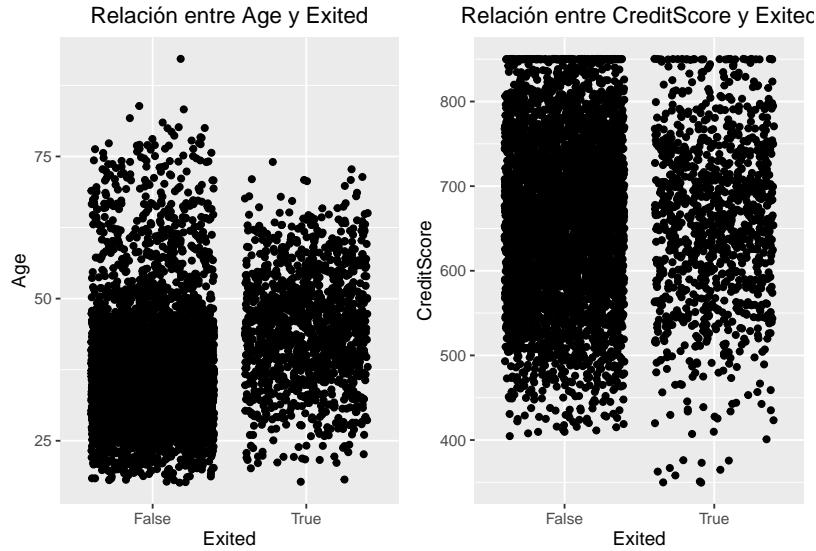
```
## [1] "Valores faltantes del atributo IsActiveMember: 0"
```

```
## [1] "Valores faltantes del atributo Geography: 0"
```

```
## [1] "Valores faltantes del atributo Exited: 0"
```

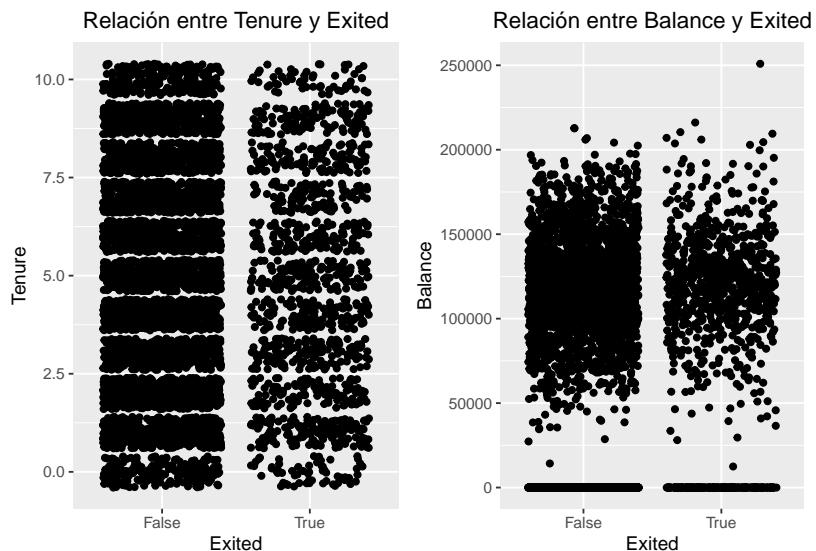
En el resto de variables categóricas, ocurre lo mismo que en el caso del atributo *Gender*, no se puede sacar ninguna observación en claro. Además, no hay valores faltantes en su caso, incluida la variable objetivo.

Por otro lado, para el caso de las variables numéricas, la exploración se realiza con diagramas de dispersión centradas en la variable objetivo *Exited*.



```
## [1] "Valores faltantes del atributo Age: 0"
## [1] "Valores faltantes del atributo CreditScore: 1317"
```

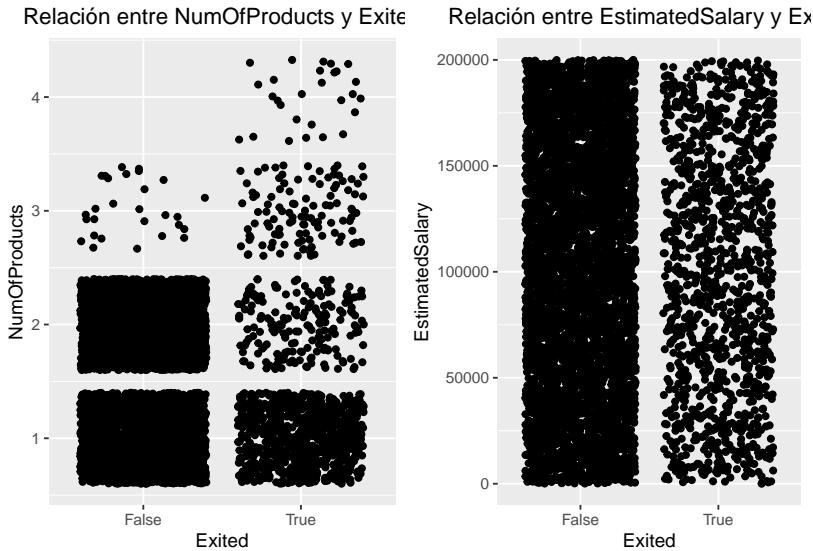
En el caso de las variables *Age* y *CreditScore* se pueden intuir intervalos para los cuales la tasa de abandono es mayor, por lo que, quizás sea interesante discretizar estas variables para poder obtener una mayor información al respecto. Destacar que del atributo *CreditScore* existe un gran número de instancias que no tienen un valor asociado, lo que implica que habrá que tomar una decisión al respecto conforme actuar.



```
## [1] "Valores faltantes del atributo Tenure: 0"
```

```
## [1] "Valores faltantes del atributo Balance: 1302"
```

Con la variable *Balance* ocurre lo mismo que con los atributos *Age* y *CreditScore* por lo que quizá discretizar esta variable es conveniente. Sin embargo, del atributo *Tenure* a primera vista, no se puede sacar nada en claro. Resaltar que del atributo *Balance* también hay un gran número de instancias sin un valor asociado.



```
## [1] "Valores faltantes del atributo NumOfProducts: 800"
```

```
## [1] "Valores faltantes del atributo EstimatedSalary: 672"
```

Con estas dos últimas variables, *NumOfProducts* y *EstimatedSalary*, se puede observar que es mucho más frecuente el caso de no abandono del banco. Con todo será necesario un análisis más exhaustivo. Por otro lado, destacar que hay instancias que no tienen un valor asociado a estos atributos, matiz a tener en cuenta para decir cómo actuar.

2.2 Limpieza de datos

En esta sección, se trata de pulir los datos disponibles una vez se tiene una comprensión global de estos. En la sección anterior se ha omitido la exploración de dos atributos: *CustomerId* y *Surname*. Esto se ha hecho intencionadamente, ya que estas dos variables no aportan ninguna información sobre los motivos por los que un cliente puede decidir abandonar o no el banco. Por este motivo, lo único que pueden hacer es introducir ruido en las predicciones. Para prevenir esto, se procede a la eliminación de dichas columnas.

```
## [1] "Variables restantes: 11"
```

Tras la eliminación de las variables irrelevantes, se procede a tratar el problema de los valores faltantes. En la sección anterior, hemos visto que las variables que proporcionan la información financiera (*CreditScore*, *Balance*, *NumOfProducts*, *HasCrCard* y *EstimatedSalary*), tienen valores faltantes en ciertas instancias. Esta problemática, exige decisiones de cómo tratar esto ya que existen modelos que no son resistentes a valores faltantes.

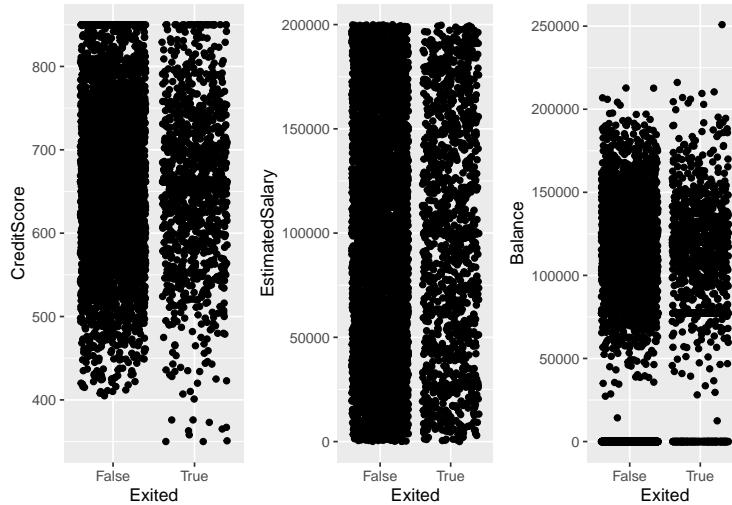
El criterio a seguir para cómo tratar los valores faltantes en cada variable difiere. En el caso de la variable *HasCrCard*, una variable categórica, solamente un 4,7% de las instancias tiene valores faltantes. Una opción

podría ser eliminar dichas instancias, pero al ser una instancia categórica con dos valores posibles, se opta por asignar a dichas instancias el valor más probable. Este valor es 1 (67,35% de instancias poseen este valor), denotando que el cliente sí posee una tarjeta de crédito.

```
## [1] "Valores con HasCrCard = 1 antes de la imputación: 4301"
```

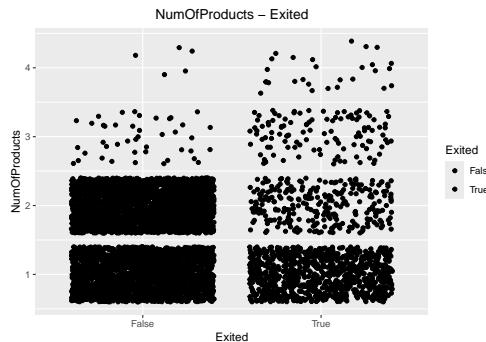
```
## [1] "Valores con HasCrCard = 1 después de la imputación: 4587"
```

En el caso de las variables *CreditScore*, *Balance*, *EstimatedSalary* el criterio para tratar los valores faltantes varía. El número de valores faltantes es mayor que en el caso anterior, por lo que la eliminación no es una opción por la pérdida de información que supondría. Por tanto, al ser valores numéricos donde en ninguna de las tres variables se observa una distribución clara, se opta por imputar la media de cada atributo a los valores faltantes en cada caso.



Si se comparan los diagramas de dispersión con los de la sección anterior se puede deducir que se ha mantenido la distribución de los datos originales, lo cual no implica la alteración de la información disponible.

Por último, para el caso de la variable *NumOfProducts* el criterio de imputación es totalmente distinto. En el gráfico de la sección anterior se puede observar que dependiendo del valor de este atributo, es mucho más probable abandonar el banco (sobretodo si *NumOfProducts* = 1). Por lo tanto, la estrategia aquí seguida es una imputación aleatoria basada en la distribución observada. Es decir, se asigna un valor en función de la distribución que ya se tiene de los valores que no son nulos. De esta forma, se respetan tanto la variabilidad como distribución original de los datos. Técnica más adecuada en este caso, que asignar la moda u opción más probable ya que puede alterar bastante la información (hay un 12% de valores faltantes para este atributo).



En comparación con el diagrama de la sección anterior se puede deducir que se ha mantenido la distribución de los datos originales. Por tanto, se preserva la información de la que se dispone.

En cuanto al tratamiento de *outliers*, observando las gráficas, se observan valores atípicos en alguna variable numérica. Es el caso de la variable *Age* con valores superiores a los 80 años. Estos valores son atípicos pero asumibles, ya que es posible que una persona de edad avanzada pueda tener una cuenta bancaria. Lo mismo ocurre con la variable *Balance*, donde se encuentran valores altos, pero no quita que no sean realistas. Por lo tanto, se ha optado por mantener los valores atípicos para no perder representatividad en segmentos críticos.

2.3 Procesado de variables categóricas

En esta sección se procesan las variables categóricas de forma que sean totalmente intepretables por los futuros modelos. Para ello, hay que diferenciar entre las variables categóricas binarias y no binarias. En el caso de las binarias, *HasCrCard* y *IsActiveMember* ya vienen representadas de forma binaria, luego no hay que modificar nada. Sin embargo, en el caso de la variable *Gender* hay que modificar los valores. Esto se debe a que un modelo no entiende un texto plano con *Male* o *Female*. Por lo que hay que cambiar el valor de *Male* o *Female* por 0 o 1, de forma que un modelo pueda procesar de forma correcta esta variable. La decisión elegida, es denotar con 1 aquellos clientes masculinos y con 0 el otro caso.

Por otro lado está la variable *Geography* que toma como valores *Spain*, *France* y *Germany*. Ocurre algo parecido, un modelo no es capaz de entender un texto plano con el nombre del país, por lo que hay que renombrar utilizando la técnica One-hot encoding. De esta manera, se tienen dos columnas por cada país, de forma que indican con un 1 si pertenece a un país concreto y con un cero en caso contrario. Se elimina una columna, ya que si no es de dos países concretos, obligatoriamente será del tercero.

Nuevas columnas tras el Procesamiento:

```
## [1] "CreditScore"      "Gender"          "Age"            "Tenure"          "Tenure"
## [5] "Balance"           "NumOfProducts"    "HasCrCard"       "IsActiveMember"  "IsActiveMember"
## [9] "EstimatedSalary"   "Exited"          "Geography_G"     "Geography_S"
```

2.4 Procesado de variables numéricas

En esta sección, se trata de procesar las variables numéricas y de describir correlaciones entre sí o con la variable objetivo.

Para ello, antes de procesar las variables numéricas, vamos a estudiar las correlaciones. Dicho esto, es necesario hacer que *Exited* tome como valores 1 si abandonó y 0 en caso contrario. Una vez hecho esto se calcula la correlación de cada variable numérica con *Exited*, obteniendo:

```
##      CreditScore          Age          Tenure        Balance      NumOfProducts
##      -0.017         0.290      -0.032         0.112      -0.080
##      EstimatedSalary      Exited
##      0.011         1.000
```

No se observa ninguna correlación lineal clara. Sin embargo si se puede deducir que la que más influye quizá sea la edad. A mayor edad, mayor posibilidad de abandono. De todas formas, la ausencia de correlaciones lineales fuertes en el resto de variables justifica aún más el uso de técnicas como discretización que sean capaces de capturar relaciones no lineales.

Se pueden estudiar correlaciones entre variables contínuas. Por ejemplo, es sensato pensar que cuanta más edad (*Age*) tiene una persona más dinero puede acumular en su cuenta bancaria (*Balance*), mayor salario puede tener (*EstimatedSalary*), por su experiencia y más años como cliente del banco puede llevar (*Tenure*). Por lo que se estudia la correlación entre estas tres variables.

```
##          Age Tenure Balance EstimatedSalary
## Age      1.000 -0.015  0.024      -0.006
## Tenure   -0.015  1.000 -0.006       0.014
## Balance   0.024 -0.006  1.000       0.017
## EstimatedSalary -0.006  0.014  0.017      1.000
```

Se observa que no existe una relación lineal entre las variables. Luego nuestras hipótesis no se cumplen. Por otro lado, esto es positivo ya que hará que el modelo sea robusto ya que no existe la multicolinealidad.

Una vez estudiadas las posibles relaciones entre variables numéricas, se procede a discretizar algunas de estas. En la primera sección, se comentó que podía ser interesante discretizar las variables *Age*, *CreditScore*, *Balance* y *NumOfProducts* ya que se podían intuir intervalos donde la tasa de abandono era más elevada. Por ello, se procede a la discretización de dichas variables.

```
## [1] "Puntos de corte encontrados en Age: c(38.5, 42.5, 45.5, 65.5)"
## [1] "Punto de corte encontrado en CreditScore: 403"
## [1] "Punto de corte encontrado en Balance: 6229.595"
## [1] "Puntos de corte encontrados en NumOfProducts: c(1.5, 2.5)"
```

Para la discretización se utiliza el método MDLP, algoritmo de discretización supervisado, que utiliza la variable *Exited* para encontrar los puntos de corte que maximizan la ganancia de información (reducción de entropía).

2.5 Construcción, formateo y estandarización de variables

En esta sección se procede a la estandarización de variables. Actualmente, se ha discretizado las variables *Age*, *CreditScore*, *Balance* y *NumOfProducts* de forma que se comportan como categóricas. Sin embargo, aún hay variables numéricas como *EstimatedSalary* (tomando valores hasta 200.000) o *Tenure* (tomando valores de 0 a 10) que tienen magnitudes muy dispares.

Muchos algoritmos de aprendizaje automático basados en distancias o pesos, son muy sensibles a la magnitud de los datos. De esta forma, si no se estandarizan los datos el modelo podría dar más importancia a un atributo que a otro simplemente por la magnitud de sus datos, sesgando así el aprendizaje.

La técnica de estandarización más común es el Z-score, que transforma los datos para que tengan una media de 0 y una desviación típica de 1.

2.6 Selección horizontal o vertical

En esta sección, se analiza la importancia de cada atributo respecto a la clase objetivo. Realmente, esta sección ya ha tenido lugar cuando se ha realizado la limpieza de las variables irrelevantes *Surname* y *CustomerId* (selección vertical). Sin embargo, aquí se busca estudiar si las variables restantes tienen una relevancia significativa.

Para ello, mediante el algoritmo *RandomForest* se va a analizar la importancia de cada variable, la ganancia de información, el poder de discriminación. Además, se van a poder detectar relaciones no lineales que han sido imposibles de deducir mediante la matriz de correlación. Cabe destacar que aunque el algoritmo *RandomForest* no se ha visto en la asignatura, es uno de los más usados en Ciencia de Datos, se basa en cientos de árboles que mediante una votación establecen la categoría de una instancia concreta. Esto nos permite medir la ganancia de información. Por lo tanto, si aplicamos el modelo al conjunto de datos de entrenamiento, se obtiene:

```
##          Age EstimatedSalary  NumOfProducts      Tenure IsActiveMember
## 315.226783    243.911869    182.245033 123.033292      75.668955
## Geography_G       Gender        Balance HasCrCard Geography_S
## 60.159908    35.011404    31.778722   30.626087     22.562034
## CreditScore
##      5.522612
```

De los datos arriba mostrados, se pueden extraer muchas conclusiones. La primera y la que más interesa en esta sección, es que todos los atributos tienen cierto nivel de relevancia. Se puede observar como *Age*, *EstimatedSalary* o *NumOfProducts* son los que más ganancia de información proporcionan. Sin embargo, variables como *CreditScore* reportan un valor menor, pero aun así lejano de 0, por lo cual es una buena razón para mantener todo el conjunto de variables actual de forma que no se pierda información. Se pueden deducir relaciones no lineales entre los atributos y la variable *Exited*. En la sección de procesado de variables numéricas, la variable *Age* era la que mostraba una mayor relación lineal con *Exited*, por lo que se puede entender porque sea la que mayor valor de ganancia reporte ahora. Por otro lado, las variables *EstimatedSalary* y *NumOfProducts* reportaban un valor de colinealidad prácticamente nulo, pero aquí se les detecta como atributos cruciales, denotando la existencia de una relación no lineal entre estas variables y la variable *Exited*.

3 Modelado

Una vez se ha hecho el procesado de los datos, en esta sección se busca encontrar el modelo que maximiza el F1-score, métrica para evaluar nuestro proyecto. Para ello, se va a realizar una comparativa entre las cuatro técnicas vistas en la asignatura: *Naive Bayes*, *Árboles de decisión*, *Redes Neuronales* y *Nearest Neighbors*. Se entrena los cuatro modelos mencionados (“nb”, “rpart”, “nn” y “knn”) haciendo énfasis en la métrica F1-score. El entrenamiento se lleva a cabo mediante validación cruzada de 10 pliegues. Esta decisión se basa principalmente en tratar de evitar el posible sobreajuste de los modelos a los conjuntos de entrenamiento.

Tras el entrenamiento de los cuatro modelos, el que se verá afectado por la calidad del procesado de datos, se pasa a la fase de validación. Esta fase consiste en analizar qué modelo de los 4 entrenados tiene un mayor rendimiento. La validación se lleva a cabo con el subconjunto de las instancias de entrenamiento apartadas al inicio. Se obtienen los siguientes resultados:

| Modelo | Accuracy | F1-Score |
|---------------------|----------|----------|
| Naive Bayes | 0.71 | 0.51 |
| Árboles de decisión | 0.66 | 0.40 |
| Red Neuronal | 0.65 | 0.53 |
| Nearest Neighbors | 0.64 | 0.45 |

Se observa que los modelos que mejor comportamiento tienen son Naive Bayes (“nb”) y el de Red Neuronal (“nn”). Hay que resaltar que aunque Naive Bayes tenga una mayor precisión, el F1-score es menor que en el caso de la Red Neuronal. Por lo tanto, al ser el F1-score la métrica de rendimiento, se va a considerar al modelo de Red Neuronal el ganador de esta comparativa. Algunas de las posibles razones de por qué obtiene un mayor valor de F1-score son:

- Es capaz de detectar interacciones complejas y no lineales entre los atributos.
- Las redes neuronales son sensibles a la escala de los datos. El escalado y centrado permite que el algoritmo converga de forma eficiente.

Tras analizar y definir la técnica a utilizar, se pasa a la fase de evaluación.

4 Evaluación

En esta sección se trata de evaluar el modelo ganador visto en la sección anterior. A la hora de la evaluación, se utiliza el conjunto de instancias de test definido al inicio del documento. Simplemente, tenemos que ejecutar el modelo ganador, en este caso, el basado en Redes Neuronales (“nn”) concretamente, en las instancias de test.

Se introduce un pequeño matiz. Normalmente, en casos donde se tiene desbalanceo en la clase objetivo, el modelo tiende a predecir la clase mayoritaria (No abandono). Para ello, entrenamos los modelos guardando el porcentaje de seguridad de su predicción. De esta forma, se determina que: si la probabilidad de que un cliente abandone el banco es de 0.45, se considerará que el cliente abandona. Tratando de contrarrestar así el conservadurismo del modelo.

Tras realizar la evaluación, hay que exportar las predicciones realizadas por el modelo a un fichero *.csv* para poder subirlo a la plataforma *Kaggle* y poder comparar los resultados con el resto de participantes en el concurso de este problema.

5 Conclusiones

Una vez finalizado el trabajo, se pueden comentar las conclusiones acerca de él. Pese a no poder detallarlo en la memoria por la restricción de páginas, con cada limpieza de datos realizada en el procesado, los resultados han ido mejorando poco a poco, lo cual hace resaltar la importancia de una buena preparación de los datos antes de lanzarse a la fase de modelado.

Por otro lado, la primera inspección y exploración de las variables con las que se va a trabajar resulta muy útil para intuir ciertas distribuciones o variables candidatas a ser discretizadas.

Personalmente, ha sido mi primer proyecto de ciencia de datos, si se puede considerar, y ha sido realmente enriquecedor. Me quedo con ganas de seguir aprendiendo más y mejorando poco a poco. Estoy contento de haber aprendido las bases, como una buena limpieza de datos es crucial antes de pasar a modelar, cómo elegir un buena métrica de evaluación y sobre todo cómo evitar el sobreajuste (separación de instancias, no filtrar información, etc.). Todo lo mencionado, son conceptos asentados gracias a la realización de este trabajo.

6 Resultados en Kaggle

En esta sección, se detallan tanto el usuario de Kaggle como el F1-score obtenido. Además, el enlace al repositorio donde se puede encontrar el código con el que se ha obtenido el resultado mostrado a continuación.

<https://github.com/raulfauste/EntregaFinalCienciaDatos>

11 Raul Fauste



0.60425

12 9m

El usuario es mi propio nombre **Raúl Fauste** y la mejor puntuación de F1-score es 0.60425. En el momento de la captura de pantalla me encuentro situado en la posición número 11 del ranking público.

En el enlace al repositorio donde ver el código elaborado, se puede observar cómo se ha encapsulado todo evitando la filtración de datos.