Universidad de Oviedo

Máster Universitario en Análisis de Datos para la Inteligencia de Negocios

Trabajo fin de Máster

Comparativa de técnicas de balanceo de datos. Aplicación a un caso real para la predicción de fuga de clientes.

Autor:
Joaquín García Abad

Tutores:
Julia Lastra García
Carlos Carleos Artime

Mayo de 2021

Joaquín García Abad
Máster en Análisis de Datos para la Inteligencia de los Negocios

Resumen:

La predicción de la fuga de clientes se ha convertido en un elemento importante en áreas de negocio en la que la rotación de clientes es muy alta. Además la fuga de clientes se presenta como un área de estudio normalmente afectada por el problema del desbalanceo de datos, en los que una clase tiene un peso mucho mayor que otra en la muestra. En problemas de clasificación o predicción de clase el desbalanceo de datos puede incurrir en una merma de la capacidad predictiva de los modelos en la clase de interés, normalmente vinculada a la clase minoritaria.

Este trabajo se centra en valorar la contribución de distintas técnicas de balanceo de datos en la capacidad predictiva de la fuga de clientes. Concretamente se analiza el impacto de ocho técnicas de balanceo de datos sobre tres clasificadores distintos (regresión logística, bosques aleatorios y gradient boosting), para estudiar la capacidad predictiva de los modelos en un contexto de datos desbalanceados.

Para el ejemplo concreto sometido a estudio puede afirmarse que las mejores contribuciones para la detección de la fuga de clientes son las aportadas por métodos de balanceo de datos no heurísticos o aleatorios con bastante diferencia respecto al resto de técnicas revisadas. En el nivel de las técnicas de clasificación son los bosques aleatorios la técnica que mejor resultados arroja.

Palabras clave: desbalanceo de datos, predicción, fuga de clientes, clasificación, técnicas de balanceo.

Abstract:

Churn prediction problem has become an important element in business areas with high customer rotation. Furthermore, churn is normally affected by the imbalanced data problem, in which one class has a much greater weight than another in the sample. In classification or prediction problems, imbalanced data may reduce the predictive power of the models for the class of interest, normally the minority class.

This work focuses on assessing the contribution of different data balancing techniques in the churn prediction. Specifically, this work analyses the impact of eight data balancing techniques over three different classifiers (logistic regression, random forests and gradient boosting) within an imbalanced data context.

For the specific example of this study, it can be confirmed that the best peformance for churn detection is provided by non-heuristic data balancing techniques, with quite better results compared to the rest of techniques. At the classification level, random forest is the technique that provides the best performances.

Key words: imbalanced data, churn prediction, classification, balance techniques.

Índice

1. Introducción.

El problema del desbalanceo de datos o desbalanceo de clases es un problema común que aparece cuando en una muestra dividida o categorizada en grupos, una de las clases o grupos es dominante, es decir, cuando una de las clases tiene un número de individuos de la muestra mucho más alto que la otra. Esta distribución de la muestra es recurrente en muchos problemas como por ejemplo el fraude donde la mayoría de elementos de la muestra no representa un fraude, estudios clínicos donde la mayoría de la muestra no presenta la patología a estudio o en el caso de la **fuga de clientes** o ***churn*** (por su terminología inglesa) donde la mayoría de los clientes continúan en la compañía. Este tipo de desbalanceos en la muestra se convierte a menudo en un problema cuando se abordan trabajos de clasificación, predicción de clase o de prevalencia del evento asociado (fraude, patología o fuga de cliente en los ejemplos anteriores). Lo que sucede en este tipo de trabajos es que estadísticamente la clase que tiene un mayor peso tiende a acumular un mayor número de clasificaciones sin afectar de manera determinante al resultado global de la clasificación. Este problema será más importante cuanto mayor sea el desbalanceo entre las clases. Si imaginamos una muestra ficticia de 100 elementos en la que tan sólo un elemento presenta la característica de interés si predecimos tan solo la clase mayoritaria (la que no presenta la característica) estaríamos siendo capaces de clasificar correctamente el 99% de los casos. En estos contextos de desbalanceo de los datos, nos encontramos de forma habitual ajustes globales muy altos por concentrarse casi exclusivamente en la clase mayoritaria. Para enfrentar este tipo de problemas, una solución habitual es la aplicación de técnicas de **balanceo de datos** o ***resampling*** (por su terminología inglesa) que modifican la distribución original de la muestra ya sea eliminando casos o instancias de la clase mayoritaria (***undersampling***) o replicando o creando nuevas instancias de la clase minoritaria (***oversampling***).

El presente trabajo enfrenta el problema del desbalanceo de datos en el contexto de la fuga de clientes. Concretamente el problema en el que se desarrolla este trabajo es la predicción de la fuga de clientes para un servicio de música en *streaming* (KKBox). Los datos con los que se trabaja provienen de este servicio, especialmente importante en Asia y que fueron proporcionados públicamente en el marco de un concurso en el que el objetivo era precisamente predecir la fuga de clientes. El estudio que se pretende en este trabajo es evaluar la capacidad predictiva de distintos modelos de clasificación empleando diferentes técnicas de balanceo de datos. Para ello partimos de dos premisas inscritas en los datos y comunes en problemas de este tipo, como ya se ha avanzado:

- **Una de las clases es dominante** y acumula el mayor número de casos en la muestra.
- La **clase minoritaria es la clase que contiene o presenta la característica de interés** en el estudio (la fuga del cliente en este caso concreto).

En nuestro caso concreto la característica o condición que se pretende predecir es si un cliente va a fugarse o no de los servicios que oferte la compañía. La clase mayoritaria la representan los clientes no fugados y la minoritaria la de los clientes fugados, en una relación aproximada de 8:100. No es este un ejemplo de desbalanceo extremo de clases, pero sí un ejemplo de estructura habitual en este tipo de problemas.

Con estas condiciones de partida lo que se pretende en este trabajo es fundamentalmente lidiar con el problema del desbalanceo de datos experimentando con distintos métodos o técnicas de *resampling.* No es el objetivo principal de este trabajo, por tanto, alcanzar el mayor nivel de ajuste o acierto en las predicciones, sino evaluar la incidencia que distintas técnicas de *resampling* tienen sobre los resultados de la clasificación. Es cierto que se pretende alcanzar buenos ajustes o niveles de predicción, pero sin centrar esfuerzos en la mejora del nivel predictivo a través del ajuste de hiperparámetros de los clasificadores. El grueso del trabajo concentra esfuerzos en:

- Entender suficientemente el comportamiento de las técnicas de *resampling* más importantes.
- Combinar el uso de distintos clasificadores con diferentes técnicas de *resampling*.
- Emplear las métricas de rendimiento adecuadas para distinguir y evaluar el impacto que el uso de una u otra combinación de técnicas, tiene sobre las tareas de clasificación aplicadas a la fuga de clientes.

Concretamente se contrastará el comportamiento de hasta ocho técnicas de *resampling* (4 técnicas *undersampling*, 3 de *oversampling* y una técnica híbrida) diferentes para tres clasificadores distintos (regresión logística, bosques aleatorios y *gradient boosting*).

La estructura y organización del trabajo consta de tres capítulos diferenciados. El primero de estos capítulos es una breve introducción a los distintos métodos y soluciones que comúnmente han servido para manejar conjuntos de datos desbalanceados en tareas de clasificación similares a las desarrolladas en este trabajo. El segundo, se centra en la descripción y explicación de las diferentes técnicas usadas. En primer lugar, se procederá a la explicación teórico-técnica de los métodos de resampling que serán de aplicación en la parte empírica. Aquí se pretende además encajar la evolución de las técnicas ofreciendo continuidades

entre unas y otras para entender el contexto de evolución de esta subdisciplina. Un segundo epígrafe, más breve, aborda la explicación de las distintas técnicas de clasificación sobre las que se medirá la influencia de la aplicación de técnicas de resampling.

En el siguiente de los capítulos se describe todo el desarrollo empírico del trabajo desde la propia selección y presentación de los datos. Se ofrecerá aquí una detallada descripción de todas las etapas de procesado y preparación de los datos para conseguir un conjunto de datos consistente, unido al desarrollo de los flujos de trabajo que han guiado el proyecto. Dentro de este capítulo se presentarán los resultados de las distintas aplicaciones empíricas de todas las técnicas, centrándose en la dimensión comparativa a través de las distintas métricas de rendimiento extraídas de los modelos.

Como resulta lógico, el trabajo se remata con un breve sumario de conclusiones en el que se destacan los hallazgos y resultados más importantes. Además, quedan planteados los elementos sobre los que podrían desarrollarse nuevas vías de estudio que enriquecerían la descripción de un problema complejo.

2. Marco en la predicción de la fuga de clientes y el desbalanceo de datos.

En este breve capítulo se pretende esbozar el contexto de desarrollo teórico en el que se ubican los principales temas tratados en este trabajo. Estos son, el desbalanceo de datos, el desempeño de los algoritmos de clasificación en este contexto y más concretamente involucrados en la predicción de la fuga de clientes (o *churn* por su terminología inglesa). El siguiente capítulo se dedica por completo a la explicación de las técnicas apropiadas para el trabajo con datos desbalanceados, pero conviene encuadrar el caso de uso concreto en el que se desenvuelven. Existe una amplia y vasta literatura dedicada al estudio del *churn* en distintos ámbitos, aplicando las metodologías que mejor se ajustan en cada uno de ellos. Durante las siguientes líneas se pretende ofrecer una visión de conjunto sobre el estudio de caso abordado en este trabajo y plantear una transición lógica a los conceptos más teóricos en torno a los que girará todo el trabajo.

La variedad metodológica en la predicción de la fuga de clientes.

La predicción de la fuga de clientes es un campo muy rico desde un punto de vista metodológico. Las aproximaciones al problema de la fuga de clientes son variadas y las soluciones técnicas pueden seguir distintos caminos, en muchos casos dependiendo del ámbito comercial o de negocio en el que se realiza el estudio. En general, la premisa fundamental que guía estos estudios es que la captación de

nuevos clientes es mucho más costosa que la retención de los clientes antiguos. Concretamente se ha llegado a cifrar entre cinco y seis veces más costosa la adquisición de nuevos clientes que la retención de los existentes (Verbeke *et al.*, 2012). Obviamente, esta afirmación, genera un contexto en el que la predicción de los clientes fugados resulta central desde una perspectiva de negocio. El interés derivado para las compañías refuerza la investigación en este campo y provoca la profusión de soluciones y propuestas a un mismo problema.

Concretamente el problema de la fuga de clientes es intensamente investigado en áreas de negocio y comerciales en las que la rotación de cliente suele ser lógica. Desde luego en este contexto las compañías de telefonía son agentes claramente interesados y probablemente los datos que más se han trabajado en la literatura. Otros ámbitos de aplicación son la contratación de servicios financieros como tarjetas de crédito o las compañías de seguros. El caso concreto de este trabajo, servicios de música y vídeo en *streaming*, no tienen aún una presencia notable, por lo que es posible considerar novedoso el acercamiento a esta parcela comercial.

Como se decía, por la amplia disponibilidad de datos y como área paradigmática del estudio del *churn*, la telefonía ha centrado la mayor parte de los trabajos. Algunos de ellos originales por poder darse únicamente en este ámbito como por ejemplo el empleo de grafos y la modelización de redes sociales para predecir la fuga. En los últimos años se han desarrollado trabajos que han tenido en cuenta la dimensión relacional entre sus clientes como determinante de la fuga, como una variable de propagación similar a la del contagio en el mundo sanitario. Estudios destacables en este sentido con los de (Backiel *et al.*, 2015); (Mitrović *et al.*, 2017);(Mitrović *et al.*, 2019) o (Salve, Mori and Ricci, 2017). La creación de grafos puede gestionarse como modelo para la predicción o como generador de variables con alto valor para la predicción en su combinación con técnicas como vecinos cercanos y k-vecinos cercanos. Ha habido incursiones también en el estudio de la fuga para tarjetas de crédito desde esta aproximación como la de (Lin, Tzeng and Chin, 2011). Lo cierto es que este tipo de trabajos no son los más habituales y pueden considerarse extraordinarios, siendo los más habituales los estudios que emplean uno o varios algoritmos de clasificación para el caso binario.

En este sentido, son muchas las técnicas empleadas y que tienen vigencia. Un espacio generalmente muy explorado en el caso de la fuga de clientes es el trabajo con redes neuronales como los de (Kumar and Kumar, 2019) y (Sivasankar and Vijaya, 2019) o con máquinas de vector soporte como los trabajos de (Farquad, Ravi and Raju, 2014) y (Gordini and Veglio, 2017). Son técnicas defendidas por sus autores por su potencia predictiva, aunque tienen el problema de dificultar la capacidad interpretativa o explicativa de los modelos, una cuestión que a nivel empresarial

puede tener una importancia central; conocer los comportamientos de los clientes y las causas principales que están detrás de la fuga.

Por la razón anterior es muy habitual el trabajo con árboles, tanto con árboles de decisión sencillos como técnicas como el *bagging* y *random forest* o el *boosting*. Un trabajo especialmente interesante con árboles de decisión y que además añade matrices de coste asociadas a la predicción, muy valioso para las compañías, es el de (Höppner *et al.*, 2020). Haciendo uso del *boosting* se destacan los trabajos de (Lu *et al.*, 2014); y (Jain, Khunteta and Srivastava, 2020) en este último caso trabajando combinadamente con regresiones logísticas. En este sentido, es destacable el desarrollo de algoritmos híbridos de predicción como (De Caigny, Coussement and De Bock, 2018) haciendo uso de la regresión y los árboles de decisión. Empleando *bagging* o *random forest* el trabajo de (Ullah *et al.*, 2019) sobresale por los buenos resultados de este tipo de técnicas en comparación con otras.

Este último caso es el más habitual dentro de la literatura académica, por lo general lo que se observa en el caso del estudio del *churn* es la comparativa de técnicas. En este sentido, los trabajos más relevantes se destacan por incluir comparativas de una gran cantidad de técnicas que permitan visualizar qué tipo de algoritmo es más adecuado para cada caso en concreto. Un trabajo notable dentro de este grupo sería el elaborado por (Maldonado, López and Vairetti, 2020) incluyendo hasta 14 técnicas diferentes para 5 conjuntos de datos distintos. Este trabajo recorre técnicas como redes neuronales, árboles de decisión, máquinas vector soporte, k-vecinos cercanos o regresión logística.

Por último, conviene subrayar los acercamientos al estudio de la fuga de clientes desde el análisis de supervivencia, un campo que ha cobrado cierta relevancia en combinación con algoritmos de clasificación como las redes neuronales y los bosques aleatorios. Dos de los trabajos más importantes son los de (Dirick, Claeskens and Baesens, 2017) y el de (Wong, 2011). Lo habitual es el trabajo con regresiones de Cox basados en el trabajo con la función de riesgo, de forma que introduciendo covariables sea posible definir tiempos de supervivencia diferentes en base a las distintas características de los participantes. La regresión de Cox ha destacado gracias a la sustitución de la regresión como método de cálculo para los coeficientes por redes neuronales por ejemplo, como es el caso de los trabajos de (Kvamme, Borgan and Scheel, 2019) que además tienen en cuenta el desbalanceo de datos considerando cómo manejar datos de supervivencia con una alta cantidad de censuras por la derecha, es decir, con una tasa de supervivencia muy alta (la mayoría de los clientes no se fuga). Otros trabajos de interés en este campo, hacen uso de algoritmos novedosos, que combinan modelos de supervivencia y modelos de aprendizaje automático como por ejemplo el de (Afrin *et al.*, 2018) en su trabajo con *random survival forests*.

El balanceo de datos orientado a la predicción de clase en la fuga de clientes.

Como avanzábamos antes, los trabajos más relevantes en la predicción de la fuga de clientes, atienden a la dimensión comparativa multitécnica. Concretamente los que más interesan a efectos de este trabajo son aquellos que además tienen en cuenta el problema del desbalanceo de datos. El trabajo más relevante en este ámbito y que ya apuntaba la centralidad de la aplicación de técnicas de balanceo de datos como necesarias es el de (Verbeke *et al.*, 2012). Este estudio es probablemente el más completo en cuanto a comparación de técnicas aplicadas al estudio de la fuga de clientes y se ubica como un nexo con otro grupo de estudios que definitivamente tienen en cuenta el desbalanceo de clases.

Este último grupo de trabajos son los que más encajan a nivel teórico con lo pretendido en este trabajo. La metodología en ellos es la comparación del rendimiento de distintos modelos de clasificación previa modificación de la muestra vía *resampling*. Los trabajos más destacados son (Zhu *et al.*, 2018); (Zhu, Baesens and vanden Broucke, 2017). En el primero se procura una comparativa de la influencia en el rendimiento del clasificador con hasta 9 técnicas distintas de balanceo de datos. En el segundo estudio se plantean las mismas técnicas de *resampling* y se proponen distintos clasificadores (redes neuronales, maquinas vector soporte, *bagging*, *boosting*, etc..) obteniendo una comparativa de hasta 34 modelos distintos aplicados a 11 conjuntos de datos. Los trabajos de Zhu, Baesens y Vanden Broucke siguen la metodología de trabajo iniciada por (Batista, Prati and Monard, 2004) o (Burez and Van den Poel, 2009) relevantes trabajos en el estudio de la fuga de clientes atendiendo a distintas técnicas de balanceo para la mejora en las tareas de clasificación.

El siguiente capítulo se centra en explicar y definir las técnicas de *resampling* que serán de aplicación en este trabajo y los algoritmos de clasificación que serán empleados para evaluar la incidencia del *resampling* en la clasificación. El desarrollo a partir de este punto sigue una estructura similar a la presentada en este último grupo de estudios destacado.

3. Métodos y técnicas de trabajo.

En este epígrafe se presentan las técnicas y modelos usados en este trabajo. Como se avanzaba en la introducción, el principal objetivo de este trabajo es evaluar la incidencia de las técnicas de balanceo de datos sobre algoritmos para predicción de clase (en nuestro caso binaria). Por esta razón, las explicaciones incluidas en este punto, se dividen en tres apartados. En un primer y breve apartado se aproximan las soluciones comunes en trabajos con datos desbalanceados. En el segundo apartado se aborda la explicación de las distintas técnicas de balanceo de datos que

se usarán en este trabajo. Por último, se da cuenta de los algoritmos de predicción o clasificación que serán usados para medir las diferencias de rendimiento en función del uso de una u otra técnica de balanceo de datos, añadiendo al final la explicación de las métricas de rendimiento apropiadas para la valoración del desempeño.

3.1. Enfoques comunes frente a problemas del desbalanceo de datos.

La finalidad de las soluciones o enfoques presentados es la de corregir los efectos provocados por el desbalanceo de clases sobre la clasificación de los modelos. En los problemas de clasificación binaria[1], recordemos, una de las clases es dominante (clase mayoritaria) y la otra clase, clase minoritaria, representa un peso mucho menor sobre el total de la muestra. Enfrentar este problema es posible desde dos aproximaciones. Una primera aproximación integra soluciones que se centran en el algoritmo, ya sea modificando su arquitectura como a través del ajuste de hiperparámetros para inducir mejores ajustes en datos desbalanceados. Una segunda aproximación consiste en modificar la estructura de datos original modificando el número de instancias que se incluirán en el estudio. Este segundo grupo de soluciones son quizás las más populares en la literatura y las que centran el interés de este trabajo y que conocemos como **técnicas de balanceo de datos** o técnicas de *resampling*.

Hasta este momento se han descrito los principales problemas relacionados con el desbalanceo de datos, que pueden provocar una merma de la capacidad predictiva de los modelos en la categoría o grupo minoritario (generalmente grupo que presenta la característica de interés). Cuando este tipo de problemas sobrevienen las soluciones o aproximaciones apuntadas en el párrafo anterior pueden permitir mejoras de rendimiento en los modelos de clasificación. En continuidad con lo anterior, en las soluciones que van desde el algoritmo hasta la modificación de la estructura original de los datos podemos destacar los siguientes enfoques.

- *Aprendizaje sensible al coste.*

Esta aproximación altera la asunción de que los errores de clasificación tienen el mismo coste independientemente de la clase que se predice. El aprendizaje sensible al coste se apoya en una matriz de costes (análoga a la matriz de confusión) que describe los costes de clasificación errónea de una instancia. Los elementos clasificados correctamente tendrían un coste 0. En la **Tabla 1** se muestra la distribución de los costes para un problema de clasificación binaria. Los costes en la

[1] El desbalanceo de clases no solo afecta a problemas de clasificación binaria. Son muchos los ejemplos en los que el uso de técnicas de balanceo se aplica a problemas multiclase (véase por ejemplo, Fernández-Navarro, Hervás-Martínez and Antonio Gutiérrez, 2011). Si bien es cierto que la literatura es más densa en el caso binario.

diagonal; C (-,-) y C (+,+) representarían los costes cero correspondientes a las clasificaciones correctas.

Tabla 1: Ejemplo de una matriz de costes y matriz de confusión.

Matriz de coste			
		PREDICCIÓN	
		0	**1**
OBSERVACIÓN	**0**	C (-,-)	C (+,-)
	1	C (-,+)	C (+,+)

Matriz de confusión			
		PREDICCIÓN	
		0	**1**
OBSERVACIÓN	**0**	VN	FN
	1	FP	VP

Si como en la **Tabla 1** partimos de las matrices de coste y confusión, el objetivo principal del aprendizaje sensible al coste sería minimizar la siguiente función en el proceso de clasificación:

$$Coste\ total = c(+,-) \cdot FN + c(-,+) \cdot FP$$

Los costes asociados no tienen por qué ser monetarios, pueden ser de tiempo o cualquier otra unidad de medida. Lo cierto es que este método no se centra en el problema del desbalanceo de clases sino en los costes asociados a los errores de clasificación. Es por ello, que resulta en un procedimiento comúnmente combinado con estrategias de *resampling* o como herramienta de valoración en el ajuste de hiperparámetros de algoritmos de clasificación (véase por ejemplo Thai-Nghe, Gantner and Schmidt-Thieme, 2010).

- *Refuerzo de aprendizaje en la clase minoritaria.*

Otras soluciones comunes en los problemas de clasificación con datos desbalanceados es la implementación de algoritmos, que internamente, tengan en cuenta el desbalanceo entre clases. De esta forma, se estaría reforzando el aprendizaje en la clase minoritaria. Usualmente se han referenciado como soluciones adaptativas del algoritmo sin abordar ninguna modificación de los datos originales. La idea general de estos métodos es la inclusión de una penalización o sesgo para balancear las clases que se traduce en ponderarlas para otorgar más importancia a la clasificación correcta en la clase minoritaria.

Este tipo de adaptaciones de los algoritmos son realmente dispares, dependen de cada modelización y técnica utilizada. No obstante, sería posible una amplia categorización o clasificación de las aproximaciones incluidas en este punto, estableciendo además continuidad con lo expuesto anteriormente (véanse Nguyen, Bouzerdoum and Phung, 2009 & Sun, Wong and Kamel, 2009) como ejemplos. En la siguiente tabla se resume este otro de soluciones para problemas de clasificación con datos desbalanceados.

Tabla 2: Tabla resumen de soluciones, a nivel de los algoritmos, frente al problema del desbalanceo de datos.

Aprendizaje en la clase minoritaria	Se centra en entrenar los modelos incluyendo únicamente instancias de la clase minoritaria
Introducción de sesgos de aprendizaje	Introduce pesos en el aprendizaje o sesgos de aprendizaje hacia la clase minoritaria: ▪ Regresión con pesos por clase (*weighted logistic regression*) ▪ *Boosting* adaptativo (*AdaBoost*) ▪ Árboles de decisión ▪ Random Forest (*weighted Random Forest*) ▪ Máquinas de vector soporte
Integración de costes en aprendizaje	Integra los costes en la fase de entrenamiento de los modelos. ▪ *Boosting* adaptativo sensible al coste ▪ Redes Neuronales sensibles al coste ▪ Árboles de decisión sensibles al coste

El *aprendizaje en la clase minoritaria* resulta como solución derivada de ciertos problemas en los que es realmente difícil recoger o incluir instancias de una de las clases. En este sentido, el problema es abordado entrenando un clasificador usando sólo instancias de una clase (algo posible en problemas de clasificación binaria). De esta forma, cualquier instancia que en la fase de test no se ajusta a la clase aprendida será considerado como un atípico. Intuitivamente existen tres caminos posibles con esta aproximación:

- Entrenar un clasificador para la clase mayoritaria
- Entrenar un clasificador para la clase minoritaria
- Entrenar clasificadores para ambas clases por separado y combinar los resultados.

En lo que respecta a la introducción de *sesgos en el aprendizaje* la idea que subyace en todos los casos es conceder una mayor importancia a los errores de predicción que se cometen con la clase minoritaria. En este sentido, el *boosting* (concretamente el *boosting* adaptativo) (Sun, Wong and Kamel, 2009) se ha presentado como una técnica importante como solución en problemas de clasificación con desbalanceo de datos. Esta técnica inicialmente concede la misma importancia o peso w_i a todas las instancias y secuencialmente incrementa los pesos para los casos que se clasificaron incorrectamente. De forma similar, son muchos algoritmos que permiten la ponderación de casos por clase como se ha visto en implementaciones de la regresión logística, árboles de decisión o máquinas de vector soporte.

Un paso más y que presenta la integración de dos de las soluciones ya mencionadas tiene que ver con la implementación de algoritmos sensibles al coste. Es decir, los costes quedan integrados en los modelos de entrenamiento. Para ahondar un poco más en lo ya expuesto se dirá que los costes en los errores en este tipo de soluciones tomarán valores entre 0 y 1. Análogamente la función de pérdida aplicada a las predicciones de un algoritmo (y que también toma valores entre 0 y 1) puede ser fácilmente minimizada centrándose fundamentalmente en la clase mayoritaria o incluso obviando la clase minoritaria (Fernández *et al.*, 2018).

Los algoritmos sensibles al coste corrigen este problema adaptando la función de pérdida al problema concreto, de tal modo que estos costes se pueden ver como factores de penalización en el proceso de entrenamiento del modelo. Estas "penalizaciones" más altas en las clases de interés (generalmente la minoritaria) centran el esfuerzo de clasificación en esas clases más difíciles de predecir y de mayor coste en sus errores.

- *Técnicas de resampling*

Este último grupo de soluciones o técnicas son las que centran el interés de este trabajo. Son el tipo de soluciones que inciden directamente sobre los datos modificando las distribuciones de alguna de las clases o incluso de las dos.

En este caso las soluciones generales a adoptar se dividen en dos grandes grupos. El primer grupo de soluciones consiste en eliminar instancias de la clase mayoritaria (***undersampling***). El segundo grupo de técnicas de *resampling* propone replicar o generar nuevas instancias de la clase minoritaria (***oversampling***). Por otro lado, podemos dividir las técnicas de balanceo de datos en heurísticas y no heurísticas. En este sentido, los procesos aleatorios o no controlados (no heurísticas) incurren en ciertos riesgos que los métodos heurísticos tratan de resolver estableciendo "controles" en la pérdida o replicado de información.

El próximo epígrafe tratará de explicar en profundidad las técnicas de *resampling* más populares en la literatura en relación al problema abordado en la parte empírica del trabajo.

3.2. Técnicas de "resampling".

Como se ha avanzado las técnicas de balanceo de datos o *resampling* pueden dividirse en dos grupos, aunque ha de matizarse conceptualmente ya que el *undersampling* y el *oversampling* pueden combinarse en lo que se denomina ***ensemble resampling*** y el funcionamiento de los distintos algoritmos de *resampling* difiere a pesar de abordar el mismo problema. La principal cuestión es qué instancias han de eliminarse o que instancias han de conservarse en el caso del

undersampling; y cómo o dónde van a replicarse nuevas instancias en el caso del *oversampling*.

Como se avanzaba, existe la posibilidad de aplicar modelos de *resampling* no heurísticos que se circunscriben al submuestreo y el sobremuestro aleatorios y heurísticos. Los modelos heurísticos refieren técnicas que otorgan una estructura matemáticamente razonada en el proceso de *resampling*. Otra clasificación posible tiene en cuenta si el algoritmo de *resampling* tiene como objetivo último balancear las clases hasta igualar el número de casos en cada una o balancear las clases siguiendo determinados criterios (limpieza de casos recurrentes, por ejemplo) sin necesidad de igualar el número de casos. En la

Tabla 3: Características de los algoritmos de *resampling* estudiados.

	Algoritmo	Tipo de método	Tipo de modificación
Underampling	*Undersampling aleatorio*	No heurístico	Igualar nº de casos en clases
	Condensed Nearest Neighbours	Heurístico	Eliminar las instancias recurrentes
	Tomek Links	Heurístico	Eliminar las instancias recurrentes
	Edited Nearest Neighbours	Heurístico	Eliminar las instancias recurrentes
	NearMiss	Heurístico	Igualar nº de casos en clases
Oversampling	*Oversampling aleatorio*	No heurístico	Igualar nº de casos en clases
	Synthetic Minority Oversampling Technique	Heurístico	Igualar nº de casos en clases
	Adaptative Synthetic Sampling	Heurístico	Igualar nº de casos en clases
Híbridos	*SMOTE+Tomek Links*	Heurístico	Igualar nº de casos en clases
	SMOTE+ENN	Heurístico	Igualar nº de casos en clases

Las próximas páginas se centrarán en mostrar y explicar las técnicas incluídas en la **Tabla 3**. Estos algoritmos pueden ser considerados los más importantes dentro de la literatura y los que además serán de aplicación en la parte empírica de este trabajo.

3.2.1. Técnicas de *undersampling*

Dentro del *undersamplig* se engloban todas las técnicas que tienen como finalidad igualar las distribuciones desbalanceadas de datos eliminando instancias de la clase mayoritaria. Estamos, por tanto, ante un grupo de técnicas que operan sobre los casos de la clase mayoritaria aislando o respetando la distribución de casos de la clase minoritaria.

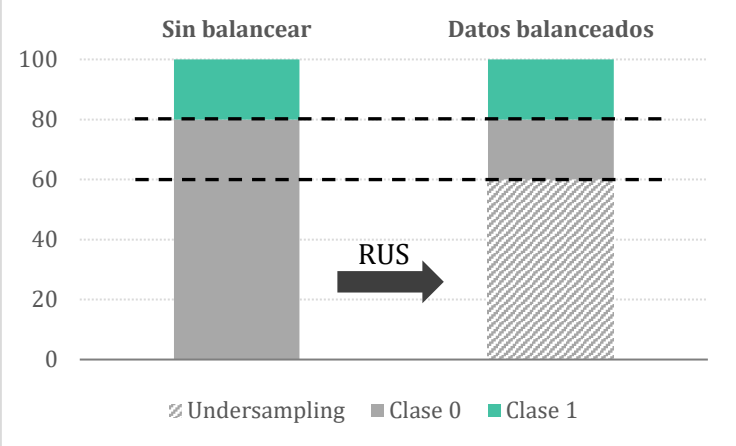
Las soluciones son distintas en función del algoritmo que decida qué instancias eliminar o qué instancias conservar. Las técnicas de este grupo van desde el *undersampling* aleatorio o *Random Undersampling*, método no heurístico, hasta soluciones que fundamentalmente se apoyan en el método de los *k-vecinos* para discernir qué instancias eliminar o conservar.

En este grupo de técnicas el riesgo asumido es la pérdida de información original vinculada a los datos de la clase mayoritaria. Es por ello que los métodos de selección heurísticos concentran su esfuerzo en seleccionar aquellas instancias que sería más pertinente eliminar de cara a conservar la variabilidad más importante vinculada a los datos.

3.2.1.1. Random Undersampling.

El ***Random Undersampling*** (RUS) es el método de *resampling* más sencillo. En este sentido, la única operación que aborda es la eliminación aleatoria de instancias de la clase mayoritaria. Lo más habitual es igualar el número de instancias de ambas clases. Es decir, reducir el número de casos de la clase mayoritaria al número de casos de la clase minoritaria tal como sencillamente se muestra en la **Figura 1**.

Figura 1: Ejemplo de la aplicación de *Random Undersampling*



El principal inconveniente del *undersampling* o submuestreo aleatorio es que este método puede descartar datos potencialmente útiles que podrían ser importantes para el proceso de clasificación posterior (Fernández *et al.*, 2018; Batista, Prati and Monard, 2004).

A pesar de que los métodos aleatorios (tanto *random undersampling*, como *random oversampling*) han demostrado buenos rendimientos en problemas de clasificación y con tiempos de ejecución poco costosos, el empleo de los métodos heurísticos vienen a corregir los posibles sesgos en la eliminación de información (Batista, Prati and Monard, 2004). El resto de aproximaciones heurísticas puede decirse que

pretenden una limpieza de casos para balancear los datos. Lo que se pretende en general es eliminar información redundante, eliminar aquellos casos con escaso aporte a la variabilidad entre clases.

3.2.1.2. Condensed Nearest Neighbours.

Como se ha apuntado la mayoría de las soluciones heurísticas para la eliminación de instancias parten o se basan en el método de los ***k*-vecinos más próximos**. Éste es un método basado en instancias o casos, las clasificaciones se realizan en base a la similitud a los casos más cercanos. Es un método, por tanto, no paramétrico. Lo que este algoritmo de clasificación hace es fijarse en los k casos o instancias más cercanas al caso de interés (a clasificar) para realizar la clasificación. Por tanto, al establecer un pronóstico no es necesario tener en cuenta todas las observaciones sino aquellas que espacialmente pueden considerarse más parecidas (es una técnica que trabaja a nivel local). El parámetro determinante para la clasificación será qué cantidad de vecinos cercanos van a tenerse en cuenta para realizar la clasificación.

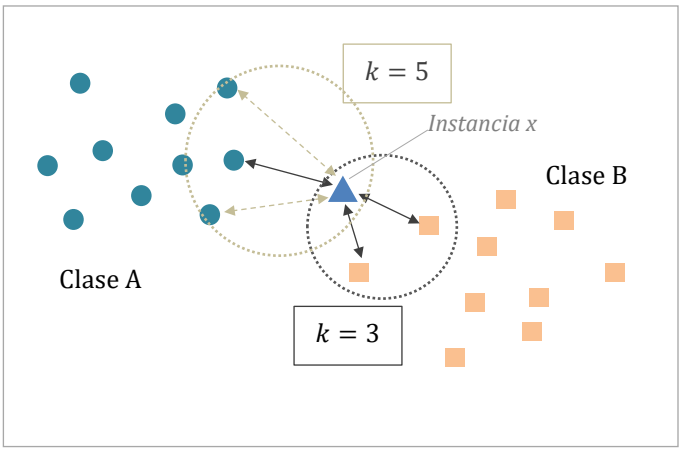
La base para la detección de los vecinos, como en otros algoritmos de clasificación o *clustering*, son las distancias entre el punto a clasificar y sus vecinos. Existen variadas formas de calcular la distancia entre puntos, aunque la más usual es la distancia euclídea. De tal modo, si tenemos dos puntos p y q con valores para una colección de n variables la distancia euclidiana será:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

El resultado de la clasificación será la asignación de la clase más común para un punto p a clasificar.

Como se puede observar en la **Figura 2**, la definición del número de k vecinos puede alterar sustancialmente los resultados. Para el ejemplo en el proceso de clasificación de la instancia x, si usamos un valor $k = 3$ la clase para la instancia a predecir será del tipo B. Sin embargo, si el valor de $k = 5$, se observa que la predicción de clase pertenecería ahora al grupo A.

Figura 2: Ilustración gráfica del funcionamiento del algoritmo k vecinos más próximos



Condensed Nearest Neighbours (CNN) fue la primera de las propuestas (Hart, 1968) basadas en la vecindad de instancias con el objetivo de eliminar (condensar) datos que no aportan información útil en un conjunto. CNN conserva el enfoque básico de la regla de los k-vecinos, persiguiendo en tres fases obtener un conjunto de datos consistente.

Partimos de un conjunto de datos $X = \{x_1, x_2, \dots, x_N\}$. Asumimos además que cada instancia u observación x_i tiene asociada una clase o etiqueta $y_i, \forall i = 1, \dots, N$. Tenemos por tanto un conjunto en el que todas las instancias presentarán la siguiente estructuta (x_1, y_1), $(x_2, y_2), \dots, (x_n, y_n)$. El objetivo de este algoritmo es crear un nuevo conjunto de datos condensado U. Este nuevo conjunto estará compuesto por lo que se denomina prototipos. El proceso iterativo para llegar a construir ese nuevo conjunto es el siguiente:

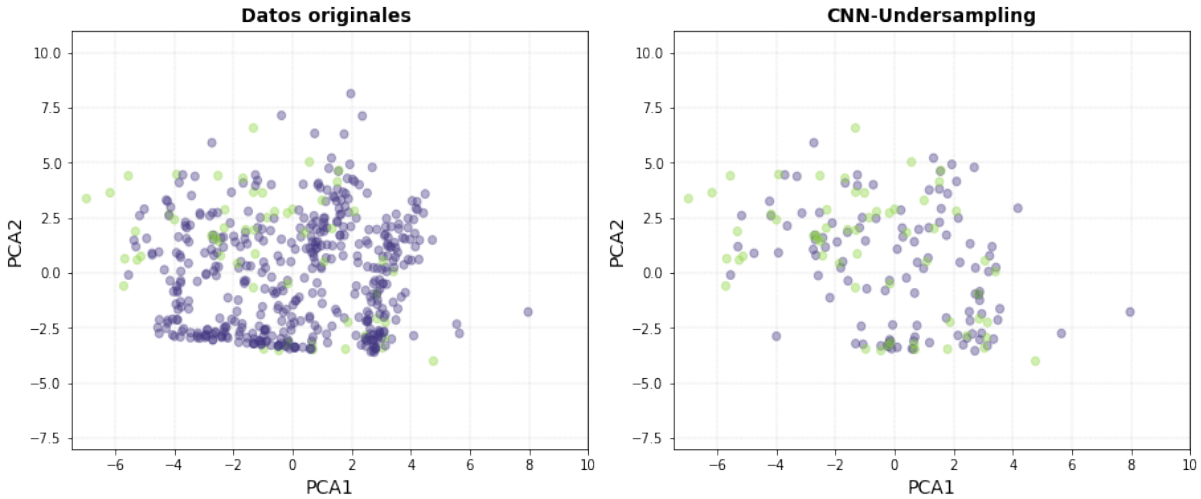
- Se muestrea un primer elemento para añadir a $U = \{x_0\}$, de forma aleatoria.
- Por la regla de los k-vecinos más cercanos con $k = 1$ (un vecino), se elige una nueva instancia que se añadirá a U cuando la etiqueta de ese vecino y la etiqueta de la instancia incluida en U no coincidan. Es decir, un prototipo es el vecino más cercano de la instancia incluida en U cuya etiqueta de clase no coincide con la clase de su vecino más cercano de X.
- Los vecinos cercanos en $k = 1$ cuyas etiquetas coinciden son absorbidos o eliminados. Estas instancias serán las consideradas recurrentes y no aparecerán representadas en el nuevo conjunto de datos U.

Este proceso se realiza para cada uno de los elementos de X . Se puede decir que estos puntos seleccionados (prototipos) son los puntos que dibujan la frontera de decisión entre las clases del conjunto de datos. En el caso de querer tan solo minorar o reducir las instancias de una clase (la mayoritaria normalmente), la solución se

adopta de la misma manera con la salvedad de que todas las instancias de la clase que no se va a reducir forman parte desde el principio del conjunto *U*.

La principal crítica que se le ha hecho a este algoritmo de limpieza de datos es la selección aleatoria de ejemplos, en las primeras iteraciones del proceso. Como vemos más arriba el arranque de la selección de la primera instancia que formará parte de *U* es aleatoria. Otros algoritmos como veremos más adelante pretenden controlar esa primera etapa.

Figura 3: *Undersampling* de la clase mayoritaria vía *Condensed Nearest Neighbor*.



Fuente: Elaboración propia con una submuestra de datos tomados del *dataset* usado en la parte empírica de este trabajo (KKBOX). Se toman 500 casos al azar para la evaluación gráfica del comportamiento de los algoritmos de *resampling*

En la **Figura 3** podemos ver una representación gráfica del tipo de limpieza o condensado de datos por el algoritmo CNN. El ejemplo mostrado se ejecuta para una muestra de 500 casos y tras la aplicación del algoritmo se aprecia una reducción bastante importante de la clase mayoritaria. Concretamente, para este ejemplo, el número de instancias de la clase mayoritaria es de 442 y después de la aplicación de la técnica se reduce a 110. La limpieza en este caso concreto es especialmente intensa. En otros métodos de limpieza que se apoyan en el método *k*-vecinos y que veremos más adelante, la eliminación de instancias recurrentes no es tan grande.

3.2.1.3. Tomek Links.

Este segundo método basado también en los vecinos más próximos surge como revisión del trabajo de Hart (1968) por parte de Ivan Tomek (1976b). Se parte del trabajo en CNN subrayando ciertas desventajas en el proceso.

La primera de las críticas apunta a la aleatoriedad de elección de las submuestras o instancias en la primera parte del proceso. Tomek apunta que la construcción del

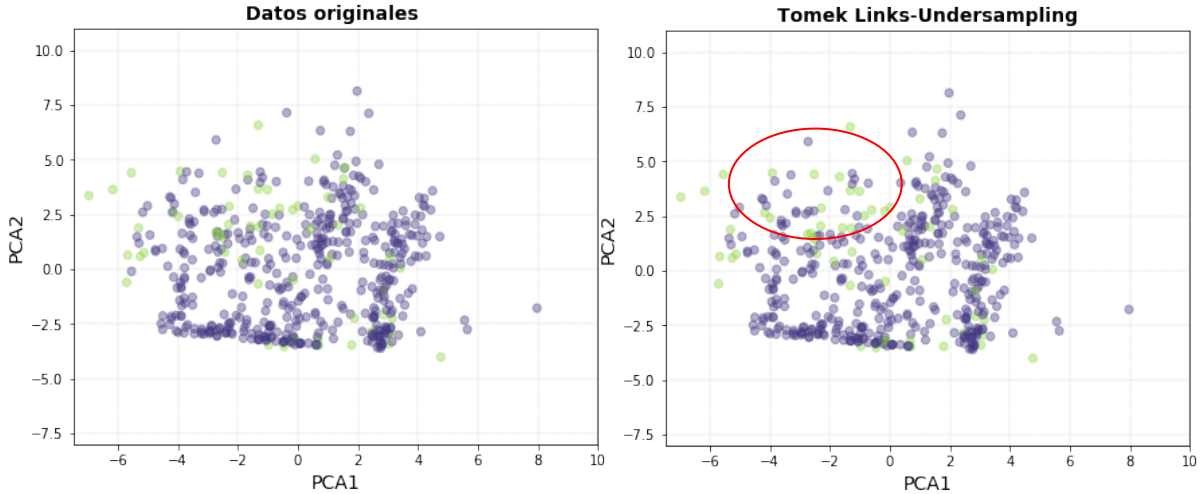
conjunto U a partir del conjunto original X parte de una elección aleatoria fundamentalmente intensa en las primeras iteraciones. Una segunda crítica es que la búsqueda de la frontera en el requisito de no coincidencia de etiquetas de los prototipos, otorga una importancia central a estos puntos en U. Una importancia que en X no tienen necesariamente para un posterior proceso de clasificación. A este punto concreto se refiere que el verdadero límite de decisión entre clases es por definición desconocido y resulta muy difícil aproximar una frontera de decisión lineal, aquello a lo que pretende acercarse Hart. Tomek propone buscar los puntos que permiten ese tipo de decisiones, pero por partes o regiones en el espacio.

La propuesta o **método *Tomek Links*** puede definirse como sigue a continuación. Partimos de dos instancias dadas del conjunto de datos original que llamaremos x_i, x_j y de distinta clase. Si se conocen las distancias d entre los puntos y no existe una tercera instancia x_l que cumpla:

$$d(x_i, x_l) < d(x_i, x_j) \vee d(x_j, x_l) < d(x_i, x_j)$$

La instancia de la clase mayoritaria será eliminada. La distancia entre el par de instancias $x_i, y\, x_j$ es lo que se llama **enlace de *Tomek*** o *Tomek Link* por su terminología en inglés. Este enlace está relacionado con la técnica k-vecinos en la selección de instancias a considerar. Los enlaces de *Tomek* han de cumplir las siguientes condiciones: a) el vecino más próximo de x_i ha de ser x_j; b) el vecino más próximo de x_j ha de ser x_i y c) $x_i\, y\, x_j$ han de pertenecer a distinta clase. Digamos, por tanto, que en contraposición a CNN lo que se pretende con este método es limpiar las regiones en las que las instancias de distinta clase están más cerca. Se centra más en qué eliminar en lugar de qué conservar. Básicamente lo que se nos permite es realizar una limpieza de las regiones que marcan la frontera entre clases.

Figura 4: *Undersampling* de la clase mayoritaria vía *Tomek Links*.



Fuente: Elaboración propia con una submuestra de datos tomados del *dataset* usado en la parte empírica de este trabajo (KKBOX).

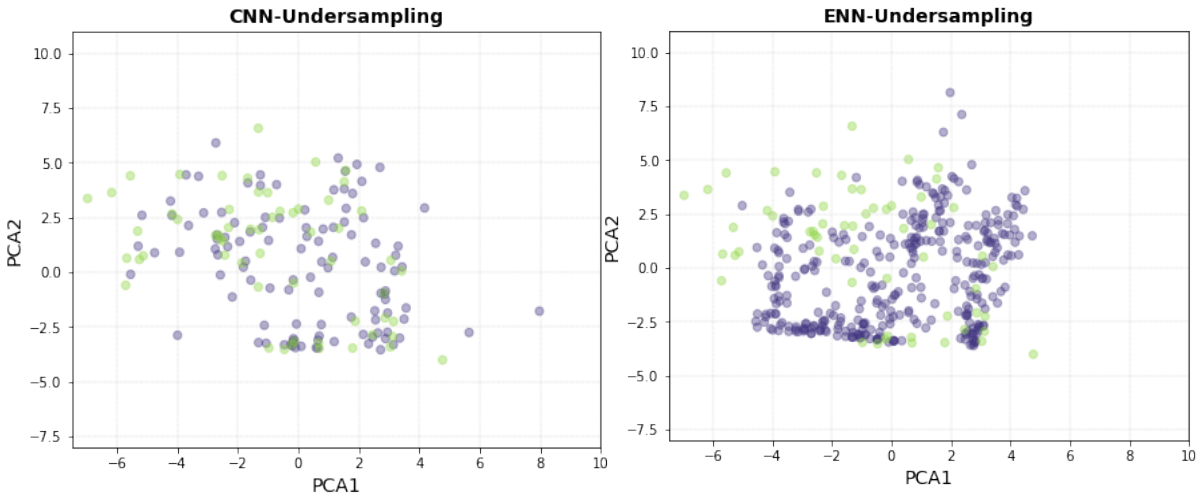
En la **Figura 4** observamos el resultado de la aplicación del método *Tomek Links* al ejemplo que se presentaba anteriormente. El número de instancias eliminadas es mucho menor que con CNN, se reduce la clase mayoritaria de 442 instancias a 415 (solo 27 instancias eliminadas). Como decíamos, la propia estructura o complejidad de los datos puede definir distintas aplicaciones del mismo método de *resampling*. Lo que queda aquí de manifiesto es que los métodos CNN y *Tomek Links* difieren claramente en las soluciones aportadas a un mismo problema. Con CNN se conservan los puntos de distinta clase más cercanos, eliminando las instancias en regiones que domina la clase mayoritaria. Con *Tomek links* apenas se aprecia (véase resaltado en rojo en la **Figura 4**) una leve "limpieza" en las zonas de contacto entre clases. Es probable que para problemas con desbalanceo acusado de clases soluciones como *Tomek Links* no sean de gran ayuda en problemas de clasificación (como veremos más adelante). El valor de este tipo de técnicas no se circunscribe al *undersamplig* sino que ofrecen también la posibilidad de quedar integradas en modelos de balanceo híbridos.

3.2.1.4. Edited Nearest Neighbours.

Si el método CNN se centraba fundamentalmente en conservar los casos que estuvieran cerca de la frontera de decisión y *Tomek Links* cambiaba esa noción para "limpiar" las regiones de contacto, ***Edited Nearest Neighbours*** va un paso más allá en ese cambio de paradigma y se acerca más a la solución propuesta por Tomek.

El método *Edited Nearest Neighbours* (ENN) opera con la regla de vecinos cercanos con $k = 3$ pero no de la misma forma que se hacía en CNN. ENN es un método de limpieza directo, similar a *Tomek Links* y no se centra en la frontera de decisión. Wilson (1972) al igual que Tomek propone una alternativa a CNN con la adecuación de la regla de los k-vecinos elevando k a 3 y vinculando la limpieza de casos en base, únicamente, a esos 3 vecinos más próximos. Es decir; a) se toma una instancia x_i del conjunto de datos perteneciente a la clase mayoritaria, b) se computan sus 3 vecinos más próximos, c) si la clase de los vecinos más próximos (al menos 2) difiere de la clase del elemento x_i esa instancia será eliminada. Como vemos la solución aportada es más sencilla e intuitiva que CNN y el criterio de decisión queda fuertemente vinculado a la regla de los k-vecinos en este caso con $k = 3$.

Figura 5: *Undersampling* de la clase mayoritaria vía ENN.



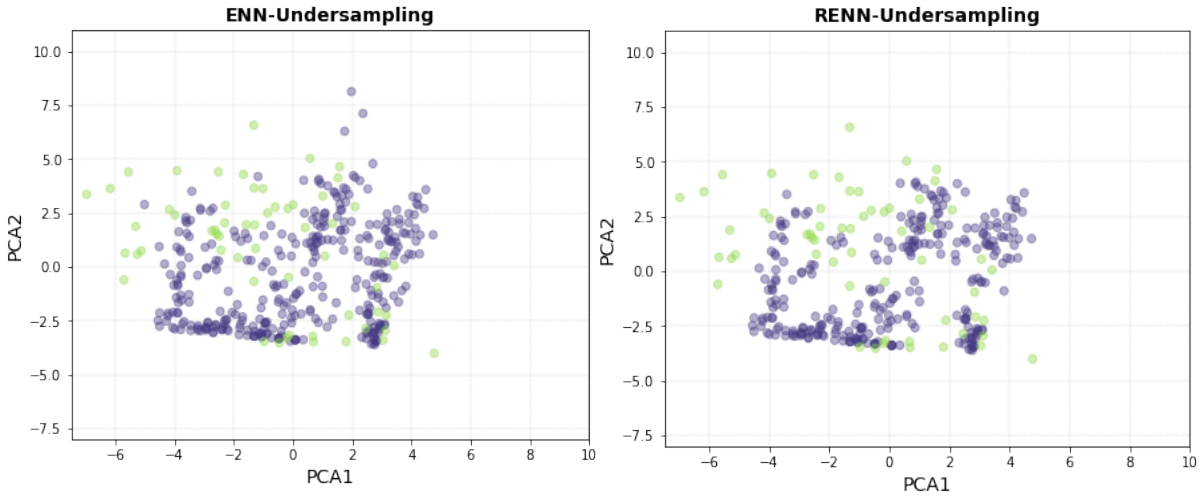
Fuente: Elaboración propia con una submuestra de datos tomados del *dataset* usado en la parte empírica de este trabajo (KKBOX).

Como es posible observar en la **Figura 5** el comportamiento de ENN sobre los datos de ejemplo es bastante similar a *Tomek Links,* aunque la limpieza en este caso parece más exhaustiva. Concretamente para este ejemplo la categoría mayoritaria se reduce hasta las 340 instancias con este tipo de limpieza, 75 instancias más de las que se eliminaban con *Tomek Links*. La representación espacial de los casos es muy similar a la obtenida con *Tomek Links*, aunque se constata una mayor limpieza de instancias de la clase mayoritaria en el segundo cuadrante y en la zona central de la nube de puntos.

Este proceso puede repetirse varias veces sobre el conjunto de datos (iterativamente) lo que se ha dado a llamar ***Repeated Edited Nearest Neighbours***. Esta solución fue adoptada también por Tomek (1976a), bautizada como "*unlimited edited*" aunque el número de repeticiones es finito en tanto en cuanto llegará un momento que por la propia estructura de ENN no podrán eliminarse más instancias del conjunto de datos.

En la **Figura 6** se comparan las distribuciones de datos resultantes para los datos de prueba. Se aprecia de forma bastante clara un mayor número de casos eliminados en las mismas regiones que apuntábamos anteriormente, en especial en el centro de la nube de puntos, sumando un total de 48 instancias más eliminadas.

Figura 6: Comparación gráfica de la aplicación de ENN y RENN.



Fuente: Elaboración propia con una submuestra de datos tomados del *dataset* usado en la parte empírica de este trabajo (KKBOX).

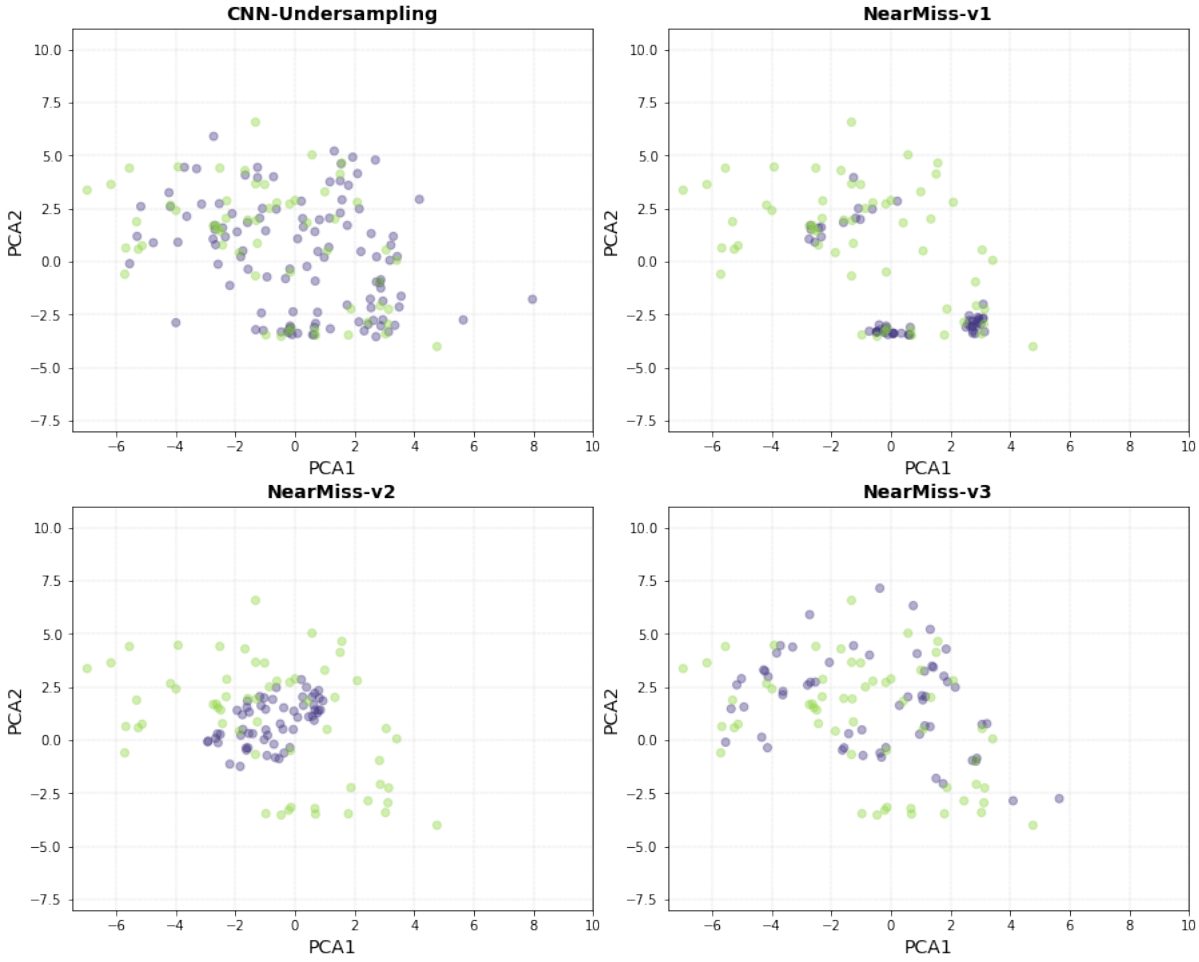
3.2.1.5. Soluciones NearMiss.

Una última familia de soluciones basadas en k-vecinos que conviene mencionar es el aportado por el **método *NearMiss*** (Zang and Mani, 2003). Es una sencilla técnica de decisión específicamente pensada para eliminar instancias de la clase mayoritaria en base a k-vecinos cercanos. Las distintas versiones de estos algoritmos se centran en la elección de los vecinos a utilizar para la eliminación de instancias en la clase mayoritaria. Aunque el objetivo principal sigue siendo reducir el tamaño de la clase mayoritaria, ha de subrayarse que este método se centra fundamentalmente en seleccionar el grupo de instancias a conservar, que induce un pequeño matiz diferente a la motivación de los dos métodos anteriores. Lo novedoso de este método es que el dato más importante para la decisión es la distancia media a los vecinos de clases opuestas. El funcionamiento es el siguiente:

- *Versión 1:* Se conservan aquellas instancias de la clase mayoritaria cuya distancia media a los 3 vecinos más próximos de la clase minoritaria sea menor. Se seleccionan los casos con distancias medias menores hasta satisfacer el mismo número de instancias en cada clase o una ratio predefinida.
- *Versión 2*: Selecciona los casos de la clase mayoritaria que obtienen una distancia media menor a sus 3 ($k = 3$) vecinos más lejanos de la clase minoritaria. Nuevamente hasta igualar el número de instancias en cada clase o satisfacer un ratio o número predefinido.
- *Versión 3*: Esta última versión opera en 2 fases. En la primera fase se selecciona un número dado de instancias de la clase mayoritaria más cercanas a las instancias de la clase minoritaria. Obtenida esa primera selección de instancias,

en una segunda fase, tan solo se conservan aquellas instancias con una distancia media mayor a los 3 vecinos más cercanos de la clase minoritaria.

Figura 7: Comparativa de las 3 implementaciones distintas del método *NearMiss* con los datos originales.



Fuente: Elaboración propia con una submuestra de datos tomados del *dataset* usado en la parte empírica de este trabajo (KKBOX).

En la **Figura 7** se aprecian las importantes discrepancias en las distribuciones espaciales de las clases usando uno u otro método de entre los explicados. En la parte empírica de este trabajo se podrán valorar las diferencias en la clasificación que resultan de la implementación de estos métodos, ofreciendo importantes diferencias. La única versión que será estudiada en el trabajo por lo observado en las representaciones gráficas del ejemplo es la versión 3. Las versiones 1 y 2 de estas aplicaciones no resultan útiles para los objetivos de este proyecto.

3.2.2. Técnicas de *oversampling*.

En este punto se integran aquellas técnicas que operan sobre los datos aumentando el tamaño de la muestra original. En problemas de desbalanceo de datos las técnicas

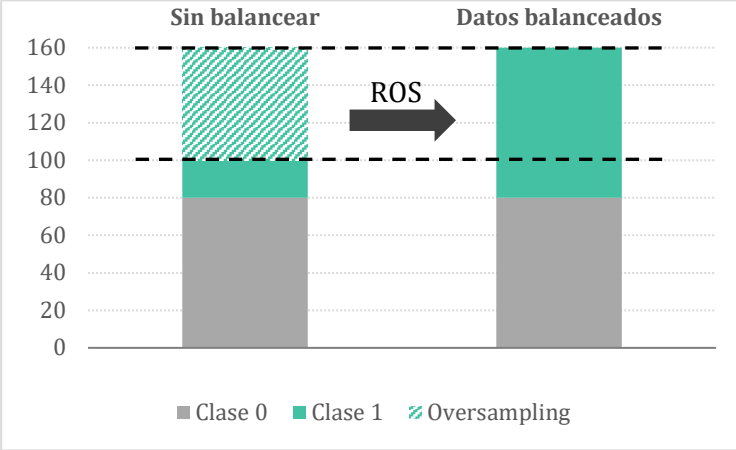
de *oversampling* tienen como finalidad replicar instancias en la clase minoritaria a partir de la información de los casos existentes o reales. Nótese el riesgo y la importancia que estas operaciones conllevan al generar información que no está incluida realmente en los datos. En las técnicas de *undersampling* se proponen métodos o modelos que describen en cada caso qué instancias de la clase mayoritaria han de conservarse o eliminarse, siempre operando sobre datos existentes.

En las siguientes páginas se propone la explicación de los tres métodos de *oversampling* más populares: uno no heurístico, el ***Random Oversampling*** y dos heurísticos sintéticos ***Synthetic Minority Over-Sampling Technique*** (***SMOTE***) y ***Adaptative Synthetic Sampling*** (***ADASYN***).

3.2.2.1. Random oversampling.

El *Random Oversampling* (ROS) es la técnica de *oversampling* más sencilla. Es el método no heurístico dentro de este grupo y básicamente replica o copia las instancias de la clase minoritaria hasta satisfacer el número de instancias o ratio deseados (normalmente hasta igualar el número de instancias de la clase mayoritaria).

Figura 8: Ejemplo de la aplicación de *Random Oversampling*



Las ventajas e inconvenientes del uso de este método son dispares en la literatura. Algunos autores señalan que no permite la mejora en clasificación de la clase minoritaria y en otros casos se demuestra un buen rendimiento en estas labores (He and Ma, 2013).

Lo que sí parece generar consenso es: primero, que el ROS puede aumentar la probabilidad de que se produzca un sobreajuste en las posteriores tareas de clasificación. Otra ventaja obvia es que el coste computacional de la clasificación

posterior es más elevado que con en la aplicación del *random undersampling* (alternativa no heurística).

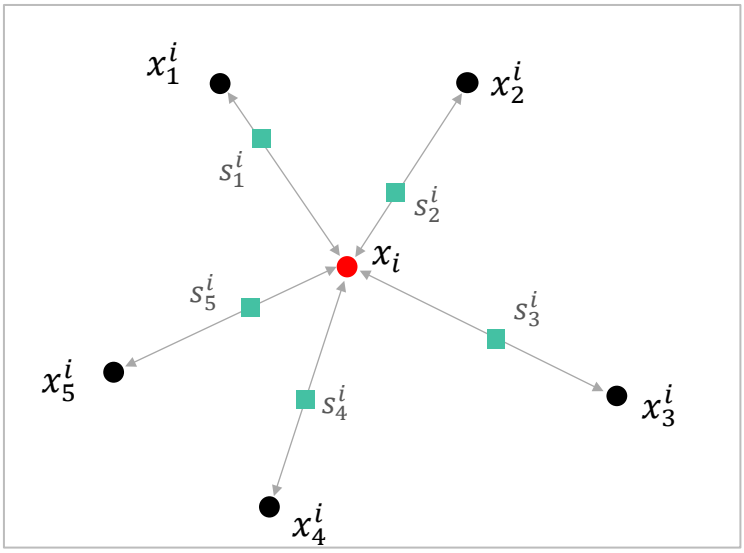
En el caso concreto de este trabajo y como veremos más adelante se les ofrece un gran valor a estas técnicas no heurísticas por dos razones; su sencillez y el hecho de que son dos de las técnicas que muestran mejores rendimientos en el problema de clasificación planteado. No obstante, el *oversampling* aleatorio ha sido en ocasiones descrito en la literatura como un método que no mejora el reconocimiento de la clase minoritaria en problemas de clasificación (Japkowicz, 2000) y que además carga con las limitaciones o riesgos ya descritos.

3.2.2.2. Synthetic Minority Oversampling Technique (SMOTE).

Para salvar las deficiencias explicadas antes, SMOTE se presenta como el método heurístico de *oversampling* más referenciado en la literatura. Este método genera instancias sintéticas a partir de las muestras o instancias de la clase minoritaria.

En esta técnica de *oversampling* vuelve a tener como apoyo el método de los k-vecinos. La propuesta de SMOTE (Chawla *et al.*, 2002) se basa en la réplica de instancias de la clase minoritaria tomando cada caso e introduciendo nuevos ejemplos sobre las líneas de unión de sus k-vecinos más cercanos. Si tenemos una instancia objetivo x_i y seleccionamos sus 5 vecinos más próximos $\{x_1^i, x_2^i, x_3^i, x_4^i, x_5^i\}$, en una primera iteración de un proceso de *oversampling* vía SMOTE, es posible la creación de 5 nuevas instancias $\{s_1^i, s_2^i, s_3^i, s_4^i, s_5^i\}$ sobre las líneas de unión entre la instancia x_i y sus vecinos más próximos. En la **Figura 9** se representa este ejemplo para ilustrar el funcionamiento del SMOTE.

Figura 9: Ejemplo gráfico del funcionamiento del SMOTE.



El proceso explicado para un conjunto de datos real crearía una "malla" de líneas entre todas las instancias de la clase minoritaria con los 5 vecinos más próximos de cada una y crearía, sobre esas líneas, las muestras sintéticas necesarias hasta satisfacer el número de instancias deseado (hasta igualar el número de la clase mayoritaria usualmente). Lo que se consigue es hacer más densa la región de decisión de la clase minoritaria.

El procedimiento es el siguiente:

- X_{min} → Conjunto de instancias de la clase minoritaria
- S → Conjunto de muestras sintéticas a reproducir
- k → Número de vecinos a computar

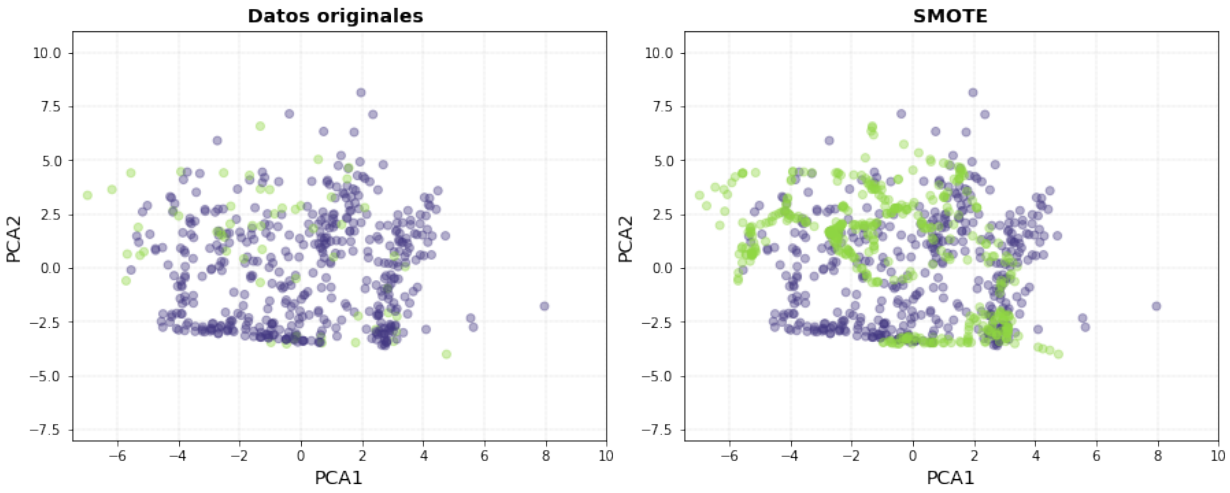
Los pasos de implementación de SMOTE son los siguientes:

- Se computan los k vecinos de X_{min}
- Se computan o reproducen las S instancias sintéticas, generadas de la siguiente forma:

$$s_j^i = x_i + \lambda \cdot (x_j^i - x_i), j = 1, \dots, k$$

Donde s_j^i es la nueva instancia a crear; x_i la instancia de la clase minoritaria sobre la que se computan los vecinos; x_j^i uno de los vecinos a la instancia x_i y λ es un número aleatorio del rango [0,1].

Figura 10: Oversampling de la clase minoritaria vía SMOTE



Fuente: Elaboración propia con una submuestra de datos tomados del *dataset* usado en la parte empírica de este trabajo (KKBOX).

En la **Figura 10** vemos el resultado del *oversampling* vía SMOTE sobre los datos de ejemplo. Se aprecia cómo la generación de las instancias sintéticas sigue las líneas de distancia de las instancias originales. Las nuevas instancias se distribuyen en el espacio de forma encadenada entre los casos originales.

3.2.2.3. Adaptative Synthetic Sampling (ADASYN).

Como aplicación particular del método SMOTE tenemos el modelo ADASYN (He *et al.*, 2008). La aplicación del método es la misma, pero ADASYN añade una distorsión a los datos para hacerlos más "realistas". Esta distorsión parte de la introducción de una ratio basada en el número de vecinos de la clase mayoritaria. Es decir, ADASYN pretende tener en cuenta la clase mayoritaria a la hora de generar las instancias sintéticas.

En el momento de computar los k vecinos de las X_{min} instancias de la clase minoritaria se calcula una ratio r_i que se explica de la siguiente forma:

$$r_i = \Delta_i / k$$

Donde Δ_i es el número de k vecinos más próximos de la clase mayoritaria por lo que $r_i \in [0,1]$. Después de normaliza r_i de acuerdo a la siguiente expresión:

$$\widehat{r_i} = r_i \Big/ \sum_{i=1}^{X_{may}} r_i$$

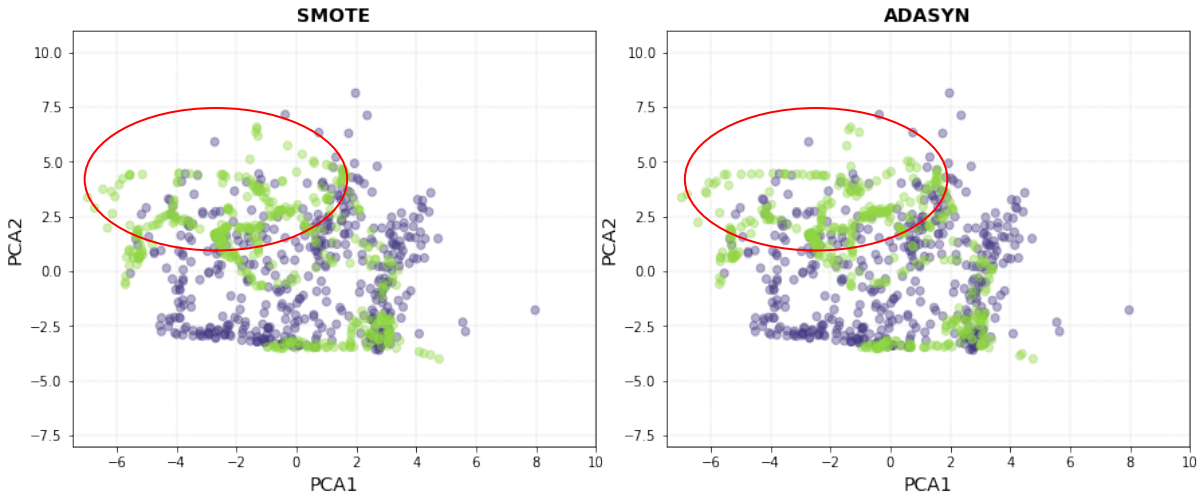
Donde X_{may} es el cardinal del conjunto de instancias de la clase mayoritaria y $\widehat{r_i}$ es una distribución de probabilidad que cumple que $(\sum_{i=1} \widehat{r_i} = 1)$.

A la hora de calcular el número de instancias sintéticas para cada instancia de la clase minoritaria x_i, se aplicaría la siguiente distorsión o corrección si se quiere:

$$g_i = \widehat{r_i} \cdot S$$

Donde g_i será el número de instancias a generar por cada x_i y S recordemos que era el conjunto de instancias sintéticas que habríamos de generar en el balanceo. La generación de instancias respetaría el mismo procedimiento que veíamos en SMOTE pero estableciendo el control del número de instancias a crear que nos da g_i.

Figura 11: Comparación gráfica entre los métodos sintéticos de *oversampling* SMOTE y ADASYN



Fuente: Elaboración propia con una submuestra de datos tomados del dataset usado en la parte empírica de este trabajo (KKBOX).

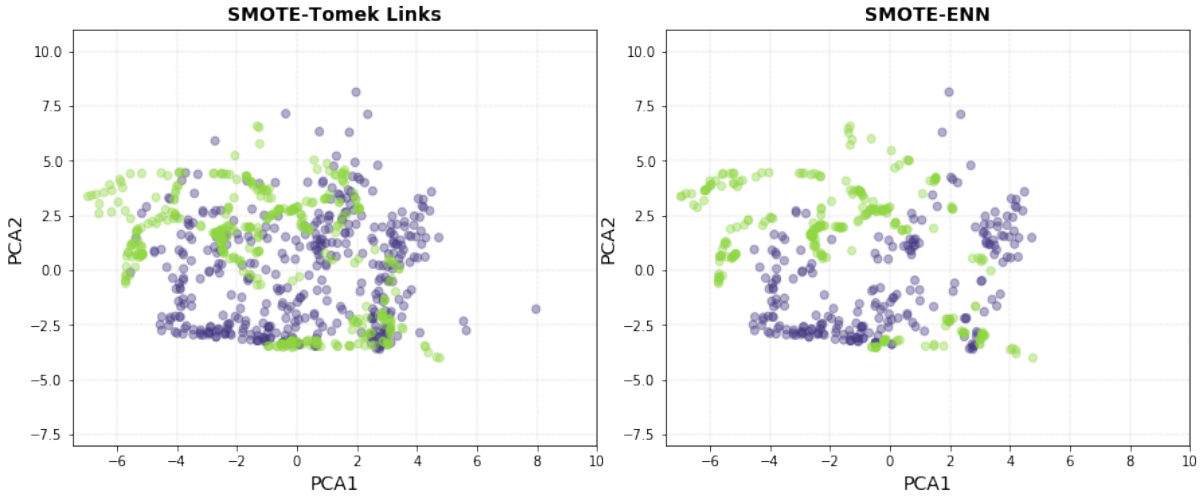
Vemos que, obviamente los dos métodos arrojan resultados muy similares, pero en el caso del ADASYN *oversampling* se aprecia una mayor densidad de instancias de la clase minoritaria en regiones con menor presencia de casos de la clase mayoritaria. Se refuerza levemente la cantidad de casos en regiones que parecen dominadas por la clase minoritaria. Los rendimientos son muy parecidos, estas dos técnicas de replicación sintética de instancias generan resultados similares mostrando diferencias pequeñas acorde a la gran similitud matemática de las dos técnicas.

3.2.3. Técnicas híbridas de *resampling*.

Por último, conviene mencionar técnicas híbridas de *undersampling* y *oversampling* que pueden ser combinadas en el balanceo de datos. Lo más habitual resulta de la aplicación de métodos ya explicados con anterioridad, combinando una técnica de eliminación de instancias redundantes en la clase mayoritaria y posteriormente la aplicación de una técnica de *oversampling* en la clase minoritaria.

Las dos técnicas híbridas más habituales son SMOTE+*Tomek Links* y SMOTE+ENN. La principal diferencia en la aplicación de una u otra técnica se circunscribe a efectuar una limpieza más o menos intensa de la clase mayoritaria, lo que afectará posteriormente a la creación de instancias sintéticas. Previsiblemente la limpieza de la clase mayoritaria será sustancialmente más intensa con el uso de ENN que con el uso de *Tomek Links*.

Figura 12: Comparación de la aplicación de las dos técnicas híbridas SMOTE+*Tomek Links* y SMOTE+ENN.



Fuente: Elaboración propia con una submuestra de datos tomados del *dataset* usado en la parte empírica de este trabajo (KKBOX).

Se observa en la **Figura 12** lo ya apuntado para los datos de prueba. La limpieza de la clase mayoritaria con SMOTE+ENN es mucho más intensa estableciendo una separación entre las clases mucho más profunda que la aportada por SMOTE+Tomek Links.

3.3. Técnicas y modelos para el contraste de las técnicas de balanceo de datos.

En este apartado se presentan; primero, las técnicas sobre las que se analizarán las diferencias de las técnicas de resampling en tareas de clasificación, y segundo, las métricas de rendimiento que permiten valorar esas diferencias.

Como primer punto se aborda una breve explicación de las tres técnicas de clasificación usadas para este trabajo (regresión logística, bosques aleatorios y boosting) que permiten la extracción de resultados de predicción comparables, ofreciendo un valor central a las predicciones probabilísticas en los problemas de clasificación binaria. En una segunda parte de este apartado se explican en profundidad qué métricas de rendimiento resultan más pertinentes en problemas de clasificación binaria con clases desbalanceadas. La medición global de los modelos quedará vinculada a métricas que permitan valorar el ajuste en la categoría que presenta la característica de interés y no solo el ajuste global del modelo.

3.3.1. Modelos para la evaluación de las técnicas de balanceo.

Los modelos de contraste elegidos para este trabajo (Regresión Logística, Random Forest, Boosting) son técnicas referenciadas ampliamente en la literatura, en

problemas de clasificación binaria y concretamente en relación con el problema de fuga de clientes. Como ya hemos visto, otros dos clasificadores muy utilizados en este tipo de trabajos son las Máquinas Vector Soporte y las Redes Neuronales. En este caso particular se han elegido estas tres técnicas por varias razones: en primer lugar, la sencillez interpretativa derivada del trabajo con árboles de decisión y un modelo lineal; segundo, la posibilidad de obtener predicciones probabilísticas de pertenencia a clase que permiten la extracción de todas las métricas de rendimiento que se pretende utilizar; tercero, el buen rendimiento contrastado de estas técnicas en problemas similares al propuesto. Añadidamente, la inclusión de la regresión logística permite establecer una solución de partida teórica, que no incurre en problemas de sobreajuste, a partir de la cual se puede valorar mejor las pérdidas y ganancias de las técnicas de resampling.

Alcanzar el máximo nivel de predicción correcta, no es uno de los objetivos principales de este trabajo, nos centramos en la evaluación del efecto que las técnicas de resampling tienen sobre las tareas de clasificación. Es por ello que en la implementación de las técnicas no se dedica una parte del trabajo a evaluar distintas combinaciones de hiperparámetros. Los clasificadores se implementan ajustando los hiperparámetros a los valores usualmente definidos en la teoría.

3.3.1.1. Regresión logística.

La primera de las técnicas seleccionadas para el contraste de las técnicas de balanceo es la Regresión Logística. Técnica ampliamente conocida y usada en problemas de clasificación binaria, provee una solución teórica para un problema en que la variable dependiente y es de tipo binario. La noción de partida es el de una regresión lineal en la que el ajuste de la variable dependiente para un conjunto de variables predictoras x_i vendría dado por la expresión:

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i + e$$

Esta solución de ajuste por mínimos cuadrados otorga coeficientes (betas) a las variables predictoras. Esta solución se reformula para el caso binario añadiendo una restricción que permita predicciones entre [0,1]. En relación a este problema se puede formular p_i como la probabilidad de que y tome el valor 1 para unos valores dados de x_i.

$$p_i = P(y = 1|x_i)$$

Y por tanto p_i será:

$$p_i = \beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i$$

Para que estos valores pertenezcan al intervalo [0,1] y así obtener una predicción interpretable como la probabilidad de pertenecer o no a la clase 1 (por extensión pertenecer a la clase 0), es necesaria una función de enlace que permita añadir esa restricción. En el caso de la regresión logística se adopta la siguiente solución:

$$p_i = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i)}}$$

Además es posible ir más allá de forma que:

$$\log \frac{p_i}{1 - p_i} = \log \left(\frac{1}{1+e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i)}} \Big/ \frac{e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i)}}{1+e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i)}} \right) =$$
$$\log \left(\frac{1}{e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i)}} \right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i = g_i$$

Esta noción es la que está detrás del elemento distintivo de la regresión logística, los o*dds ratios.* Este elemento se interpreta como el cociente entre dos odds asociados: el que define el modelo de base y el que resulta de un incremento en las variables predictoras x_i.

$$Odd_i = \frac{p_i}{1 - p_i} = e^{\beta_0} \cdot \prod_{i=1}^{k} e^{\beta_i x_i}$$

Por tanto, los coeficientes en β en la regresión logística definen los cambios en la probabilidad de pertenencia a la clase 1 (o experimentar el evento) cuando se incrementa en una unidad el valor de la variable x_i a la que nos estemos refiriendo.

En el caso del presente trabajo se trabaja con una regresión tipo LASSO. La regresión LASSO funciona de la misma manera que la regresión logística convencional. Es decir, la combinación lineal de los regresores más un error. La principal diferencia es la función de pérdida aplicada en el cálculo de esos parámetros. En lugar de minimizar la suma de los errores cuadráticos se aplica una penalización a los coeficientes.

En cualquier caso, partimos de la premisa que las predicciones de una regresión logística adquieren esta forma:

$$\hat{y} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \cdots + \beta_i x_i)}}$$

Lo que pretendemos minimizar en un trabajo de clasificación es el coste de las predicciones es decir pretendemos que el siguiente valor sea mínimo:

$$\sum_{j}^{n} (y_j - \hat{y}_j)$$

La regresión de tipo LASSO añade un término en la expresión que tiene en cuenta la magnitud total y absoluta de los coeficientes calculados en la regresión de forma que:

$$\sum_{j}^{n}(y_j - \hat{y}_j) + \lambda \sum_{i}^{p} |\beta_i|$$

El término añadido además incluye un parámetro λ que ajusta la dimensión de la penalización. Si es igual a 0 obtendremos una regresión ordinaria, pero si es lo suficientemente grande permitirá el ajuste de coeficientes iguales a 0. Este método se presenta como una buena herramienta para la selección de variables en problemas que involucran un gran número de ellas y además permite evitar problemas de colinealidad o varianza compartida que puedan afectar a las predicciones resultantes.

En el caso particular de este trabajo la regresión logística tipo LASSO permite implementar el modelo depurado sin acudir a un método de selección de variables por Akaike o bayesiano.

3.3.1.2. Random Forest o Bosques Aleatorios.

La segunda técnica de clasificación elegida para la evaluación de las técnicas de *resampling* son los bosques aleatorios o *random forest* por su terminología inglesa. Esta técnica se basa en generar un gran número de árboles de decisión que difieren unos de otros en dos elementos:

- Se usan distintas submuestras aleatorias (con reemplazamiento) de entre los datos de entrenamiento (*bagging*).
- Se usan distintos grupos de variables predictoras en cada árbol.

Esta técnica se basa en la evaluación de los clasificadores débiles, que serían los distintos árboles de decisión construidos que son combinados para la definición de un clasificador fuerte, que representaría la predicción final del modelo. En concreto ésta es una técnica que permite el trabajo con árboles de decisión en condiciones de mayor estabilidad. La combinación de árboles distintos (gracias a las muestras *bagging*), corrige la volatilidad de los árboles de decisión simples cuando hay cambios en los datos.

El procedimiento general de esta técnica parte de un conjunto de datos para los que tenemos valores para un conjunto de variables independientes $X = (X_1,\ X_2 \dots X_p)$ y valores para una variable dependiente Y, en este caso de tipo binario, que queremos predecir con una función $f(X)$. En la predicción resulta una función de pérdida del tipo $L(Y, f(X))$ donde el error de clasificación será.

$$L(Y, f(X)) = \begin{cases} 0 \; si \; Y = f(X) \\ 1 \; si \; Y \neq f(X) \end{cases}$$

El objetivo del clasificador será minimizar el valor esperado de la función de pérdida, es decir, la búsqueda del menor error posible en la predicción.

$$E_{XY}(L(Y, f(X)))$$

Minimizar la anterior expresión permite ofrecer la siguiente variante condicional.

$$f(x) = E(Y|X = x)$$

Lo que se pretende es ofrecer como valor de la predicción final, el valor más veces predicho para cada uno de los casos del conjunto de datos, en todos los árboles. Este procedimiento de predicción es lo que se llama voto a la clase de la mayoría. Estas probabilidades representan la proporción de veces que un individuo ha sido asignado a cada clase. La clase que alcanza el máximo de esas probabilidades es la clase predicha para el individuo.

Recapitulando para ofrecer un resumen del funcionamiento del algoritmo, se definen las siguientes fases:

1. Selección aleatoria de una muestra con reposición de tamaño N (bootstrap) una en cada iteración o árbol construido.
2. Construcción de los B árboles definidos por el modelo:
   a. Selección aleatoria de un número de variables de entre los p predictores (usualmente en problemas de clasificación $p/3$)
   b. Se mide el error para cada X seleccionada usando los individuos que han quedado fuera de la muestra (Out Of Bagging)
   c. Se escoge la mejor partición de las X incluidas y se repite este proceso para cada nodo.
3. Predicciones de la forma explicada, a través del voto a la clase más frecuente.

Esta técnica ofrece mejores rendimientos y mayor estabilidad en las predicciones que los árboles simples. Por otro lado, es una técnica sencilla, no solo por el trabajo con árboles, sino que ofrece la posibilidad de buenos ajustes con parámetros teóricos, que en este caso concreto resultan en el trabajo con 100 árboles y $p/3$ predictores de entrada a los árboles, como ya se apuntaba.

3.3.1.3. Boosting.

Otra técnica que garantiza estándares de comparabilidad y se fundamenta además en el trabajo con árboles de decisión es el grupo de técnicas que se engloban en lo que se denomina "boosting". La diferencia con respecto a la idea propuesta por los Bosques Aleatorios, radica en encadenar los árboles de decisión para alcanzar un

clasificador robusto en vez de promediar los resultados de los árboles construidos. En este sentido, lo que propone esta técnica es ofrecer un mayor peso en cada iteración a los casos o individuos mal clasificados en la fase anterior. Por otro lado, este conjunto de técnicas opera con el conjunto de datos completo.

Existen diferentes implementaciones, siendo el *boosting* adaptativo el más popular. De forma general, se puede definir el *boosting* como un modelo secuencial aditivo en el que cada árbol de la secuencia se ajusta a los residuos del árbol anterior. Se busca un estimador robusto $H(x)$ y lo que hacemos en cada iteración a través de los clasificadores débiles $h(t)$ de forma que:

$$h_i(0) = 0$$
$$h_i(1) = f_1(x_i) = h_i(0) + f_2(x_i)$$
$$h_i(2) = f_1(x_i) + f_2(x_i) = h_i(1) + f_2(x_i)$$
$$\dots$$
$$h_i(t) = \sum_{i=1}^{t} f_k(x_i) = h_i(t-1) + f_t(x_i)$$

La función h tiene como objetivo aportar las predicciones de clase para los elementos del conjunto de datos (n). Por lo tanto, el objetivo de esa predicción será cometer el menor error posible, es decir, minimizar la función de pérdida. Nuestra función objetivo puede ser por tanto $e(t)$, función que queremos hacer mínima.

$$e_t = \sum_{i=1}^{n} l\big(y_i, h_i(t)\big)$$

Conociendo este error, estamos en disposición de calcular las ponderaciones de los clasificadores débiles $h(t)$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - e_t}{e_t}\right)$$

Como se observa cuanto mayor sea el error menor será la ponderación asociada al clasificador débil $h(t)$ que estemos tratando.

Si consideramos $l\big(y_i, h_i(t)\big) = D_t(i)$ es posible expresar la nueva distribución de pesos como:

$$D_{t+1}(i) = \frac{D_t(i)\exp\left(-\alpha_t\, y_i, h_i(t)\right)}{z_t}$$

Por su parte en la distribución de pesos las clasificaciones correctas contarán con un menor peso en la siguiente etapa. El factor z_t es usado para conseguir que $D_{t+1}(i)$ obtenga valores entre 0 y 1, pudiendo así interpretarse como una probabilidad.

Con todo, el clasificador robusto $H(x)$ será un promedio ponderado de los clasificadores obtenidos:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t \; h(t)\right)$$

Resumidamente en el caso del *boosting* lo que se pretende es primero obtener una serie de clasificadores intermedios que ponderadamente conformarán un clasificador robusto final. En cada etapa *t,* además, se otorgará una menor importancia a los individuos del conjunto de datos que hayan sido correctamente clasificados en la etapa anterior. Una variante más reciente es el *gradient boosting* aplicación concreta con la que trabajaremos. En este caso particular lo que se trata, es proponer una función de error de la que queremos minimizar su gradiente.

3.3.2. La medición del rendimiento de modelos con datos desbalanceados.

En este punto conviene un repaso a las distintas métricas de rendimiento que permitirán valorar el desempeño de los modelos de predicción y el impacto de las técnicas de balanceo de datos. Las motivaciones principales en la selección de las distintas métricas utilizadas son dos:

- Permitir la evaluación de las predicciones para cada una de las clases en un contexto de desbalanceo de datos.
- Garantizar un esquema comparativo en cuanto para cuantificar el impacto de las técnicas de balanceo sobre los modelos de predicción.

3.3.2.1. Métricas de rendimiento derivadas de la matriz de confusión.

La medición del rendimiento de los modelos en problemas de clasificación, es una cuestión ampliamente referenciada y que depende del tipo de problema que se enfrente. Ante un problema de clasificación la solución más habitual para la evaluación del desempeño de un modelo sería la validación cruzada y posterior medición de la *exactitud* (*accuracy* por su terminología inglesa).

La validación de los modelos consiste en la división de los datos en dos grupos. Un primer grupo de entrenamiento sobre el que se construyen los modelos y un segundo grupo de datos de test sobre los que se validan los resultados. En problemas de clasificación binaria, sobre los datos de test, se realizarían las predicciones de pertenencia a una u otra clase. Consecuentemente la exactitud del modelo vendría determinada por la proporción de casos correctamente clasificados sobre el total de casos.

La *exactitud* representa una medida global de rendimiento que puede incurrir en un sesgo respecto a la evaluación del modelo sobre datos concretos. Cuando se trabaja con conjuntos de datos desbalanceados esta medida no representa el desempeño real, o que al menos puede interesar, de un modelo. En conjuntos de datos altamente desbalanceados el valor de la exactitud puede ser muy alto por la correcta clasificación de la clase mayoritaria; normalmente la parte de la muestra que no presenta el evento de interés. En este sentido, los algoritmos tienen una probabilidad mucho más alta de clasificar correctamente la clase mayoritaria. Estamos ante problemas en los que es posible encontrar proporciones de la clase minoritaria del 1 o 2%, usuales en la detección de fraude o la predicción de fuga de clientes (Chawla et al., 2002; Zhu, Baesens and vanden Broucke, 2017 o Deng et al., 2019). Por lo tanto, la asignación constante de la etiqueta de la clase mayoritaria en una predicción, obtendría una exactitud del 99%. Estaríamos, por tanto, ante un modelo que podría simplemente obviar la clase minoritaria para alcanzar un alto valor de la exactitud.

Por todo esto, el uso de la exactitud como medida del desempeño de los modelos en contextos con alto desbalanceo de clases no es recomendable. Como alternativas se presentan medidas bien referenciadas (véase por ejemplo Hand, 2012) y de uso cotidiano en trabajos con datos afectados por el desbalanceo de clases (Burez and Van den Poel, 2009; Verbeke *et al.*, 2012; Zhang *et al.*, 2019; Xuan, Liu and Li, 2018; etc.).

Para abordar las distintas medidas de rendimiento o desempeño de un modelo predictivo partimos de la matriz de confusión que resulta de la aplicación de las predicciones. En problemas de clasificación binaria una matriz 2x2 ordena los resultados de las predicciones aportadas por un modelo en cuatro resultados posibles. Con las etiquetas asignadas partimos de la premisa de que la clase minoritaria es representada por la clase "1", siendo aquella que presenta la característica de interés.

Tabla 4: Matriz de confusión.

		PREDICCIÓN	
		0	**1**
OBSERVACIÓN	**0**	VN	FP
	1	FN	VP

- *VN = Verdaderos negativos. Se predice que no presentan la característica de interés y realmente no la presentan.*
- *FP = Falsos positivos. Se predice que presentan la característica de interés y realmente no la presentan.*
- *FN = Falsos negativos. Se predice que no presentan la característica de interés y realmente la presentan.*

- *VP = Verdaderos positivos. Se predice que presentan la característica de interés y realmente la presentan.*

Partiendo de la matriz de confusión la exactitud o "accuracy" quedaría definida de la siguiente forma:

$$accuracy = \frac{VN + VP}{VN + VP + FN + FP}$$

La suma de los valores en la diagonal contabiliza el número de casos correctamente clasificados y la suma de todos los valores de la matriz el total de casos predichos. Como apuntábamos la exactitud ofrece una medida global que no permite evaluar el poder de clasificación del modelo en relación al evento estudiado. Para ello partiendo de la misma matriz de confusión es posible calcular tres medidas que sí permiten valorar el rendimiento del modelo para la característica de interés. Estas medidas son la *precisión,* la *sensibilidad*, *exhaustividad* o *recall* por su denominación en inglés y el *valor f* o *f-score.*

- La *precisión* se formaliza de la siguiente manera:

$$P = \frac{VP}{VP + FP}$$

La precisión representa la proporción de predicciones correctas sobre el total de casos que presentan la característica de interés. En nuestra posterior aplicación práctica la precisión representaría la proporción de las predicciones de fuga que son fugas realmente.

- La *sensibilidad* o *recall* es:

$$S = \frac{VP}{VP + FN}$$

La sensibilidad a diferencia de la precisión representa la proporción correcta de predicciones a la clase que presenta la característica de interés. Para el caso concreto que abordaremos más adelante obtendríamos la proporción de fugas de clientes que hemos sido capaces de predecir correctamente.

- *F-score:*

El valor F es una medida resumen que opera con las dos componentes anteriores. Matemáticamente es la media armónica entre la precisión y la sensibilidad. Se establece como una medida resumen de dos componentes que evalúan la calidad de las predicciones. El valor F se expresa como:

$$F\_score = 2 \cdot \frac{precision \cdot sensibilidad}{precision + sensibilidad}$$

Como vemos en el valor F se otorga la misma importancia a la precisión y sensibilidad[2] y ofrece una medida global de rendimiento del modelo, centrada en la predicción de la característica que tratamos de predecir. En ese sentido, sería más adecuada que la exactitud cuando se trata de evaluar la calidad de las predicciones del evento o característica

Las métricas detalladas anteriormente tienen su aplicación a modelos multiclase de forma sencilla como las medias de cada indicador para todas las clases. Las métricas introducidas para la versión binaria podrían quedar reformuladas para un problema multiclase como se indica en la Tabla 5.

Tabla 5: Exactitud, precisión, exhaustividad y f-score para versión multiclase.

Métrica	Formulación	Notación
Exactitud	$acc_i = \frac{VN_i + VP_i}{VN_i + VP_i + FN_i + FP_i} \rightarrow acc = \frac{\sum_{i=1}^{k} acc_i}{k}$	$acc_i = exactitud\ en\ cada\ clase$ $acc = exactitud\ del\ modelo$ $k = número\ de\ clases$ $\forall i = 1, \dots, k$
Precisión	$p_i = \frac{VP_i}{VP_i + FP_i} \rightarrow p = \frac{\sum_{i=1}^{k} p_i}{k}$	$p_i = precisión\ en\ cada\ clase$ $p = precisión\ del\ modelo$ $k = número\ de\ clases$ $\forall i = 1, \dots, k$
Sensibilidad	$S_i = \frac{VP_i}{VP_i + FN_i} \rightarrow S = \frac{\sum_{i=1}^{k} s_i}{k}$	$S_i = sensibilidad\ en\ cada\ clase$ $S = sensibilidad\ del\ modelo$ $k = número\ de\ clases$ $\forall i = 1, \dots, k$
F-score	$F_i = 2 \cdot \frac{r_i \cdot p_i}{r_i + p_i} \rightarrow F = \frac{\sum_{i=1}^{k} F_i}{k}$	$F_i = F\ score\ en\ cada\ clase$ $F = F\ score\ del\ modelo$ $\forall i = 1, \dots, k$

Como se observa las métricas explicadas se resuelven en problemas multiclase a través de las medias aritméticas de los indicadores en las *k* clases. Las métricas se calculan dentro de cada clase frente al resto de clases agregadas. Es decir, para cada clase k_i *se* calculan los indicadores como si fuese ésta la clase de interés. Las medias de las métricas para cada clase son las medidas resumen de evaluación de un modelo de clasificación con más de dos clases o etiquetas.

[2] Aunque esto no tiene por qué ser así, es posible ofrecer un coeficiente de ponderación β a la precisión de tal modo que el valor f quede redefinido como:

$$F_\beta = (1 + \beta^2) \cdot \frac{precisión \cdot sensibilidad}{\beta^2 \cdot (precisión + sensibilidad)}$$

3.3.2.2. Curvas ROC (Receiver Operating Characteristic).

Las medidas introducidas en el punto anterior son preámbulo de un método de evaluación de modelos que permite además añadir un soporte gráfico de gran utilidad para la comparación.

Conocido ya el concepto de *sensibilidad* se introduce el de *especificidad* que se expresa como:

$$E = \frac{VN}{VN\ +\ FP}$$

Nótese que la especificidad representa la misma medida que la sensibilidad para la clase que no presenta el evento de interés. Representa la proporción entre los casos que no presentan la característica que somos capaces de predecir correctamente.

Para la construcción de las curvas ROC se trabaja con el complementario de la especificidad:

$$1 - E = 1 - \frac{VN}{VN\ +\ FP} = \frac{VN + FP}{VN + FP} - \frac{VN}{VN + FP} = \frac{FP}{VN + FP}$$

El resultado como se puede observar es la proporción de falsos positivos entre los negativos. Es decir, la proporción de predicciones erróneas de entre las predicciones del grupo que no presenta la característica.

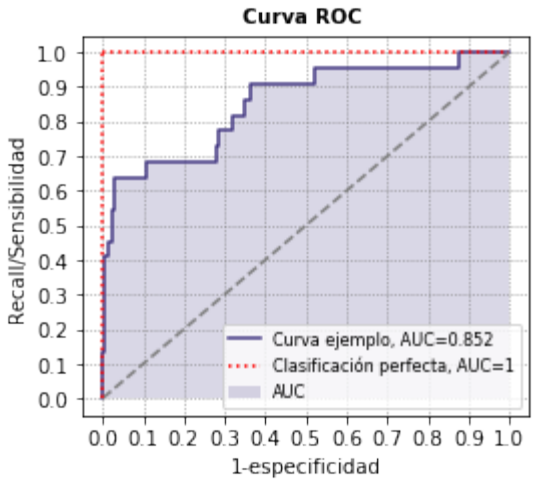
Se constata, además, que esta proporción es a su vez la complementaria de la *sensibilidad*. Por tanto, la *sensibilidad* y *especificidad* representan probabilidades de clasificar correctamente asignando tan solo una de las etiquetas respectivamente. En el caso de la *sensibilidad,* la probabilidad de que un individuo se clasifique en el grupo que presenta la característica y en el caso de *especificidad,* la probabilidad de que un individuo sea clasificado en el grupo que no presenta la característica de interés. La especificidad y la sensibilidad son medidas de evaluación muy importantes en problemas sanitarios. En estos casos es de especial interés conseguir el mayor valor para ambas medidas, lo que se traduce en tratar de detectar el mayor número de verdaderos positivos, así como de no generar muchos falsos positivos. También la relación entre especificidad y sensibilidad es muy importante en cuanto a la generación de alertas en pruebas de diagnóstico sanitario (véase por ejemplo Aramburu La Torre, 2020).

Las curvas ROC representan gráficamente estas dos componentes evaluadas en un punto *c* que representa un umbral a partir del cual se hace la separación de las clases.

$$ROC(c)\begin{cases} x = S(c) \\ y = 1 - E\ (c) \end{cases}$$

En ese punto los valores en x e y son buenas medidas para abordar la evaluación de los modelos. En este y otros trabajos estas representaciones gráficas tienen un gran valor para la dimensión comparativa de distintas implementaciones o soluciones a los problemas de clasificación planteados.

Figura 13: Ejemplo de curva ROC aplicada a un modelo de predicción binaria



En la Figura 13, se presenta un ejemplo de curva para las predicciones aportadas por cierto modelo. Como vemos para los distintos puntos de evaluación (c) se presentan pareados los valores para la sensibilidad y $1 - especificidad$.

El objetivo de un buen modelo de predicción sería hacer máxima la *sensibilidad* en y mínima $1 - especificidad$ en x, lo que hablaría de una gran capacidad predictiva o gran poder de separación entre clases. Un modelo que predijese perfectamente obtendría una curva como la que se muestra en la figura en color rojo. Esa clasificación perfecta dejaría por debajo un área de magnitud 1, la totalidad del cuadrante representado. El área bajo la curva (AUC) representada es la medida resumen asociada a las curvas ROC que ofrece una métrica global de la calidad del modelo. Esta medida de calidad de las predicciones es muy útil desde el punto de vista comparativo. Permite valorar los resultados de diferentes modelos en una misma visualización. No obstante, su lectura puede resultar compleja.

3.3.2.3. Curvas Precision-Recall (curvas PR).

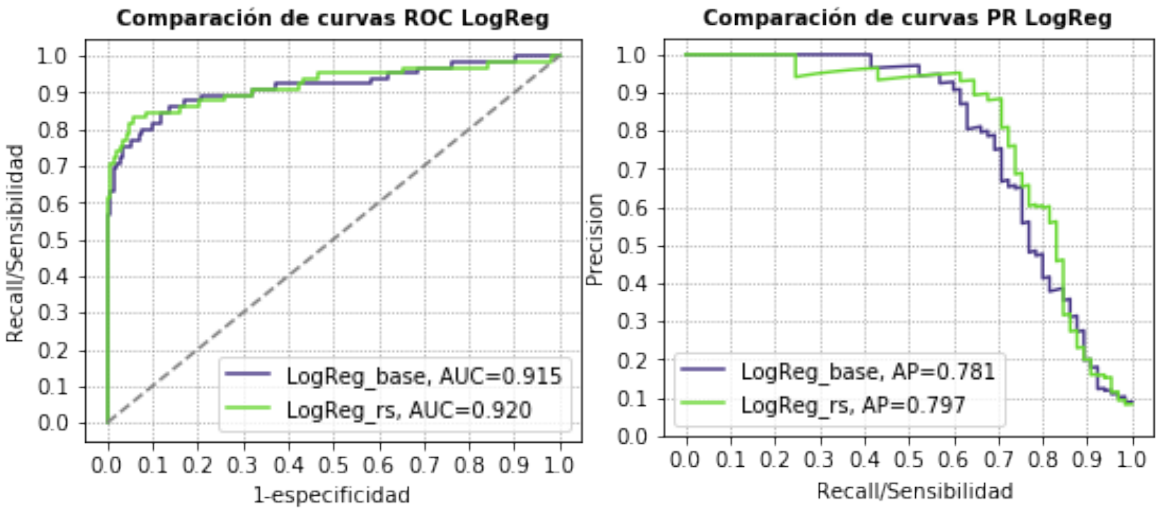
Otro importante recurso en la medición de la calidad de los modelos y que sintetiza visualmente las métricas ya explicadas, son las curvas *Precision-Recall* (PRC). Si bien es una visualización menos habitual en este tipo de trabajos que las curvas ROC, se ha reforzado su uso como alternativa para la evaluación y comparación del desempeño de los modelos. Se ha enfatizado el valor de las curvas *Precision-Recall* a la hora de evaluar los cambios en las predicciones para conjuntos de datos

desbalanceados después de aplicar técnicas de balanceo. En algunos problemas, las curvas *Precision-Recall* permiten visualizar de manera más clara las diferencias en la capacidad predictiva a la clase minoritaria (en la sensibilidad) en relación a la de la clase mayoritaria. Las curvas *Precision-Recall* se representan en un espacio muy similar al de las curvas ROC en el que se formaliza:

$$PR(c)\begin{cases} x = S(c) \\ y = P(c) \end{cases}$$

En un punto c se evalua la sensibilidad (en el eje x) y la precisión (en el eje y). Las curvas PR permiten ver mejor el desplazamiento de las curvas en conjuntos de datos desbalanceados frente a datos balanceados. De forma resumida se puede decir que el uso creciente de las curvas *Precision-Recall* en la evaluación de modelos con datos desbalanceados, se debe a su capacidad para resaltar más diferencias que en las curvas ROC es más difícil apreciar (Boyd, Eng and Page, 2013; Saito and Rehmsmeier, 2015) . En este sentido, un gran cambio en el número de falsos positivos puede tener una gran incidencia en la sensibilidad (entendida como proporción). Estos cambios quedan peor registrados en las curvas ROC y mejor reflejados en las curvas PR al enfrentar sensibilidad y precisión (Davis and Goadrich, 2006). Para problemas como los planteados, en el espacio PR, se están ignorando los verdaderos negativos, centrando la atención en el ajuste de la que en nuestro caso será la categoría de interés.

Figura 14: Ejemplo comparativo con curvas ROC y precisión-recall en datos antes y después de la aplicación de técnicas de balanceo.



En la **Figura 14** , se enfrentan ejemplos de curvas PR y curvas ROC para un modelo de regresión logística en una muestra desbalanceada. El modelo se aplica a los datos sin balancear (línea azul) y a los datos balanceados (línea verde). Mientras que en las curvas ROC resultan observables pequeñas variaciones de la sensibilidad, los cambios en la precisión (recordamos, importante medida de la calidad de nuestra

predicción en la categoría que presenta la característica de interés) son difícilmente interpretables. Las curvas PR matizan significativamente los cambios vinculados a la aplicación de técnicas de balanceo. Se aprecia la importancia de las ganancias (o pérdidas) en la precisión y la sensibilidad y permite ubicar espacialmente las regiones (puntos o umbrales) en las que se hacen realmente importantes las diferencias. Para este trabajo y por su importancia formativa, se plantea la complementariedad de ambas representaciones a la hora de evaluar la aplicación de distintos modelos y distintas técnicas de balanceo.

Como métricas de resumen en el caso particular de las curvas PR al igual que en el caso de las curvas ROC es posible el cálculo del *área bajo la curva*. Usualmente en el trabajo con curvas PR se calcula la *precisión media* como métrica de resumen alternativa que relaciona la precisión y la sensibilidad/recall de los modelos. Los resultados de estos dos indicadores pueden diferir ligeramente. Con la misma lógica para el cálculo de un área por el método de los trapecios, se parte de T particiones calculando el área de la forma en que se indica abajo en cada punto t siendo S y P la sensibilidad y precisión en cada punto t.

Precisión media: $$AP = \sum_{t=1}^{T} (S_t - S_{t-1}) \cdot P_t$$

Como se observa, la precisión media pondera las diferencias de la sensibilidad entre dos puntos *(t-1, t)* consecutivos con la precisión en el punto final (en este caso t). La precisión media es una métrica de resumen de los modelos, que al igual que sucede con las representaciones de las curvas, permitirá matizar el valor del área bajo la curva ROC vinculadas a la calidad de las predicciones que se hacen a la categoría de interés. Lo realmente importante es que una curva el mejor valor del AUC no tiene por qué coincidir con el mejor modelo del espacio PR (Davis and Goadrich, 2006). Al igual que explicábamos en las visualizaciones, este tipo de discrepancias permiten matizar y describir el rendimiento de las técnicas y modelos a utilizar.

3.3.2.4. Pérdida logística (log-loss) como medida alternativa.

Para rematar este punto conviene destacar que existen otras métricas de resumen que pueden resultar más adecuadas cuando lo que se pretende es comparar el desempeño de distintos modelos. Las más habituales son las relacionadas con las funciones de pérdida en la clasificación. Se han apuntado en la literatura discrepancias entre las métricas derivadas de este conjunto de funciones y el AUC, corroborando una mejor fiabilidad de las primeras (Figini and Maggi, 2014). La importancia de estas métricas alternativas radica en que permiten la valoración de la pérdida en las predicciones probabilísticas. Es decir, en estos casos el rendimiento no queda vinculado en origen a la contabilización de casos incorrecta o

correctamente clasificados. Lo que permiten estas funciones de pérdida es ponderar la desviación de las predicciones probabilísticas respecto a la etiqueta o clase real observada. Este tipo de evaluación ofrece una medida más precisa del rendimiento de los modelos de predicción.

De entre este grupo de funciones, en los problemas de clasificación binaria, es habitual el uso de la *perdida logística* o *pérdida de entropía cruzada.* Como se apuntaba, partiendo de las predicciones probabilísticas aportadas por los modelos se es capaz de medir la incertidumbre de nuestro clasificador de la siguiente forma:

$$L_{\log}(y,p) = \sum_{(y,p)\in D} -\log \Pr(y|p) = \sum_{(y,p)\in D} -(y \cdot \log(p) + (1-y) \cdot \log(1-p))$$

Donde $y \in \{0,1\}$ son las etiquetas de clase reales u observadas y p es el valor de la predicción probabilística aportada por el clasificador entendida como $p = \Pr(y = 1)$. D representaría el conjunto de datos sobre el que calculamos las pérdidas en los datos pareados para cada caso de *(y, p).*

4. Aplicación a los datos.

En este capítulo se desarrolla la parte empírica del trabajo. Las siguientes páginas estarán centradas en la aplicación de las técnicas explicadas y la medición del rendimiento de los clasificadores combinados con las distintas técnicas de *resampling*. Esta parte del trabajo se divide en tres secciones. La primera se centra en la presentación del conjunto de datos utilizado, exponiendo la propia selección de datos, las operaciones de limpieza y preparación, además de la organización de estas tareas y la relación de variables final del conjunto. La segunda parte de este capítulo explica en detalle las librerías y técnicas utilizadas para completar el trabajo. Por último y como sección más relevante, se presentan los resultados finales de la clasificación en combinación con las distintas técnicas de *resampling*, en la que se exponen todas las métricas de rendimiento ya explicadas.

4.1. Presentación y selección de los datos.

Uno de los puntos importantes que ha guiado la propia evolución y ejecución de este trabajo ha sido la selección de un conjunto de datos que propusiese los retos que se derivan de las temáticas tratadas. Desde un principio y dado el contexto en el que se desarrolla este proyecto se ha buscado trabajar con datos originales, en bruto y que permitan el desarrollo de un ejercicio real. La selección final considerada más relevante y la que satisface las premisas anteriores es la proporcionada por la plataforma asiática de música en *streaming* KKBox. Estos datos se ofrecen en

abierto[3] en la plataforma *Kaggle* vinculados al concurso organizado en el marco de la "*11th ACM International Conference on Web Search and Data Mining (WSDM 2018)*". Los datos ofrecidos son datos reales de una plataforma de música en *streaming* que se facilitan para estudiar y predecir la fuga de clientes de los servicios ofrecidos por esta plataforma.

Se decide trabajar con estos datos por las siguientes razones:

- Permiten acercarse a la definición de un problema real.
- Presentan complejidad en su estructura, tanto para la creación de las variables finales como para la limpieza y depuración de los datos. Desde un punto de vista formativo permite:
  a. Trabajar con un alto volumen de datos.
  b. Libertad en la creación y construcción de las variables finales. Aquí se plantean importantes retos en cuanto a la definición de algunas variables agregadas.
  c. Acercarse al problema del desbalanceo de datos y la predicción de fuga de clientes a través de datos de uso real.

4.1.1. Evolución del trabajo con los datos.

Antes de explicar en profundidad el trabajo de limpieza y preparación de datos, así como la parte empírica, es conveniente enmarcar la evolución que ha seguido el trabajo en el contexto de las prácticas curriculares en el que se ha realizado.

Partiendo del estudio de la fuga de clientes como tema principal de las prácticas, las primeras etapas se centraron en el estudio de las distintas aproximaciones posibles. Algunas ya han sido explicadas en el epígrafe anterior y fueron consideradas como posibles opciones centrales de este estudio. Una de las ideas iniciales propuestas para este trabajo consistió en la posibilidad de comparar modelos de aprendizaje automático y análisis de supervivencia para la predicción de la fuga de clientes. Esta idea fue posteriormente descartada por dos razones. En primer lugar, el tamaño del trabajo crecía excesivamente. En segundo lugar, la selección que se ha hecho de los datos podía implicar el desarrollo de dos procesos de limpieza y preparación de datos separados para los dos conjuntos de técnicas. Las particularidades de los datos usados en este trabajo, y que se verán más adelante, generaban conflictos en cuanto a cómo considerar algunos de los clientes si se usaba el análisis de supervivencia para estudiar la fuga de clientes. No obstante, partes importantes del proceso de preparación de datos han tenido como objetivo dar sentido a los historiales de suscripciones de los clientes de la plataforma KKBox, con el objetivo de que la aplicación del análisis de supervivencia pudiese llevarse a cabo. Parte de ese trabajo,

[3] Los datos están disponibles en: https://www.kaggle.com/c/kkbox-churn-prediction-challenge

se ve reflejado en este texto, a través de la creación de variables como los tiempos de vida de los clientes o la detección de discontinuidades en el servicio.

En cualquier caso, todo el proceso de trabajo de preparación de los datos ha sido una parte central en el desarrollo de las prácticas curriculares en las que se ha realizado este trabajo. El conjunto de datos escogido plantea un reto importante en cuanto al gran tamaño de la muestra y en cuanto a las limitaciones de la información original. En este sentido, ha de tenerse en cuenta que la información no es excesivamente rica, contando tan solo con un resumen del tipo de uso del servicio durante el último mes e información relativa al historial de suscripciones de los clientes, que necesita de un importante trabajo interpretativo. Por otro lado, la información sociodemográfica de los clientes es casi inexistente ya que, o está ausente o la que está declarada no es válida (como son los casos de sexo y edad respectivamente). Otra limitación importante es la escasa información proporcionada sobre las distintas variables, acciones como cancelaciones o autorrenovaciones no son explicadas por lo que no es posible interpretar la información. En otros casos, asimetrías entre los precios de las suscripciones y de los realmente pagados, conflictos de fechas de inicio y final de las suscripciones o duplicidades y solapes no son explicados. Por todas estas razones el trabajo con estos datos resulta complejo y dificulta un análisis exploratorio al uso por no poder avanzar en las explicaciones. No obstante, como se puede apreciar en los anexos hay un importante trabajo para adecuar los datos a una estructura lógica y generar nuevas variables que permitan mejores resultados en la fuga de clientes. Como veremos más adelante, ese trabajo de preparación se justifica gracias a los buenos resultados de los que parten los modelos en las tareas de clasificación.

El flujo de trabajo con los datos y todas las operaciones que se explicarán a continuación fueron desarrolladas en el lenguaje de programación Python en su versión 3. El aprendizaje de este lenguaje forma parte del proceso formativo de las prácticas y se presenta como un hito más a superar. En los anexos 2 y 3, se recoge todo el desarrollo de código de todas las transformaciones realizadas a los datos y en el anexo 4 se detalla el código para la extracción de resultados. Como se observa, todo el proceso se agrupa en 3 bloques de código. En el primero (*pipeline 1*), se recogen las operaciones necesarias para la obtención de una versión completa del conjunto de datos con el que se va a trabajar. En el segundo bloque (*pipeline 2*) se desarrollan otro tipo de funciones que abordan la preparación del conjunto de datos para la aplicación de los modelos. Las operaciones aquí recogidas pueden considerarse un protocolo que se ajusta a las metodologías de trabajo del centro de prácticas y cuestiones relacionadas con el soporte informático. El tercer y último bloque, recoge el código que se encarga de construir los modelos, validarlos y extraer las métricas de rendimiento. En este sentido, las figuras y tablas incluidas en la presentación de resultados y las tablas de métricas para la evaluación de los

modelos se generan de forma automática a través de ese bloque de código. La idea general de todo el código es la de simular un entorno en el que la adquisición de nuevos datos o actualizaciones periódicas de los mismos, puedan generar en esas tres etapas los mismos resultados. La creación del conjunto de datos completo primero, la preparación para la construcción de modelos en segundo lugar y la extracción de resultados.

A partir de este punto se presentará todo el proceso de preparación de datos detallando las distintas operaciones y fases que componen el proceso. Se mostrarán también algunas cuestiones claves a la hora de generar nueva información necesaria para la construcción de los modelos.

4.1.2. Presentación de archivos y variables en crudo.

Los datos proporcionados se organizan en 4 ficheros con distinta información en cada uno de ellos. Uno de los ficheros tan sólo ofrece la información vinculada a la variable dependiente; si el cliente se ha fugado o no se ha fugado. En el resto de ficheros se ofrecen los datos sobre los que se deben construir las distintas variables que se incluirán en el estudio. La estructura de archivos es la siguiente.

- *train*: Recoge tan solo dos variables y tiene una dimensión de 970.960 filas × 2 columnas:

- *msno*: Id interna del cliente
- *is_churn:* Indica si el cliente se ha fugado = 1 o si no se ha fugado = 0

Conceptualmente es muy importante matizar que la fuga del cliente no queda circunscrita a la cancelación directa del servicio por parte del usuario (componente activa) sino que la fuga se define como la ausencia de renovación o contratación de una nueva suscripción durante los 30 días siguientes a la caducidad de la última suscripción activa o la propia cancelación de servicios.

Con esta información los datos que tenemos en este fichero son los de clientes que habrían de renovar sus suscripciones durante el mes de marzo de 2017. Los clientes fugados son aquellos que no renovaron en ese periodo sus suscripciones y los no fugados los que sí lo hicieron bajo la definición de fuga descrita. Esta concepción de fuga tiene importantes implicaciones en los cálculos posteriores para algunas de las variables finales.

- *user_logs:* En este conjunto de datos se ofrece información relativa al uso del servicio contratado, entre el 01/03/2017 y el 31/03/2017, último mes de servicio (dimensión 18.396.362 × 9). Cada registro representa la actividad de un usuario en un día concreto del servicio. Por tanto, no se obtienen registros unívocos por usuario, lo que tenemos es un registro de uso en donde cada fila representa una

conexión en un día concreto al servicio. De esta manera tenemos un registro de cada uno de los días que un usuario usó el servicio y qué tipo de uso hizo en relación a las siguientes variables:

- *msno*: Identificador del usuario. Valores unívocos que identifican a cada cliente.
- *date*: Fecha del registro de uso.
- *num_25*: Número de canciones reproducidas por debajo del 25% de su duración total.
- *num_50*: Número de canciones reproducidas entre el 25 y el 50% de su duración total.
- *num_75*: Número de canciones reproducidas entre el 50 y el 75% de su duración total.
- *num_985*: Número de canciones reproducidas entre el 75 y el 98,5% de su duración total.
- *num_100*: Número de canciones reproducidas en su duración total.
- *num_unq*: Número de canciones distintas reproducidas (no repetidas).
- *total_secs*: Segundos totales de reproducción.

Como vemos se ofrece una información relativamente detallada en cuanto a la intensidad y tipo de uso, sin entrar a valorar elementos categóricos como el soporte de la música escuchada (listas, álbumes, radio), artistas, géneros, etc. Como veremos lo más importante será la obtención de los valores resumen agregados para todos los usuarios ya que, como avanzamos, los identificadores de usuario son repetidos (un usuario puede hacer uso muchos días a lo largo del mes de referencia).

- ***transactions:*** Dimensión de 22.978.755 filas × 10 columnas. Este archivo incluye la información más complicada de tratar y de entender y al mismo tiempo parte de la información más relevante en el estudio de la fuga de clientes. Por ejemplo, la posibilidad de evaluar los tiempos de permanencia en el servicio y los historiales de permanencia de los usuarios. Este archivo contiene todas las transacciones (o suscripciones) que un usuario efectúa a lo largo de su vida como cliente del servicio. Existen distintos planes a contratar y pueden ser planes que se autorrenuevan o no, pueden existir cambios de plan, cancelaciones de esos planes, distintos precios de contratación, etc. De nuevo las filas no son unívocas, cada fila representa una suscripción y durante el tiempo como clientes un usuario puede realizar múltiples suscripciones.

- *msno*: Identificador del usuario.
- *payment_method_id*: Método de pago usado.
- *payment_plan_days*: Número de días del plan contratado.
- *plan_list_price*: Precio del plan contratado.

- *actual_amount_paid*: Precio pagado (se entiende que puede diferir del anterior por descuentos, promociones, etc.)
- *is_auto_renew*: Informa si la transacción es una autorrenovación (no se ofrece más información al respecto en cuanto al funcionamiento de la autorrenovación).
- *transaction_date*: Es la fecha en la que tiene lugar la transacción
- *membership_expire_date*: Fecha de caducidad de la suscripción.
- *is_cancel*: Si el usuario cancela o no su suscripción durante esa transacción.

Como veremos lo más importante aquí es determinar qué tipo de relación establecer entre las fechas y las duraciones de los servicios para los cálculos de los tiempos de suscripción. También se desprende importante información respecto a cantidades pagadas, descuentos obtenidos o fugas anteriores de los clientes incluidos en el estudio.

- *members:* Incluye un mínimo de información sociodemográfica afectada por valores perdidos y atípicos, pero unívoca y referida a cada usuario (dimensión 6769473×6). En este caso cada fila representa un único usuario.

- *msno*: Identificador del usuario.
- *city*: Ciudad.
- *bd*: Edad.
- *gender*: Sexo.
- *registered_via*: Método de suscripción.
- *registration_init_time*: Fecha en la que se realizó el primer registro.

Se aprecia en esta presentación de los datos una cierta complejidad en las operaciones necesarias para la extracción de información resumen consistente de cara al estudio pretendido. El volumen de datos es considerable y algunas de las operaciones de agregación para obtener indicadores para cada usuario son complejas. En el siguiente punto se presentan las soluciones concretas por las que se ha optado para la obtención de un conjunto de datos final con el que poder trabajar.

4.1.3. Construcción del conjunto de datos KKbox.

La construcción del conjunto de datos final parte de la definición de usuarios que entran a formar parte. Este primer paso viene definido por la información que tenemos respecto a la variable respuesta *is_churn* (usuario fugado/usuario no fugado) definida en el conjunto de datos *train.* Los 970.960 usuarios sobre los que poseemos esta información son los usuarios que formarán la base de datos. A partir de ahí, las agregaciones que se realicen del resto de información contenida en los otros los ficheros serán añadidas para esos usuarios. Como se apuntaba, el resto de

información ha de ser tratada para conseguir medidas resumen para cada uno de los usuarios. Tanto los registros de uso del servicio, como la información de las transacciones, requieren un procesamiento más o menos complejo para extraer medidas unívocas (una por cada usuario).

a. Operaciones en user_logs

Como ya se avanzaba este archivo contiene información diaria de uso del servicio de música en *streaming* KKbox. Cada fila de las 18.396.362 representa el uso en un día por parte de un usuario concreto durante el último mes del servicio. Las variables incluidas contabilizan el número de canciones reproducidas en distintos porcentajes de su longitud y el número de segundos de reproducción total por día. En este caso, la solución adoptada para la obtención de indicadores resumen resulta bastante sencilla. La opción por la que se ha optado es agregar las variables como la suma del número de canciones reproducidas en cada tramo de porcentaje y la suma de segundos totales de reproducción por usuario en el mes de referencia. Además, se incluye una nueva variable que sería la contabilización del número de días que los usuarios hicieron uso del servicio.

Con lo explicado se obtendría un nuevo conjunto de datos agregado que ofrecería esta información:

msno	Identificador de cliente
num_25	Suma del número de canciones escuchadas a menos del 25% de su longitud en el último mes
num_50	número de canciones escuchadas entre el 25% y 50% de su longitud en el último mes
num_75	Suma del número de canciones escuchadas entre el 50% y 75% de su longitud en el último mes
num_985	Suma del número de canciones escuchadas entre el 75% y 98.5% de su longitud en el último mes
num_100	Suma del número de canciones escuchadas entre el 98.5% y 100% de su longitud en el último mes
num_unq	Suma del número de canciones distintas escuchadas en el último mes
total_secs	Suma de los segundos de reproducción en el último mes
n_activity	Cuenta del número de días en los que se registra actividad en el último mes

Se completan estas agregaciones para conseguir medidas resumen que puedan unirse a la base original a través de la identificación de los clientes de forma unívoca.

b. Operaciones en transactions.csv

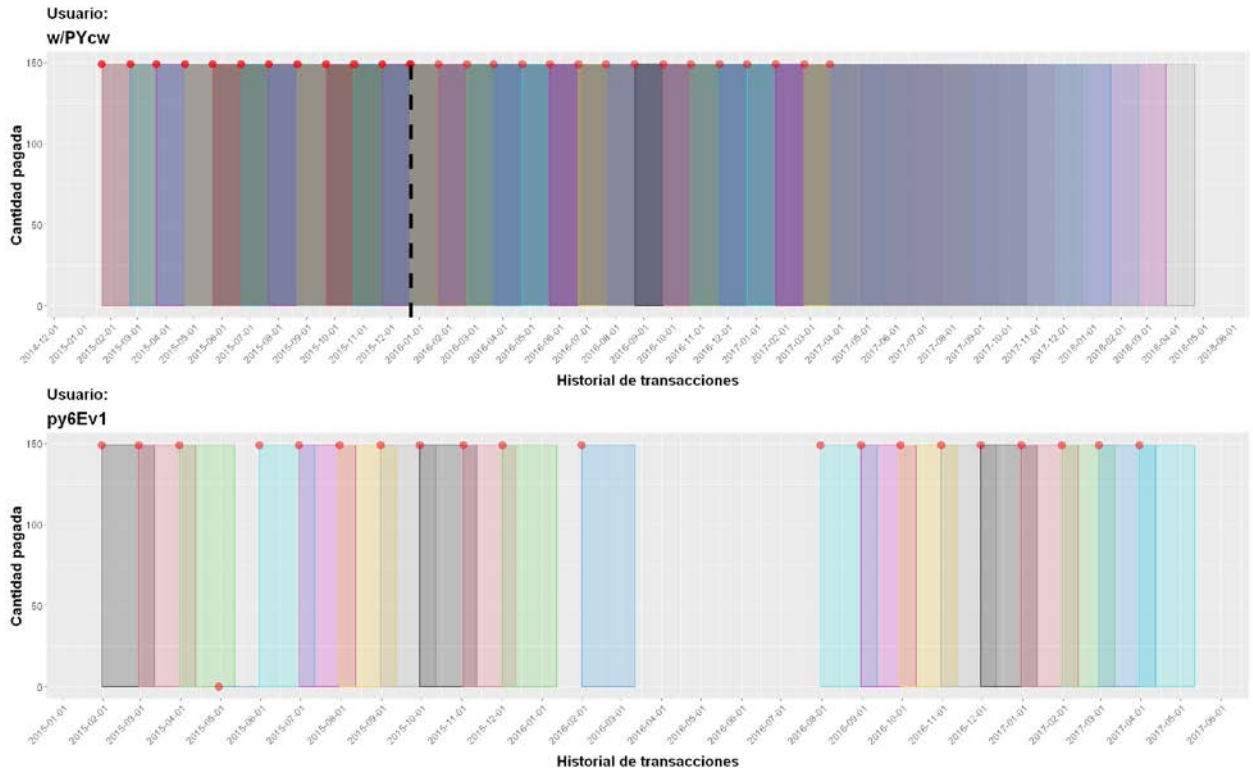
Las operaciones en este conjunto de datos son las más complejas. La información relativa a transacciones puede ser en muchos casos compleja. Lo que buscan las transformaciones realizadas es fundamentalmente:

- La correcta ordenación de las fechas de inicio y fin de las transacciones.
- Eliminación de transacciones duplicadas o sin información útil.

- Extraer otra información deducible del estudio de los historiales de suscripción.

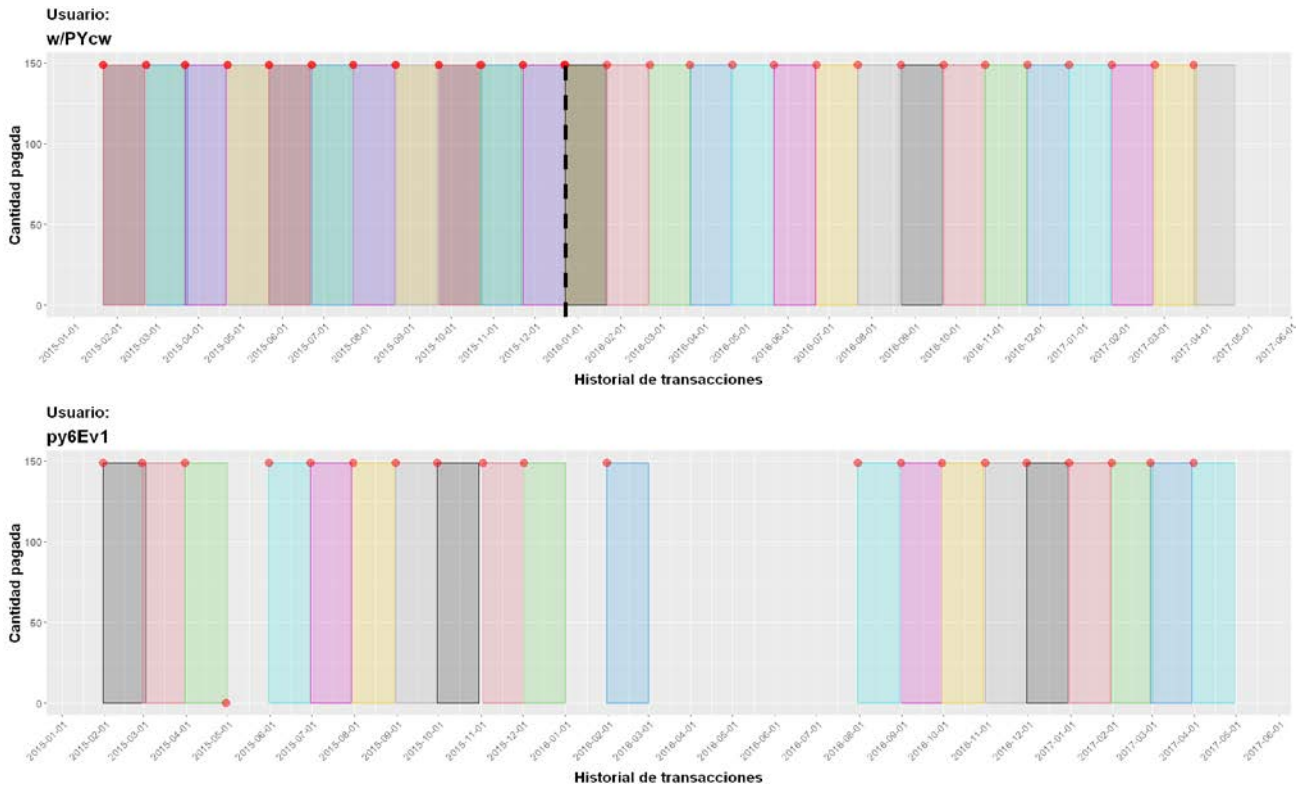
La información más confusa en relación a las distintas transacciones para algunos usuarios es la ordenación de las fechas de caducidad de las transacciones. En muchos casos se muestran fechas de caducidad más tardías a las que corresponderían realmente. Igualmente, en usuarios con múltiples transacciones se producen solapes o extensiones de las suscripciones que carecen de sentido en base al resto de información disponible. Es posible ilustrar los problemas y la depuración que necesitan los datos incluidos en este archivo gráficamente. En la **Figura 15** es posible observar dos ejemplos del tipo de problemas asociados a los historiales de suscripción de algunos usuarios. Cada uno de los rectángulos representa una suscripción de ese usuario siendo la base del mismo el tiempo de duración de la suscripción, y la altura, el valor monetario de las suscripciones. La línea negra discontinua corresponde a un momento en el que una suscripción es cancelada por el usuario. Como vemos, y esto es lo más frecuente, el valor de las suscripciones o planes contratados es siempre el mismo y la renovación de las mismas se lleva a cabo de forma automática (una autorrenovación se representa por un punto rojo en la parte alta). Los problemas surgen, sólo entre algunos usuarios, cuando las duraciones de los planes contratados contados a partir de las fechas de inicio de la suscripción (*transaction_date*) hasta la fecha de caducidad (*membership_ expire_date*), difieren de las duraciones reales en función del número de días contratados en el plan. Se desconoce el origen de estas deficiencias en la recogida de datos, pero genera problemas en relación al cálculo de casi cualquier variable vinculada a las transacciones de los usuarios. Es decir, afectaría a montantes totales pagados, a números totales de suscripción, a cambios de plan o a discontinuidades en la contratación de servicios. Es por ello, que es obligada la corrección de las fechas de caducidad de las suscripciones y generar las fechas reales de vencimiento en función del número de días contratados en el plan o suscripción.

Figura 15: Ejemplos de historiales de suscripción de dos usuarios con problemas en los registros de fechas.



Las correcciones en las fechas permiten limpiar y obtener una secuencia lógica de las suscripciones de los usuarios. En este sentido, ofrecer coherencia a los historiales de los usuarios permite observar nuevos fenómenos a partir de los cuales poder construir nuevas variables. En la **Figura 16**, es posible ilustrar algunas de las peculiaridades del estudio de estos datos. Por ejemplo, el primer usuario mostrado, realiza una cancelación de servicios en un momento dado (línea negra discontinua) lo que no significó una fuga de ese cliente en ese momento, porque como se puede observar siguió contratando y disfrutando los servicios. Por el contrario, el segundo usuario mostrado presenta una discontinuidad en el servicio sin haber mediado una cancelación. En este segundo caso estaríamos ante un cliente que presenta una fuga en el pasado bajo la definición de fuga establecida por el propio servicio (30 días sin renovación o contratación de una nueva suscripción).

Figura 16: Mismos ejemplos de historiales de suscripción anteriores con corrección de los problemas en los registros de fechas.



Este tipo de información, que es posible extraer de los historiales de suscripción, puede ser relevante en cuanto al comportamiento de los clientes. No está directamente relacionada con el uso del servicio, es información que describe comportamientos distintos relacionados con la contratación de los servicios. Como veremos a continuación disfrutar del servicio de forma intermitente, cambiar de planes de suscripción con más frecuencia y cuestiones similares, pueden desvelar mayor o menor tendencia a que el cliente se fugue. Realizada la limpieza y la corrección de los historiales es posible el cálculo de información final relacionada con las transacciones como pueden ser la cantidad de tiempo suscrito a servicios, las interrupciones del servicio en el pasado, la duración de la suscripción actual, tiempo suscrito en otras suscripciones pasadas, etc.

Cuestiones que se han considerado importantes de cara a la predicción de la fuga de clientes y que se derivan del trabajo con las fechas, duraciones y valores de las suscripciones son:

- *Número de fugas en el pasado:*

Este trabajo se centra en el estudio de la fuga de clientes y estamos en disposición de detectar fugas pasadas en los historiales de los usuarios. Se ha considerado

importante la detección de los intervalos de tiempo en lo que existieron interrupciones en la prestación de los servicios ya sea por cancelación directa o por ausencia de renovación/contratación. En este sentido, se ha considerado que comportamientos en cuanto a la contratación intermitente puede diferir entre los usuarios que finalmente se fugan y los que no.

Tabla 6: Número medio de fugas en el pasado en los historiales de suscripción

	Número medio de fugas en pasado
No fugado	0,42
Fugado	0,58

Atendiendo a la **Tabla 6** se aprecia una tendencia mayor a la existencia de fugas pasadas en los historiales de suscripción de los usuarios que se fugan en el estudio. Esto desvela la posibilidad de un comportamiento más activo en cuanto a una contratación intermitente de los servicios, es decir, la posibilidad de que los clientes fugados en el pasado tengan una mayor propensión a la fuga.

- *Cálculos de tiempo de los periodos de suscripción:*

Otro output de cierta centralidad en los estudios de fuga de clientes, como se ha apuntado al principio de este epígrafe, es el tiempo de vida del cliente muy importante por ejemplo en el análisis de supervivencia. Dadas las peculiaridades de los datos y del servicio estudiado, se ha considerado importante conocer el tiempo total e ininterrumpido que un cliente ha disfrutado de los servicios. En el caso de clientes fugados en el pasado, estos tiempos de vida de los clientes serán los relativos al último bloque de suscripciones activo antes de la fecha fin del estudio (marzo de 2017). Además, se calcula el número de días totales de suscripción a servicios de la plataforma, hayan o no mediado fugas en el pasado.

Tabla 7: Duración (días) promedio del último bloque de suscripciones activo.

	Tiempo medio de vida
No fugado	427,89
Fugado	363,46

En la **Tabla 7** se muestran las diferencias de la duración de las suscripciones medias entre los clientes que se fugan y los que no. Podría interpretarse que los clientes que no se fugan tienden a ser los clientes más antiguos. En relación a lo visto antes, entre los clientes fugados, las fugas puede que se asocien a clientes más dinámicos o activos en cuanto a la posibilidad de cancelar sus servicios.

Tabla 8: Días promedio disfrutados en suscripciones anteriores al bloque activo.

	Número de días suscrito en el pasado
No fugado	59,03
Fugado	68,50

La **Tabla 8** matiza lo expuesto, constatando que los clientes que se fugan han disfrutado en promedio un mayor número de días vinculadas a bloques de suscripción pasada. Es decir, los clientes fugados pueden presentar un comportamiento más propenso a interrumpir y reanudar de forma alternativa la suscripción a los servicios en mayor grado que los clientes no fugados. Se obtienen así dos medidas de interés que se complementan. Si bien el número de días totales de suscripción en los historiales de los clientes es muy similar, entre los clientes fugados parece existir una mayor tendencia a que esos días se disfruten en una forma más intermitente a lo largo del tiempo.

Además, tomando estas dos medidas de forma separada conseguimos no considerar de igual manera a un cliente que ha contratado una suscripción en una fecha cercana al fin del estudio, con otro que hace lo mismo pero que ha tenido relación comercial con el servicio de forma prolongada en el pasado.

- *Número de cambios de plan:*

En el estudio de los historiales resulta también relevante conocer la cantidad de veces que se cambia de plan de suscripción. Si bien lo habitual es la renovación de un mismo tipo de plan, el cambio o alternancia del tipo de plan contratado se desvela como una medida interesante para el estudio entre clases.

Tabla 9: Número medio de cambios de plan en los historiales de los usuarios.

	Número medio de cambios de plan
No fugado	0,46
Fugado	0,90

La **Tabla 9** desvela la importante diferencia de cantidad media de veces que los usuarios cambian de plan en función de si son clientes fugados o no. Entre los fugados la tendencia a cambiarse de plan bastante más alta que entre los clientes fugados. Nuevamente se desvela un perfil, entre los clientes fugados, más activo en lo que a contratación de servicios se refiere.

Tabla 10: Proporciones de usuarios en función del plan más comúnmente contratado.

	Otro	***Plan 30 días***
No fugado	0,29%	99,71%
Fugado	21,00%	78,99%

Tendencias que confirman lo dicho puede ser constatadas en la **Tabla 10** en relación al tipo de plan comúnmente contratado. Entre los no fugados el plan de 30 días es prácticamente el único plan contratado y además vemos que es menos habitual el cambio de un tipo de plan a otro. Entre los fugados el plan de 30 días es también el tipo de plan más usual, pero se muestra claramente que hay un mayor peso de otros planes en este grupo de clientes. Se vuelve, por tanto, a desvelar comportamientos más cambiantes dentro del grupo de los fugados. El grupo de los no fugados parece

mostrar, en relación a la contratación de servicios, un comportamiento más estable o poco cambiante.

- *Cálculo de los descuentos disfrutados:*

Por último, otra cuestión que resulta relativamente fácil de extraer, es la detección de descuentos. Se había apuntado la falta de información acerca de discordancias entre los precios de los planes y las cantidades pagadas en algunas suscripciones. Se puede entender que estas diferencias resultan de descuentos o promociones que los clientes pueden disfrutar. A pesar de no tener información concreta, es posible esperar diferencias entre los grupos de la variable dependiente en relación a lo que llamamos descuentos disfrutados.

Tabla 11: Descuento promedio disfrutado entre los grupos de la variable dependiente.

	Descuento medio disfrutado
No fugado	15,67
Fugado	19,07

En la **Tabla 11** se aprecia una mayor cantidad promedio de descuentos disfrutados entre los clientes fugados. Relacionado con lo ya visto en otras variables creadas, podría entenderse que los clientes fugados se benefician en mayor medida que los no fugados de descuentos a causa de un comportamiento más activo o menos estático en relación a la contratación de servicios.

El tratamiento de toda esta información tiene como garantía final la creación de variables para cada usuario que permitan la predicción de la fuga de clientes. La finalidad de todas las transformaciones propuestas es extraer la mayor información posible de los datos sin la creación excesiva de nuevas variables[4]. Lo que se pretende asegurar es un punto de partida óptimo antes de la aplicación de las técnicas de *resampling*. Es decir, que a partir del conjunto de datos final se esté en disposición de alcanzar unos resultados aceptables en las tareas de clasificación sin modificar la muestra.

c. Transformaciones para la creación del conjunto de datos final.

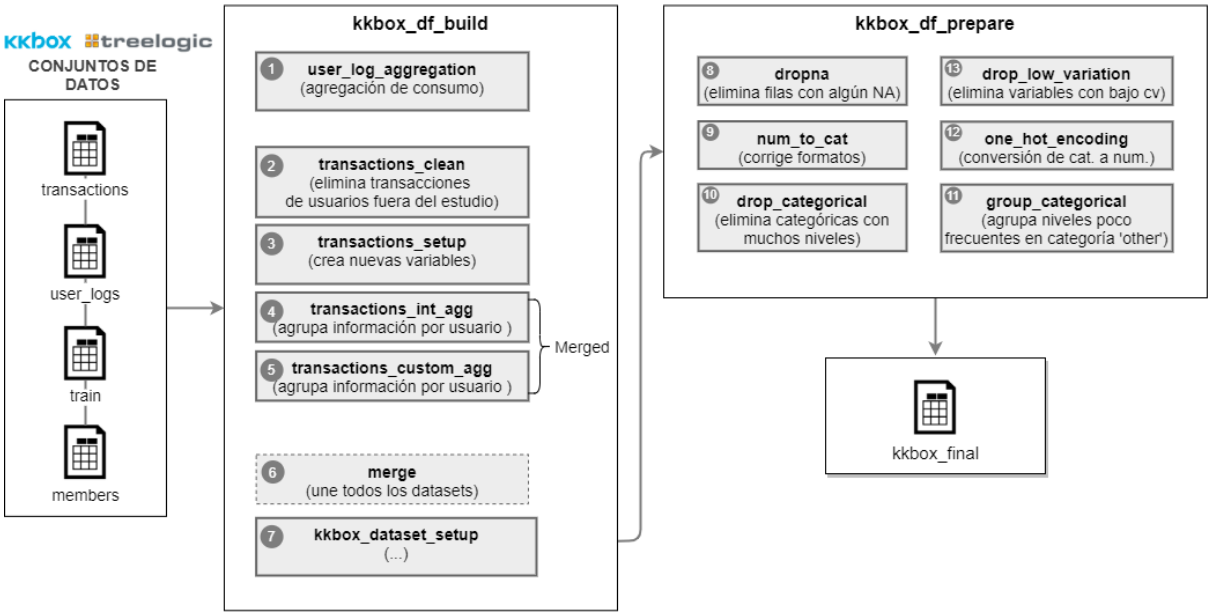
La construcción del conjunto de datos final depende, por tanto, de la ejecución de todas las tareas descritas en los dos puntos anteriores y de la unión de los distintos ficheros de datos en un solo conjunto unívoco (un usuario una fila). Una vez que los

[4] Como ejemplo el conjunto de datos "user_logs" representaban los usos de la plataforma por parte de los usuarios en el último mes de suscripción. Habría sido posible analizar un espacio temporal mayor en relación con estos usos para analizar evoluciones y cambios en los patrones de comportamiento. Dado el gran volumen de datos con el que se trabaja, este tipo de soluciones habrían implicado la creación de un gran número de variables, el aumento de la carga computacional tanto para las tareas de resampling como para las de clasificación y el cambio de las tecnologías empleadas.

distintos archivos pueden fusionarse, comienza otra fase de acondicionamiento de datos para poder aplicar las técnicas y algoritmos que se proponen.

En la **Figura 17** se presentan sintéticamente las tareas de limpieza y preparación de los datos. Las acciones de 1 a 5 de la caja *kkbox_df_build* corresponderían a las transformaciones en los datos ya explicadas. En la casilla número 6 se formalizaría la unión de los archivos transformados de *user_logs* y *transactions* además de la información contenida en *members* que no requiere ninguna transformación. De este último archivo tan sólo se extrae la información de aquellos usuarios incluidos en el archivo base (*train*) con la información relativa a la fuga de clientes. En ese preciso momento se obtiene una primera versión de conjunto de datos que será definitivo, previa preparación y adecuación a una estructura consistente para la aplicación de modelos.

Figura 17: Esquema del proceso de limpieza y construcción del conjunto de datos.



En la caja 7 se llevaría a cabo una serie de transformaciones necesarias para la corrección de fechas y el cálculo de los tiempos de registro, de permanencia en el servicio y de permanencia desde el último periodo de suscripciones activo. La finalidad principal de estas trasformaciones es eliminar las variables con formato fecha, inservibles para el trabajo con modelos de clasificación. Se llevan además a cabo otras operaciones necesarias para la coherencia de los datos. Concretamente las transformaciones llevadas a cabo en ese punto son:

- Formato a las variables tipo fecha (día/mes/año)
- Cálculo de la fecha de inicio del periodo de transacciones activo
- Corregir campos de fechas de caducidad falseados por suscripciones gratuitas.

- Cálculos de tiempos para las fechas de antigüedad[5]:
  - Número de días totales desde el registro en el servicio (creación de una cuenta).
  - Número de días totales suscrito en todas las suscripciones del historial.
  - Número de días de duración del último periodo de suscripciones activo.
- Corrección de casos ausentes en las variables de uso (*user_logs*). En este caso se imputan los valores perdidos con valor 0. Un usuario que no aparece registrado en los datos de uso del último mes, es un usuario que hace un uso igual a 0 y no un caso perdido.
- Se eliminan las variables que tengan más de un 30% de casos perdidos (tan solo afecta a la variable sexo procedente de *members*). Al querer trabajar con casos completos no se deseaba una reducción drástica del número de casos.

Para adecuar el conjunto de datos a su versión definitiva para la aplicación de modelos se ejecuta el conjunto de funciones de *kkbox_df_prepare* (véase **Figura 17**). Este conjunto de operaciones pretende optimizar los datos para el trabajo con las distintas técnicas usadas en este trabajo. Este conjunto de funciones, excepto la selección de casos completos, corresponde a protocolos estandarizados en el trabajo con datos dentro de la empresa en la que se cursaron las prácticas. Las operaciones se realizan fundamentalmente por el soporte informático usado y optimizar los datos para la construcción de los modelos con los paquetes y librerías utilizados. Las acciones son:

- La selección de **casos completos**. Ante el gran volumen de datos no se realiza imputación de datos perdidos. Se opta por conservar en el estudio sólo los individuos que tienen información para todas las variables. Se asume con esta decisión la pérdida de aproximadamente 100.000 individuos pasando de 970.960 casos a 860.966 (un 11,3 % del total de la muestra).
- Se ha previsto también la eliminación de variables categóricas con un alto número de categorías[6]. En este conjunto de datos no afecta este control por tanto se respetan todas las variables, pero a pesar de no afectar, se ha estimado necesaria la **agrupación de categorías con una baja frecuencia** en algunas variables. Concretamente, se calculan las frecuencias ordenadas de todas las categorías de respuesta y se agrupan aquellas que estén por encima del 85% (umbral elegido para este trabajo). Esta codificación afecta de la siguiente manera:

[5] Estas fechas pretendían ser la base del análisis de supervivencia en el caso de haberse realizado.
[6] El protocolo es eliminar variables con un número de categorías que excedan una proporción del tamaño del conjunto de datos. Este tipo de controles tienen una aplicación más clara en conjuntos de datos pequeños razón por la que en este caso no se eliminó ninguna variable por este método.

- Se agrupan 27 niveles para la variable *last_plan_paid.* Resultan 2 categorías.
- Se agrupan 28 niveles para la variable *last_payment_method.* Resultan 7 categorías.
- Se agrupan 31 niveles para la variable *usual_plan.* Resultan 2 categorías.
- Se agrupan 15 niveles para la variable *city.* Resultan 7 categorías
- Se agrupan 2 niveles para la variable *registered_via.* Resultan 4 categorías.

- En la fase siguiente se realiza la **codificación "One Hot"** de todas las variables categóricas. Esta codificación consiste en convertir cada categoría de respuesta en una variable numérica binomial o dicotómica (0 si no presenta esa característica y 1 si la presenta).
- Por último, se comprueba la varianza para todas las variables resultantes del proceso. En este paso se prevé **la eliminación de aquellas variables con una dispersión cercana a 0**. En el caso concreto analizado no aplica la eliminación de variables de este control[7].

Aplicadas todas estas transformaciones y codificaciones se obtiene un conjunto de datos final con una dimensión de 860.966 filas y 45 columnas. Esta versión final del conjunto de datos es la empleada para la aplicación de los modelos.

4.1.4. Resumen del conjunto de datos final.

Todas las transformaciones para obtener la versión del conjunto de datos final, pretenden, en su propio desarrollo, dos cuestiones exportables a un entorno de trabajo real con estos datos:

1. Simular un marco sistemático para el tratamiento de los datos KKBox. Este maro permitiría replicar todas las operaciones de obtención, limpieza y preparación de datos en el caso de que se propusiese una actualización de los mismos. En este sentido, se desarrolla un ciclo para el aprendizaje por máquina permanente, actualizable y reproducible.
2. Dotar a los datos de una estructura y organización óptima para la predicción de la fuga de clientes. Durante la construcción del conjunto final se pretende asegurar la robustez de los datos, de forma que la capacidad predictiva de los modelos aplicados tenga un buen punto de partida para la mejora ya sea vía técnicas de balanceo de datos (como es el caso) o por ajuste de hiperparámetros.

La estructura del conjunto de datos se detalla en la **Tabla 12**.

[7] Concretamente se toma como valor de referencia el coeficiente de variación entre la desviación típica y la media $\sigma/\bar{X}$ que se compara con el valor de un umbral que para este estudio se ha fijado por defecto en 0,1.

Tabla 12: Estructura del conjunto de datos final.

Nombre de variable	Descripción	tipo
msno	Id de usuario	Discreta
is_churn	Cliente fugado (1) / cliente no fugado (0)	Discreta dicotómica
total_payment_days	Días totales de suscripción a servicios	Numérica
total_amount_paid	Cantidad de dinero pagada en todas las suscripciones	Numérica
n_auto_renew	Número de suscripciones autorrenovadas	Numérica
n_cancel	Número de cancelaciones de servicio	Numérica
n_churn_past	Número de fugas en el pasado	Numérica
discount_amount	Cantidad de dinero disfrutado en descuentos	Numérica
last_is_cancel	Cancelación / no cancelación de la última suscripción activa	Numérica
n_transactions	Número de suscripciones disfrutadas	Numérica
n_plan_changes	Número de cambios de plan	Numérica
past_subscript_days	Número de días totales suscrito a servicios	Numérica
num_25	Número de canciones escuchadas a menos del 25% de su duración en el último mes	Numérica
num_50	Número de canciones escuchadas entre el 25% y 50% de su duración en el último mes	Numérica
num_75	Número de canciones escuchadas entre el 50% y 75% de su duración en el último mes	Numérica
num_985	Número de canciones escuchadas entre el 75% y 98,5% de su duración en el último mes	Numérica
num_100	Número de canciones escuchadas entre el 98,5% y 100% de su duración en el último mes	Numérica
num_unq	Número de canciones distintas escuchadas en el último mes	Numérica
total_secs	Segundos de reproducción de canciones en el último mes	Numérica
n_activity	Número de días en los que hay actividad registrada durante el último mes	Numérica
days_registration	Días desde la fecha de registro (creación de cuenta)	Numérica
days_first_trans	Días desde el inicio de la primera suscripción	Numérica
ternure_end_obs	Días de duración del último periodo de suscripciones activas	Numérica
last_plan_paid (2)	Último plan contratado	One hot 2 dicotómicas: **Planes**: "30", "other"
last_payment_method (7)	Último método de pago	One hot 7 dicotómicas: **Métodos**: "36", "37", "38", "39", "40", "41", "other"
usual_plan (2)	Plan más comunmente contratado en el historial de suscripciones	One Hot 2 dicotómicas: **Planes**: "30", "other"
city (7)	Ciudad	One Hot 7 dicotómicas: **Ciudad**: "1", "4", "5", "13", "15", "22", "other"
registered_via (4)	Vía por la que se formalizó el registro	One Hot 4 dicotómicas: **Vía**: "3", "7", "9", "other"

Consolidada la estructura de datos final se está en disposición de aplicar las técnicas ya descritas y analizar las diferencias en el rendimiento de las técnicas de clasificación aplicando los distintos métodos para el balanceo de datos. El siguiente epígrafe recoge los resultados detallados de la parte empírica de este trabajo.

4.2. Resultados y evaluación de modelos: aplicación de métodos de balanceo y evaluación de impacto sobre los modelos predictivos.

En este epígrafe se presentan los resultados derivados de la aplicación de las técnicas de *resampling* en combinación con distintos modelos de clasificación. Todas las operaciones para la construcción y aplicación de los modelos se han realizado en el lenguaje Python concretamente su versión 3.6.

La aplicación de las técnicas de clasificación y la extracción de métricas se realiza con el paquete para la ciencia de datos *scikit-learn*[8], una completa librería que permite integrar aplicación de modelos y extracción de métricas sin más dependencias que las que se incluyen en ella. En lo relacionado con los algoritmos de balanceo de datos se ha trabajado íntegramente con el paquete *imbalanced-learn*[9], librería que, al igual que la anterior, permite depender de tan solo un paquete para la aplicación de técnicas de *resampling*. En cuanto a los soportes gráficos, la herramienta de visualización elegida es *matplotlib*[10]. Se ha elegido el empleo de estas librerías y no otras por permitir un número de dependencias reducido en la elaboración del trabajo y además ser tres plataformas de uso ampliamente extendido.

Ya explicados los modelos, en esta parte de experimentación, se propone el empleo de 8 técnicas de balanceo en combinación con las 3 técnicas de clasificación ya explicadas:

- Regresión logística tipo Lasso (*LogReg*).
- Bosques aleatorios (*RandomForest*).
- *Gradient Boosting* (*GraBoost*).

En cuanto a los algoritmos de *resampling* no se presentan todas las técnicas explicadas en el capítulo anterior. En el caso de *Condensed Nearest Neighbours* se ha debido dejar fuera por el alto coste computacional al implementar la técnica con un gran volumen de datos. En cuanto a las soluciones *NearMiss* se presenta la opción 3 al ser la única versión que ofrece resultados de interés. Resumidamente las técnicas

[8] https://scikit-learn.org/stable/
[9] https://imbalanced-learn.readthedocs.io/en/stable/
[10] https://matplotlib.org/

de balanceo de datos que se presentan en este capítulo son las detalladas en la siguiente tabla:

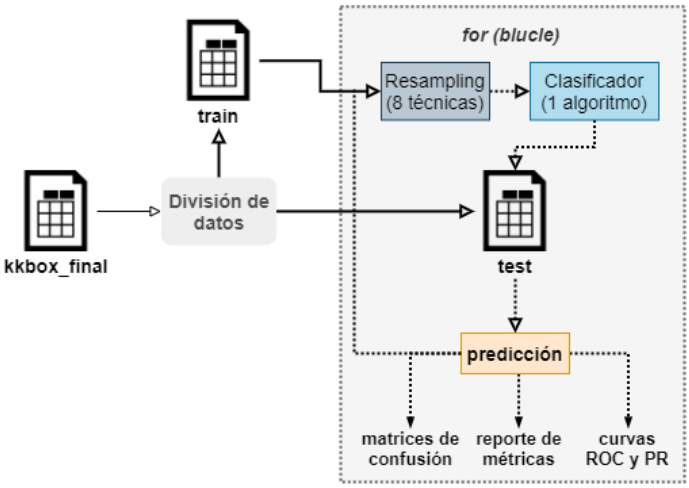
Tabla 13: Relación de técnicas de balanceo de datos presentadas.

Técnica	alias	tipo
Random Undersampling	*rus*	*Undersampling*
Edited Nearest Neighbours	*enn*	*Undersampling*
Tomek Links	*tml*	*Undersampling*
Near Miss (versión 3)	*nm3*	*Undersampling*
Random Oversampling	*ros*	*Oversampling*
SMOTE	*smot*	*Oversampling*
ADASYN	*adas*	*Oversampling*
SMOTE + Tomek Links	*smo_tl*	Híbrido

El flujo de trabajo seguido en esta parte empírica se presenta en la **Figura 18**. Partiendo del conjunto de datos final, se dividen los datos en datos de entrenamiento y datos de test. El conjunto de datos de entrenamiento representa el 80% total de los datos y sobre ellos se realizará la construcción de los modelos. El 20% restante, corresponde al conjunto de test sobre el que se validan los modelos. Realizada esta división se procede iterativamente en un bucle a realizar una secuencia de operaciones con cada algoritmo de clasificación. Cada iteración corresponde a una técnica de balanceo diferente. La secuencia es:

1. **Balanceo** de datos del conjunto de entrenamiento
2. Construcción del **modelo** de clasificación
3. Validación en datos de test (**predicciones**):
   - Extracción de métricas de rendimiento.
   - Visualizaciones.

Figura 18: Esquema del flujo de trabajo en la aplicación de modelos.



La secuencia anteriormente explicada se realiza por separado para cada una de las tres técnicas de clasificación. De esta forma es posible observar la influencia que las técnicas de balanceo de datos tienen sobre el rendimiento de los clasificadores. Explicados el flujo de trabajo y las aplicaciones concretas puestas en funcionamiento se pasa a la presentación de resultados en los puntos siguientes.

4.2.1. Aplicación de algoritmos de *resampling* y técnicas de clasificación.

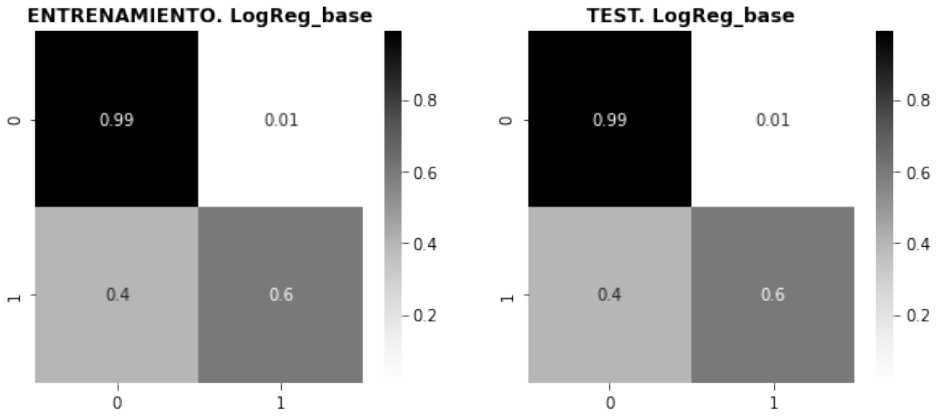
A continuación, se procede a mostrar las diferencias de rendimiento en labores de clasificación con cada uno de los modelos escogidos en combinación con cada una de las técnicas de balanceo de datos ya presentadas. Las configuraciones de hiperparámetros empleadas para cada uno de los clasificadores fueron:

- **Regresión logística tipo LASSO:**
  - Máximo de 1000 iteraciones (valor que evita problemas de convergencia del algoritmo)
- **Random Forest**:
  - Número de árboles = 100
  - Número de casos mínimo para crear un nodo = 2
  - N variables de entrada a para determinar la decisión en un nodo = $\sqrt{p}$
- **Gradient Boosting:**
  - Número de árboles = 100
  - Número de casos mínimo para crear un nodo = 2

4.2.1.1. Resultados con regresión logística.

Como se avanzaba en el capítulo metodológico una de las técnicas empleadas en las labores de clasificación es la regresión logística y concretamente la regresión tipo LASSO.

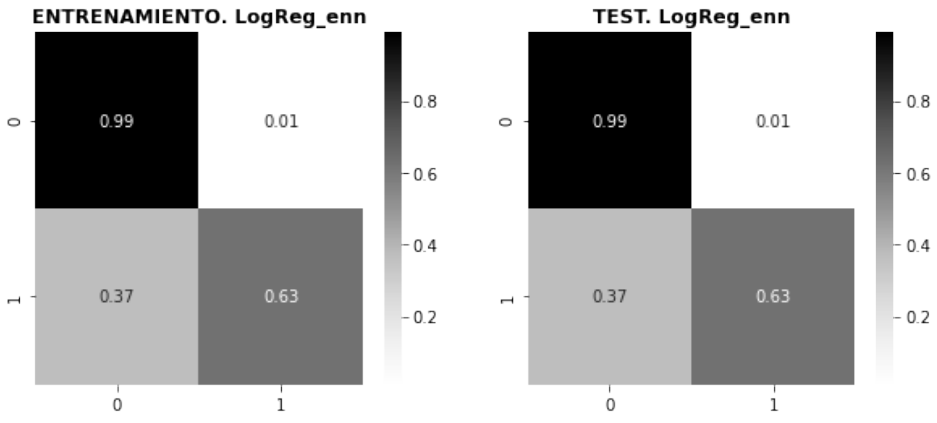
Figura 19: Matrices de clasificación del modelo de regresión logística sin balanceo.



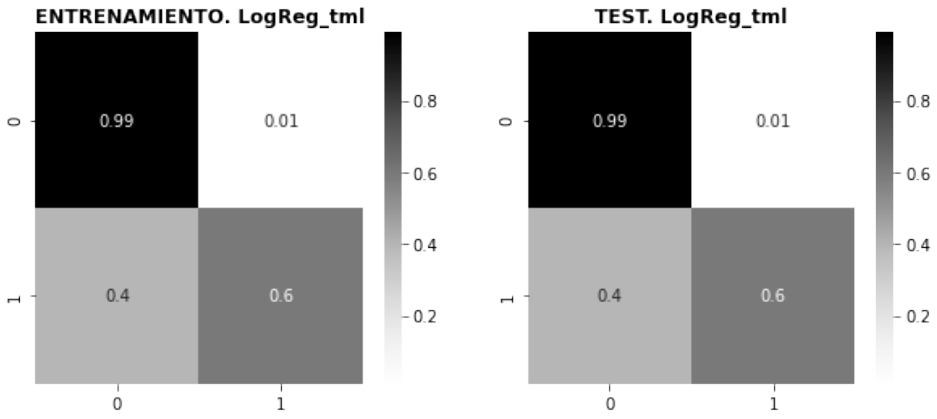
La **Figura 19** muestra los resultados de clasificación del modelo de regresión logística sin la aplicación de técnicas de balanceo. Los resultados de clasificación ajustan casi perfectamente la predicción de clase entre los clientes no fugados (categoría 0). Vemos que el problemas común y recurrente, que avanzábamos en la introducción, con datos desbalanceados es un valor notablemente menor de la proporción de casos bien clasificados en la categoría de interés (clientes fugados); el valor es concretamente de un 59,8% de casos bien clasificados en esta categoría.

Se constata en estos resultados que el punto de partida en las tareas de clasificación es relativamente bueno. El resultado global de ajuste del modelo es del 95,7% de casos bien clasificados y un valor F1 del 0,726, lo que está indicando cierto equilibrio entre la precisión y sensibilidad. Estos resultados de alguna manera justifican positivamente el trabajo previo en la preparación y limpieza de la información como paso clave para garantizar buenos resultados a partir de los cuáles sea posible mejorar los modelos. La pregunta, que surge aquí es ¿la predicción correcta de un 60% de las fugas es suficiente en trabajos de estas características? En principio la respuesta a esta pregunta sería no. Lo cierto es que hay un 40% de fugas que no se está en disposición de detectar. En relación a los intereses directos que podría tener la compañía[11], se estaría renunciando a la posibilidad de conservar un porcentaje considerable de clientes en el caso de iniciar alguna acción de retención. Veamos, por tanto, a partir de este punto la influencia que las distintas técnicas de balanceo tienen sobre los resultados de una regresión logística.

Figura 20: Matrices de clasificación de los modelos de regresión logística previa aplicación de técnicas de *undersampling* (Edited Nearest Neighbours y Tomek Links).



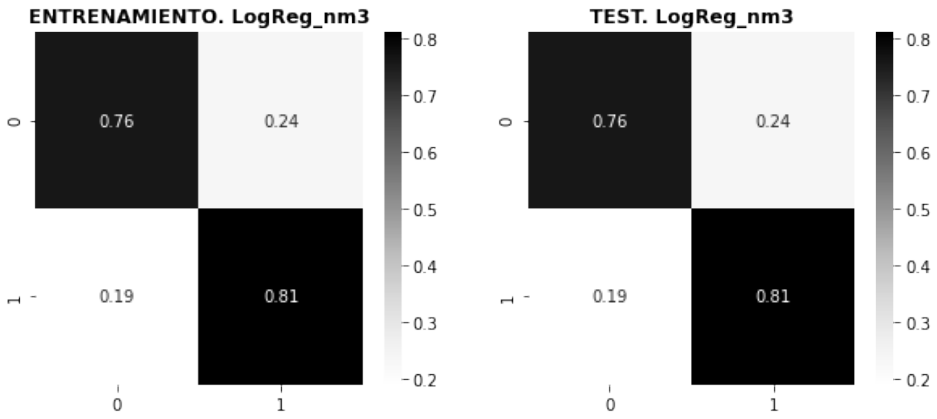
[11] Los costes de captación de nuevos clientes en relación al mantenimiento de los existentes, se referencian en el capítulo 3 de este trabajo.



En la figura **Figura 20** se pueden observar las diferencias de rendimiento en la aplicación de dos algoritmos de *resampling* que se basan en la eliminación de instancias recurrentes, como son *Edited Nearest Neighbours (ENN)* y *Tomek Links.* Se observa en las matrices de clasificación una mejora mínima en las predicciones a la clase minoritaria (clientes fugados). Vía sensibilidad, para el caso de ENN, la mejora de las predicciones no supera el 3% con un valor para la precisión del 0,63. En el caso del empleo de *Tomek Links* el valor se mantiene en el 0,6 sin inducir mejoras en la clasificación.

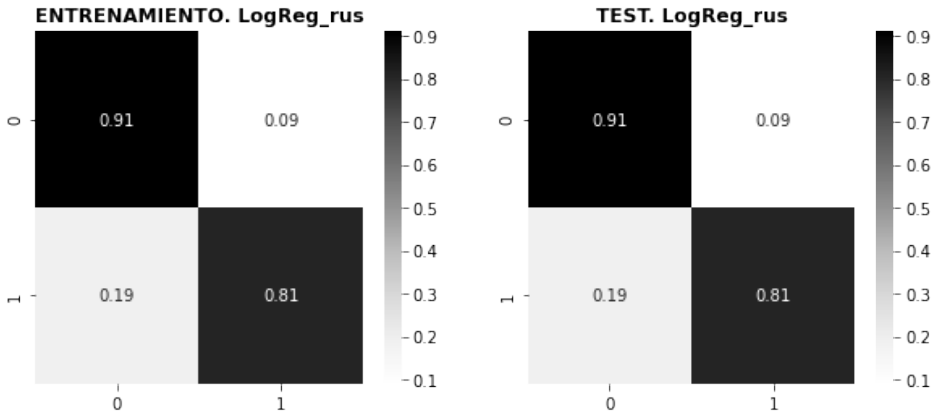
Veamos lo que sucede ahora con la eliminación de instancias siguiendo el método *NearMiss.* Recordemos que esta técnica se centraba en la decidir qué instancias conservar en la clase mayoritaria, y además, las decisiones de vecindad para hacerlo se fundamentaban en distancias medias a los vecinos opuestos.

Figura 21: Matrices de clasificación de los modelos de regresión logística previa aplicación de *undersampling* con la versión 3 de *Near Miss*.



bservamos que esta solución no resulta óptima en lo que a clasificación global se refiere. La pérdida en las predicciones a la clase mayoritaria es enorme y aunque el ajuste entre los fugados crezca notablemente, el efecto de esta solución otorga una gran inestabilidad al modelo.

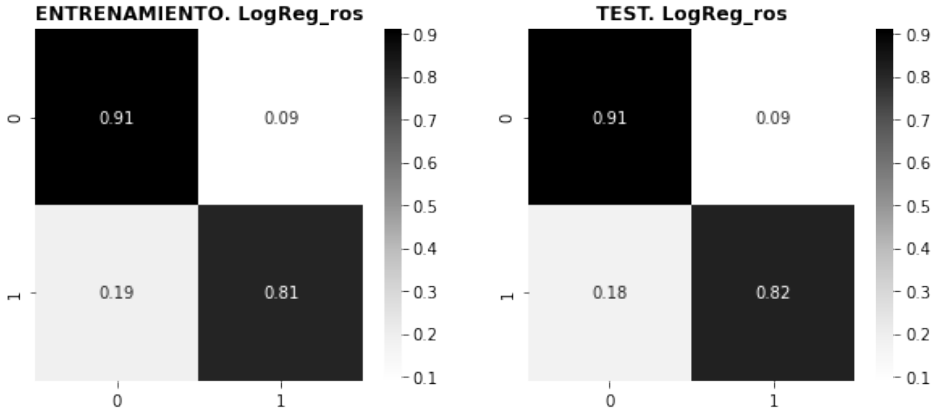
Figura 22: Matrices de clasificación de los modelos de regresión logística previa aplicación de *Random Undersampling*.



Siguiendo con la evaluación del rendimiento en las tareas de clasificación de métodos de *undersampling,* vemos en la **Figura 22**, que, en este grupo de técnicas, los resultados más destacables en cuanto a la mejora de las predicciones en el grupo de los clientes fugados es la aportada por el método no heurístico (*Random Undersampling*). La sensibilidad en este caso aumenta hasta el 0,81 (no tanto como en el caso anterior) pero sin vulnerar de forma drástica el ajuste de la clase mayoritaria.

En este sentido, y pasando al conjunto de técnicas de balanceo por *oversampling,* nótese que el *random oversampling* (**Figura 23**) ofrece variaciones en el rendimiento muy similares a las ofrecidas por el *undersampling* aleatorio. Desde el punto de vista de la clasificación los resultados son prácticamente idénticos.

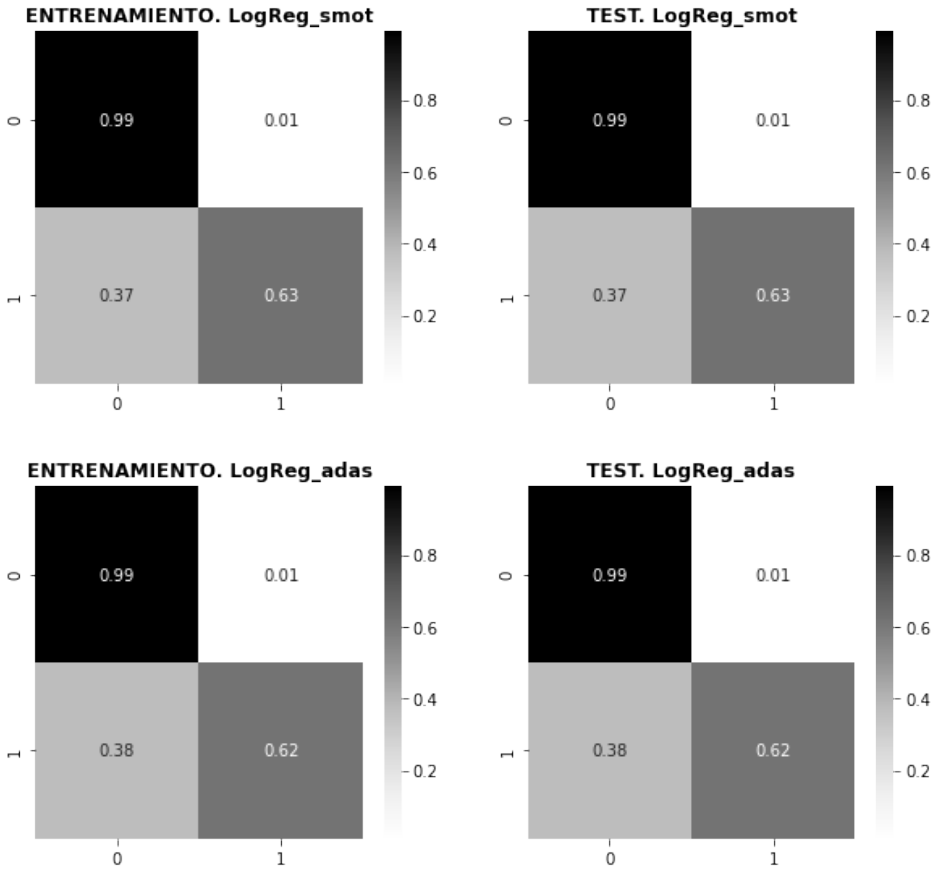
Figura 23: Matrices de clasificación de los modelos de regresión logística previa aplicación de *Random Oversampling*.



Veamos en este punto los resultados que ofrecen las técnicas de *oversampling* sintéticas, que tenían como objetivo replicar instancias de la clase minoritaria en las líneas que unirían vecinos de la misma clase. Analizando la **Figura 24** parece que en

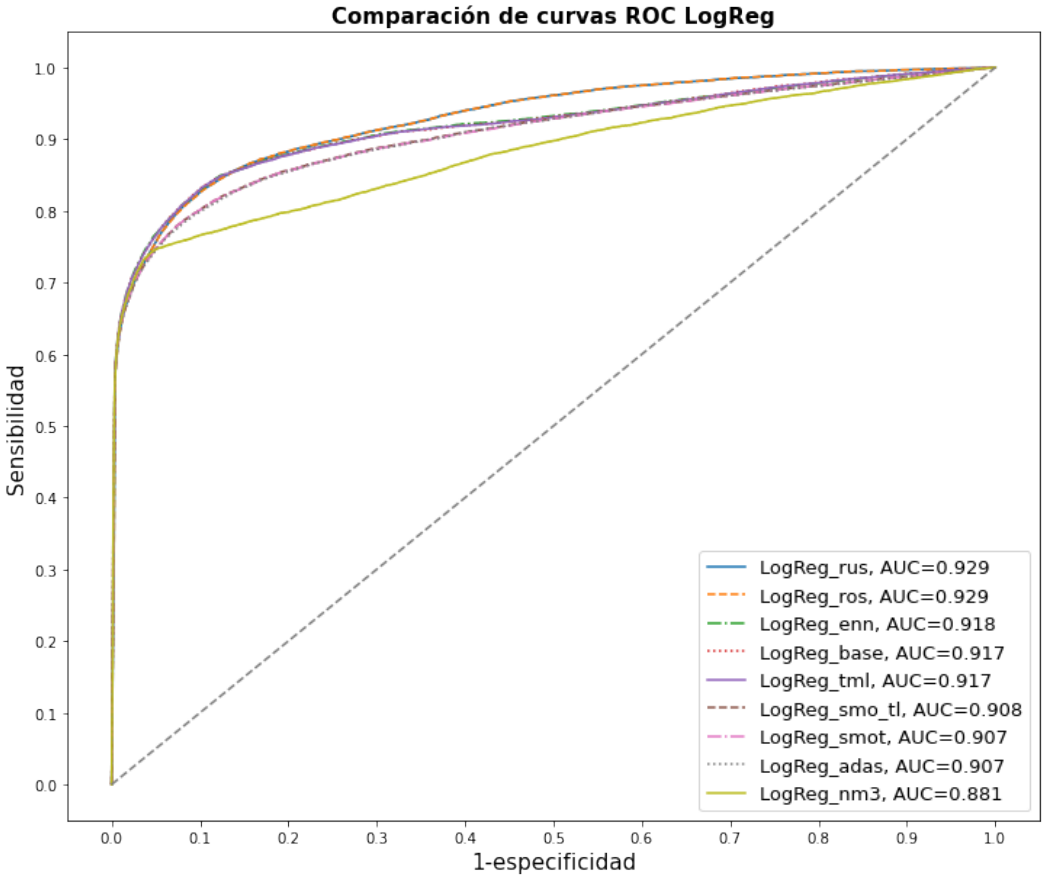
el caso concreto de estudio la ocupación de esos espacios por nuevas instancias no mejora sustancialmente el trabajo de clasificación.

Figura 24: Matrices de clasificación de los modelos de regresión logística previa aplicación de *oversampling* sintético con SMOTE y ADASYN.



En nuestro estudio de caso, el rendimiento de las técnicas que eliminan instancias recurrentes de la clase mayoritaria como *Tomek Links* o *ENN* ofrecen resultados muy similares a los ofrecidos por algoritmos sintéticos de *oversampling.* El algoritmo híbrido elegido para este trabajo *SMOTE+Tomek Links* arroja los mismos resultados de clasificación aportados por *SMOTE o ENN.*

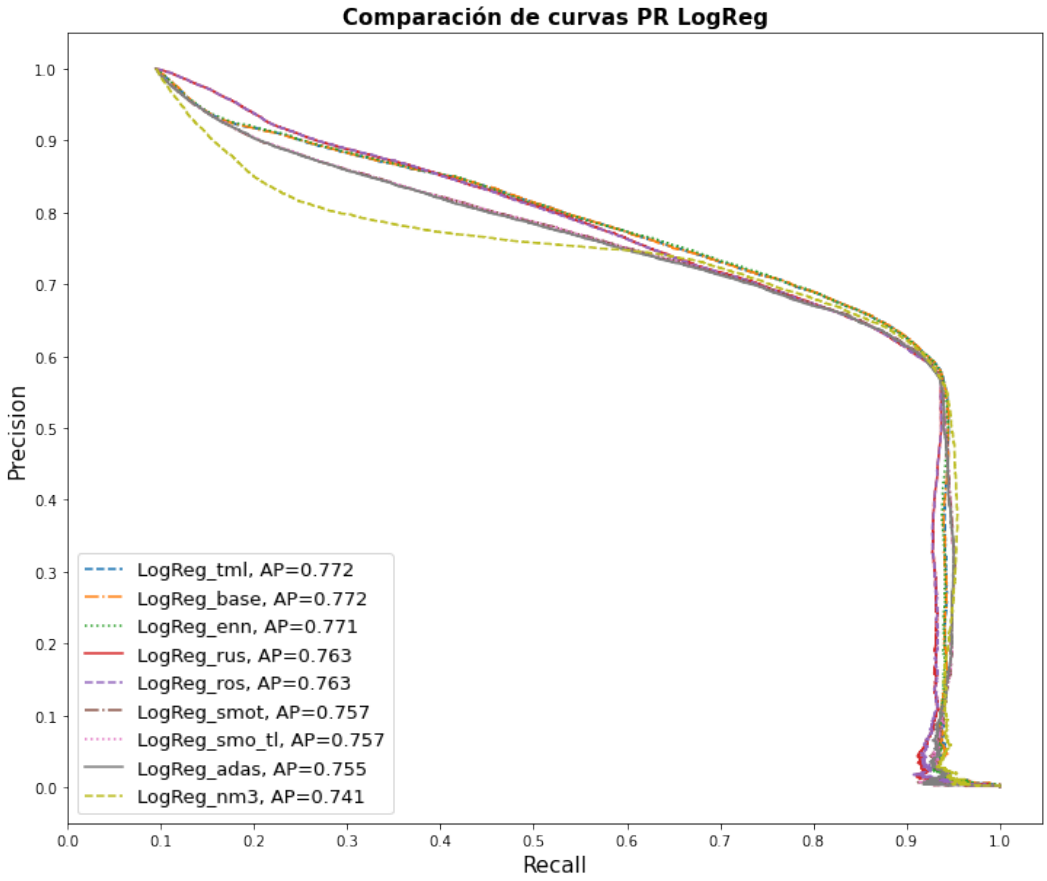
Figura 25: Comparativa de curvas ROC de las técnicas de *resampling* combinadas con una regresión logística tipo LASSO.



Si analizamos la **Figura 25** podemos observar algunas de las diferencias observadas hasta ahora. Los mejores valores para el Área Bajo la Curva (AUC) y que ocupan posiciones más altas de las curvas en sensibilidad para niveles bajos de *1-especificidad* son los métodos de *resampling* no heurísticos como el RUS y el ROS.

Observando la **Figura 26** se confirma que ambos métodos (RUS y ROS) parten de mayores niveles de presión que el modelo base para niveles bajos de sensibilidad. Es decir, lo que indican las gráficas es que el RUS y el ROS sostienen sus buenos resultados globales perdiendo precisión para mejorar notablemente la sensibilidad de los modelos.

Figura 26: Comparativa de curvas PR de las técnicas de *resampling* combinadas con una regresión logística tipo LASSO.



Los resultados anteriormente expuestos pueden matizarse de forma más extensa en base a lo mostrado en la **Tabla 14**. En primer lugar, vemos que los modelos que tienen un mejor ajuste global si se tiene en cuenta la exactitud son aquellos que se parecen más al modelo base, aquellos que hemos adelantado que ofrecían pocas mejoras en las tareas de clasificación en la detección de fugas de clientes.

Tabla 14: Métricas de rendimiento de los trabajos de clasificación de una regresión logística usando los distintos métodos de resampling.

Modelo	Accuracy	Sensibilidad	Precisión	F1-score	AUC	AP	Log-loss
LogReg_base	0,957	0,598	**0,924**	0,726	0,917	**0,772**	**0,145**
LogReg_rus	0,902	**0,814**	0,488	0,610	**0,929**	0,763	0,297
LogReg_enn	**0,958**	0,634	0,890	**0,740**	0,918	0,771	0,147
LogReg_tml	**0,957**	0,604	**0,919**	0,729	0,917	**0,772**	**0,145**
LogReg_nm3	0,767	0,810	0,263	0,397	0,881	0,741	0,441
LogReg_ros	0,900	**0,816**	0,483	0,607	**0,929**	0,763	0,298
LogReg_smot	**0,957**	0,629	0,881	0,734	0,907	0,757	0,159
LogReg_adas	**0,957**	0,624	0,887	0,733	0,907	0,755	0,160
LogReg_smo_tl	**0,957**	0,630	0,880	**0,734**	0,908	0,757	0,159

No obstante, estos resultados son buenos fundamentalmente gracias al ajuste entre los clientes no fugados (clase mayoritaria), ya que presentan los peores valores de

sensibilidad. En este apartado queda reflejado que los métodos no heurísticos ofrecen los mejores resultados en esta línea, pero bajan en exactitud a niveles del 90%.

Como resulta lógico las mejoras predictivas en la categoría minoritaria (clientes fugados) llevan aparejada una merma de la precisión del modelo, es decir, estaríamos sobredimensionando la fuga de clientes en nuestras predicciones obteniendo niveles de precisión bastante bajos si los comparamos con los valores de los modelos base o *LogReg_tml.* Pero, a tenor de lo observado en la **Figura 25** y **Figura 26** esta pérdida está uniformemente repartida en los distintos puntos de la curva (al contrario de lo que sucede con *NearMiss* por ejemplo). Además, teniendo en cuenta medidas globales como la precisión media (AP) queda matizada esa pérdida. Tomando como medida global de rendimiento el AUC tanto ROS como RUS ofrecen los mejores resultados.

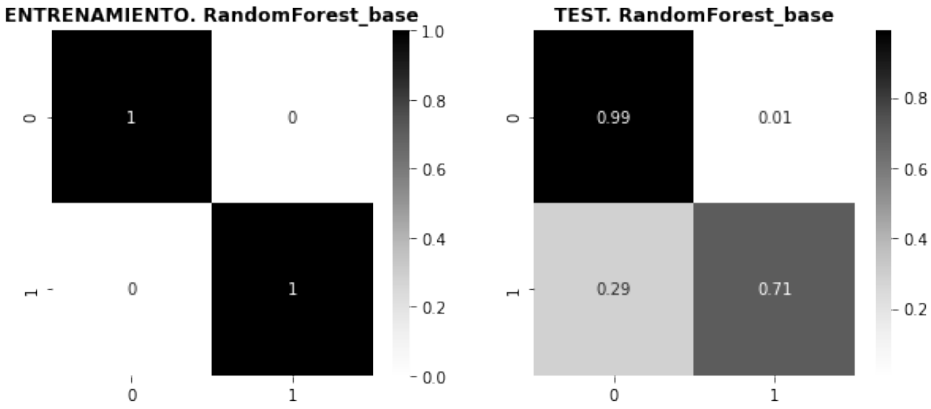
Pero conviene apuntar y matizar el tipo de problemas al que nos enfrentamos en los trabajos de clasificación binaria. Si se analiza la pérdida logística (Log-Loss) de nuestras predicciones probabilísticas (no de clase) vemos que tanto RUS como ROS ofrecen los peores resultados después de NearMiss. Es decir, las mejorar predictivas mostradas por los métodos no heurísticos se vinculan al problema que se trata de resolver, asumiendo una pérdida en la precisión de las predicciones.

4.2.1.2. Resultados con bosques aleatorios.

El trabajo con bosques aleatorios modifica sustancialmente los resultados observados en la regresión logística. Se parte de la base de que es la técnica que ofrece los mejores resultados de clasificación de forma global de entre los tres clasificadores escogidos. Esta técnica tiene en la base de su funcionamiento el trabajo con muestras *bootstrap* (*bagging*). Es por ello, que de alguna manera al aplicar técnicas de balanceo vamos a incidir sobre la muestra dos veces; una en el propio momento de balancear los datos y otra en cuanto a que el "resampling" influirá en las muestras bagging que realizará la propia técnica.

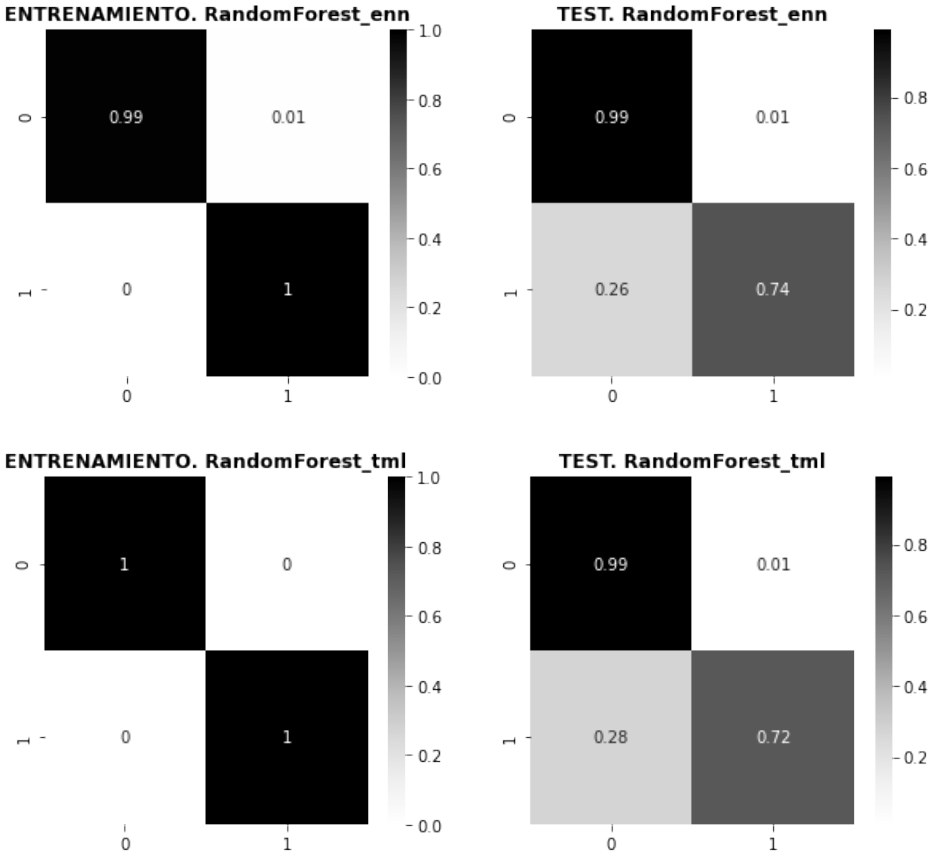
La aplicación del modelo base, ofrece una mejora muy notable en el problema planteado. Concretamente, el modelo aumenta su exactitud en 10 puntos, todos ellos gracias a la mejora de la sensibilidad del modelo (clasificación correcta en la clase minoritaria y de interés en nuestro caso) prediciendo correctamente un 71,5% de las fugas reales de clientes. Se parte, en este caso, de resultados notablemente mejores para predecir las fugas.

Figura 27: Matrices de clasificación del modelo con bosques aleatorios sin balanceo.



Echando un vistazo a las primeras implementaciones de *undersampling* (*ENN* y *Tomek Links*) se desliza una primera conclusión, y es que, en cualquier caso, el trabajo con esta técnica tiende claramente al sobreajuste si se analizan los resultados sobre los datos de entrenamiento.

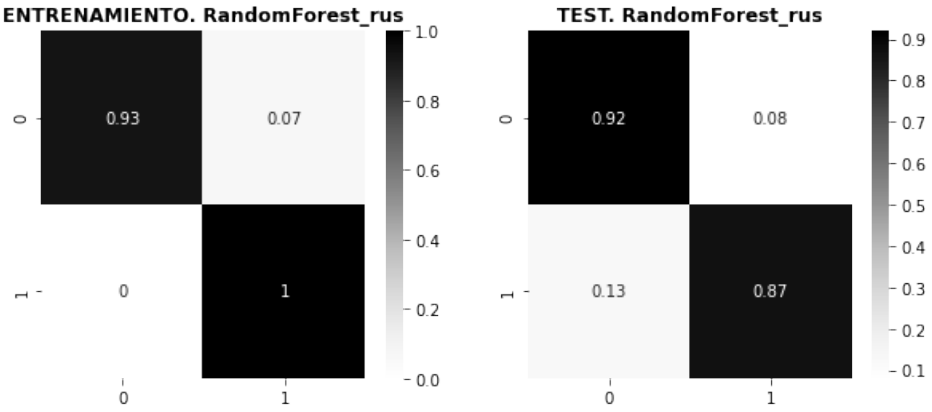
Figura 28: Matrices de clasificación de los modelos de bosques aleatorios previa aplicación de *ENN y Tomek Links*.



Volviendo a las técnicas de balanceo, las predicciones de clase vuelven a ser muy estables respecto al modelo base. ENN eleva hasta el 73,7% las clasificaciones correctas en la clase minoritaria y con Tomek Links apenas se aprecian diferencias (de 71,5 a 71,9%), mostrando niveles de exactitud de una milésima menos, por lo que no están mejorando el modelo propuesto de base. Siguiendo en la línea en lo visto enlos resultados de la regresión logística, la solución *NearMiss* eleva hasta el 0,85 la sensibilidad asumiendo una merma todavía mayor en el rendimiento sobre la clase mayoritaria con un descenso a un 66% de no fugados correctamente clasificados, desvelándose como una técnica inservible para el problema propuesto (véase **Tabla 15**) si se quieren buenos niveles de predicción en ambas categorías.

Como ya sucedía con la regresión logística el rendimiento de los bosques aleatorios previa aplicación de *undersampling* aleatorio ofrece los mejores resultados desde el punto de vista de la predicción de la fuga de clientes.

Figura 29: Matrices de clasificación de modelo con bosques aleatorios logística previa aplicación de *Random Undersampling*.



Como se ve en la **Figura 29** la mejora de los resultados de clasificación en la clase minoritaria es muy destacada elevando la predicción correcta al 86,6%, minimizando a pérdida en la clase mayoritaria, que se mantiene en un 93% de predicción correcta para los no fugados.

En lo que respecta a las técnicas de *oversampling* no es posible constatar mejoras más allá de algunas similares a las ya mostradas por los algoritmos de limpieza *ENN* o *Tomek Links.* Analizando la **Tabla 15** se confirman los resultados explicados. En primer lugar, se hace patente la estabilidad de los resultados de las técnicas de balanceo de datos. Estaríamos en disposición de afirmar que en el trabajo con bosques aleatorios replicar instancias ficticias no contribuye a mejorar las tareas de clasificación. Aunque por lo que se deriva de las lecturas de pérdida por regresión logística (log-loss) tampoco contribuyen a aumentar ruido que pueda interferir negativamente en la clasificación. Es más, las predicciones mejoran en cuanto a su

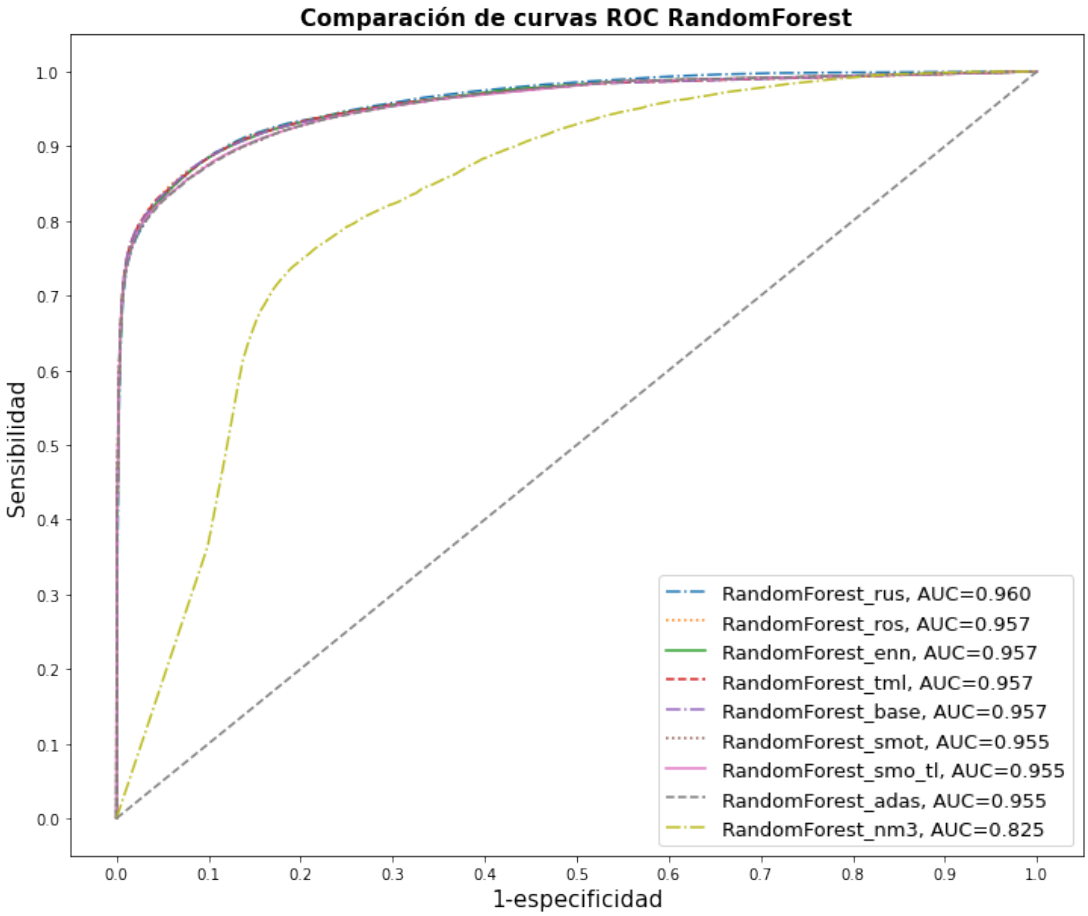
precisión probabilística en comparación con lo mostrado por el modelo base y los modelos que usan *undersampling.* Por lo tanto, los algoritmos de *oversampling,* al menos en relación a los datos tratados, no contribuyen a mejorar las predicciones de clase, aunque parece que mejoran mínimamente los cálculos de las predicciones probabilísticas.

Tabla 15: Métricas de rendimiento de los trabajos de clasificación con bosques aleatorios usando los distintos métodos de resampling.

Modelo	Accuracy	Sensibilidad	Precisión	F1-score	AUC	AP	Log-loss
RandomForest_base	**0,967**	0,715	**0,913**	0,802	0,957	**0,853**	0,156
RandomForest_rus	0,915	**0,866**	0,531	0,658	**0,960**	0,840	0,268
RandomForest_enn	0,966	0,737	0,879	**0,802**	0,957	0,848	0,157
RandomForest_tml	**0,966**	0,719	**0,908**	**0,802**	0,957	**0,853**	0,157
RandomForest_nm3	0,673	**0,850**	0,204	0,329	0,825	0,267	366333,000
RandomForest_ros	0,966	0,733	0,891	**0,804**	**0,957**	0,852	**0,155**
RandomForest_smot	0,966	0,720	0,900	0,800	0,955	0,847	0,155
RandomForest_adas	0,966	0,718	0,901	0,799	0,955	0,846	**0,154**
RandomForest_smo_tl	0,966	0,720	0,900	0,800	0,955	0,846	0,155

En lo que respecta al modelo que mejor predice en la clase minoritaria *RandomForest_rus,* se puede corroborar lo adelantado con la regresión logística. La mejora vía sensibilidad, 0,866 en este caso, revierte en una merma de la precisión que cae al 0,531. No obstante, los resultados son mejores para todas las métricas de lo que lo eran para la regresión logística. En general, en el trabajo con los bosques aleatorios se constata una gran estabilidad de todos los modelos siendo menos volátiles que en el caso anterior. Globalmente la precisión media (AP) de los modelos mejora notablemente y las predicciones probabilísticas (por Log-loss) parecen ser más estables independientemente de la técnica de balanceo que se emplee. En cuanto al valor F1, la merma en la precisión vuelve a hacer descender esta métrica de manera bastante clara respecto al modelo base. El principal problema en este sentido es que el propio desbalanceo de clases afecta a estas medidas, de forma que una proporción muy pequeña de falsos negativos (predichos como fugas y que no lo son) se traduce en un incremento muy significativo en el denominador de la precisión dado que los verdaderos positivos forman parte de la clase minoritaria y por lo tanto van a tender a ser un número más pequeño, mientras los falsos negativos (pertenecientes a la clase mayoritaria) van a tender a ser un número relativamente grande. En este punto surge la necesidad de ponderar qué es más importante, ser capaces de predecir el mayor número de fugas posibles o contener la cantidad de falsas alertas (predicciones de fuga falsas).

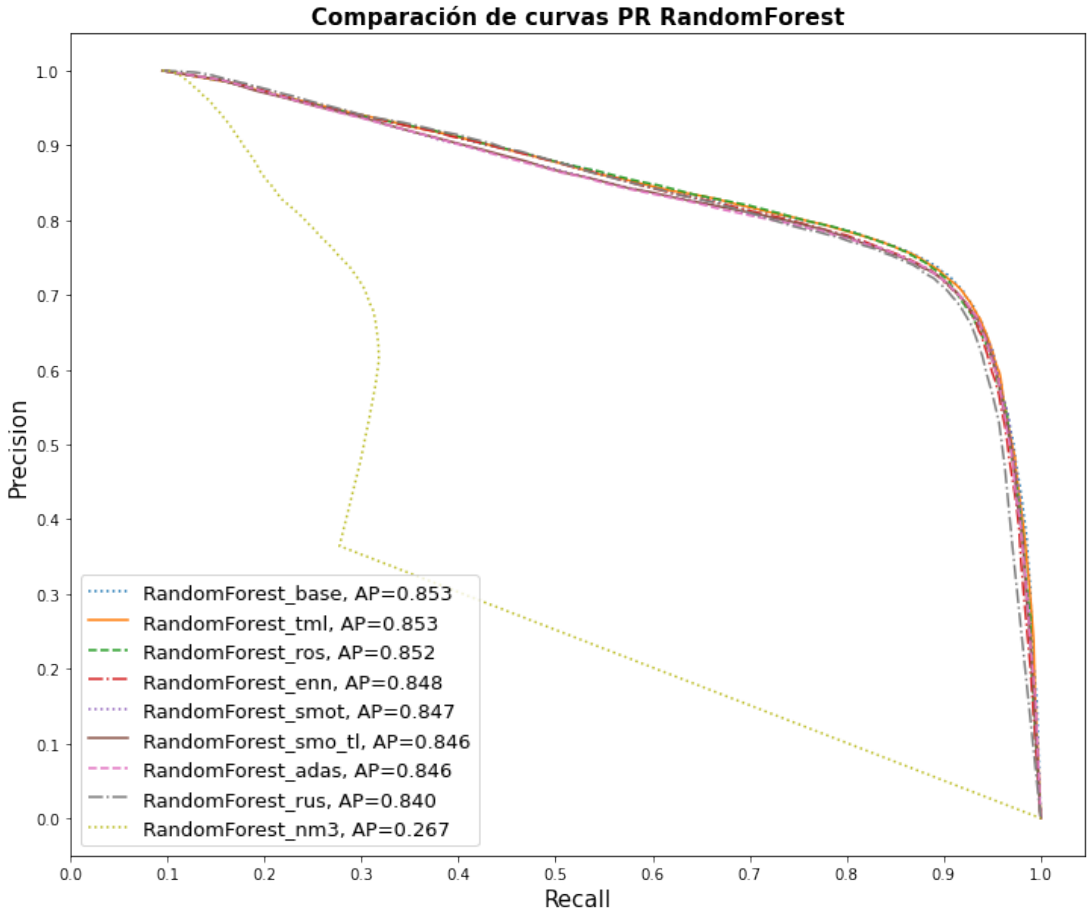
Figura 30: Comparativa de curvas ROC de las técnicas de *resampling* combinadas con bosques aleatorios.



Lo anteriormente expuesto se observa en las gráficas de las curvas ROC (**Figura 30**) y curvas PR (**¡Error! La autoreferencia al marcador no es válida.**). En lo que se refiere al modelo con mayor AUC, el RUS, se aprecian los mejores niveles de sensibilidad. En las curvas PR se aprecia ligeramente como existe una pérdida de precisión en relación al resto de modelos, aunque esta separación no se hace excesivamente grande y no afecta notablemente a la precisión media.

Los modelos ofrecidos por los bosques aleatorios empleando cualquier técnica de *resampling* (excepto *NearMiss*) son relativamente estables y ofrecen resultados similares.

Figura 31: Comparativa de curvas PR de las técnicas de *resampling* combinadas con bosques aleatorios.

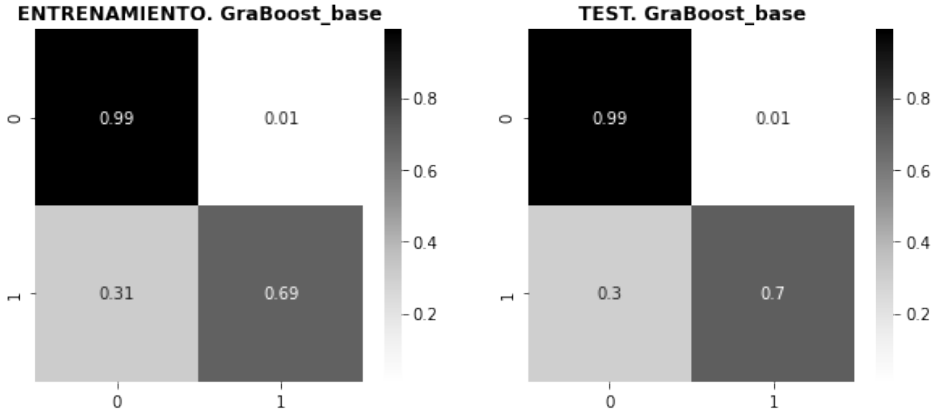


4.2.1.3. Resultados con gradient boosting.

La última de las técnicas de clasificación sobre las que se evalúan los métodos de *resampling* analizados es el *Gradient Boosting*. Como ya se había adelantado es un método que se basa en los clasificadores débiles para construir un clasificador robusto, ofreciendo mayor peso en la clasificación, en cada iteración, a los individuos mal clasificados. En términos generales los resultados que arroja esta técnica en combinación con el balanceo de datos, muestran rendimientos similares a los bosques aleatorios, certificándose diferencias más importantes entre los modelos. Por otro lado, esas variaciones se asemejan a las ofrecidas por la regresión logística.

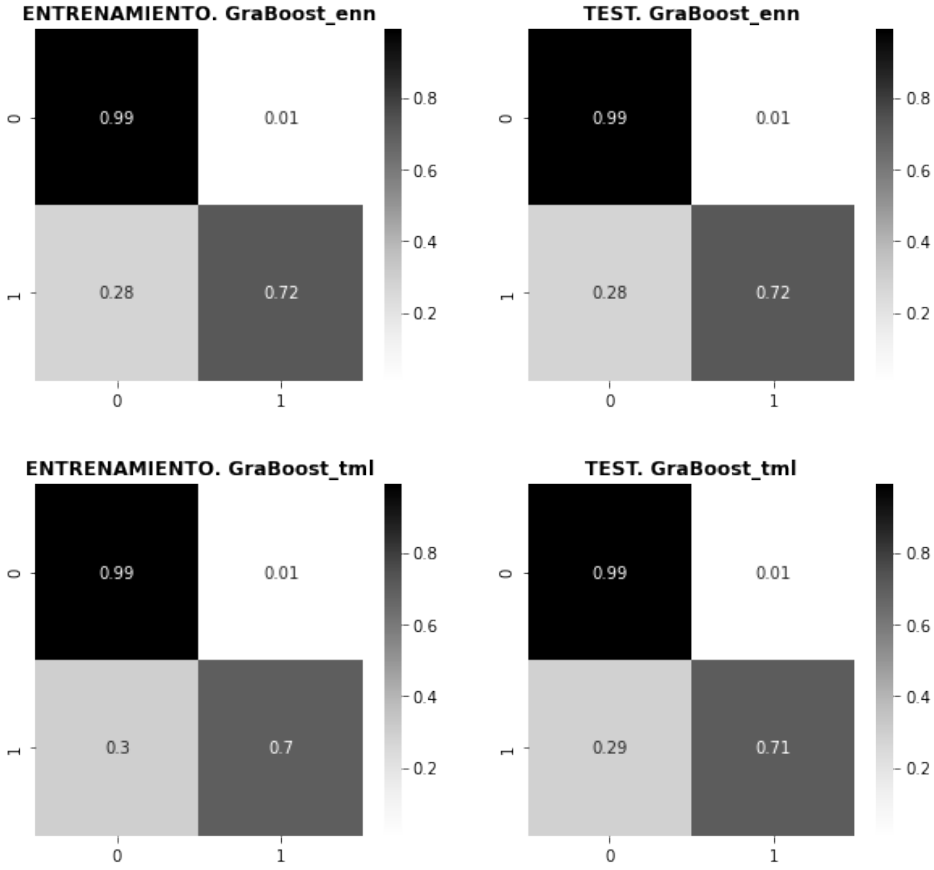
La aplicación de *Gradient Boosting* sin la aplicación de técnicas de balanceo de datos, mejora el rendimiento ofrecido por el modelo teórico (véase **Figura 32**) que determina la regresión logística. En línea con lo que sucedía en los bosques aleatorios, la exactitud se eleva al 0,965 gracias a una ganancia en la sensibilidad de aproximadamente el 10%, situándose en 0,7.

Figura 32: Matrices de clasificación del modelo con *gradient boosting* sin balanceo.



Una cuestión importante de esta técnica en comparación con los bosques aleatorios, es que no se genera sobreajuste y el ajuste sobre los datos de entrenamiento y test es estable para todos los modelos ejecutados.

Figura 33: Matrices de clasificación de los modelos de *gradient boosting* previa aplicación de *ENN y Tomek Links*.

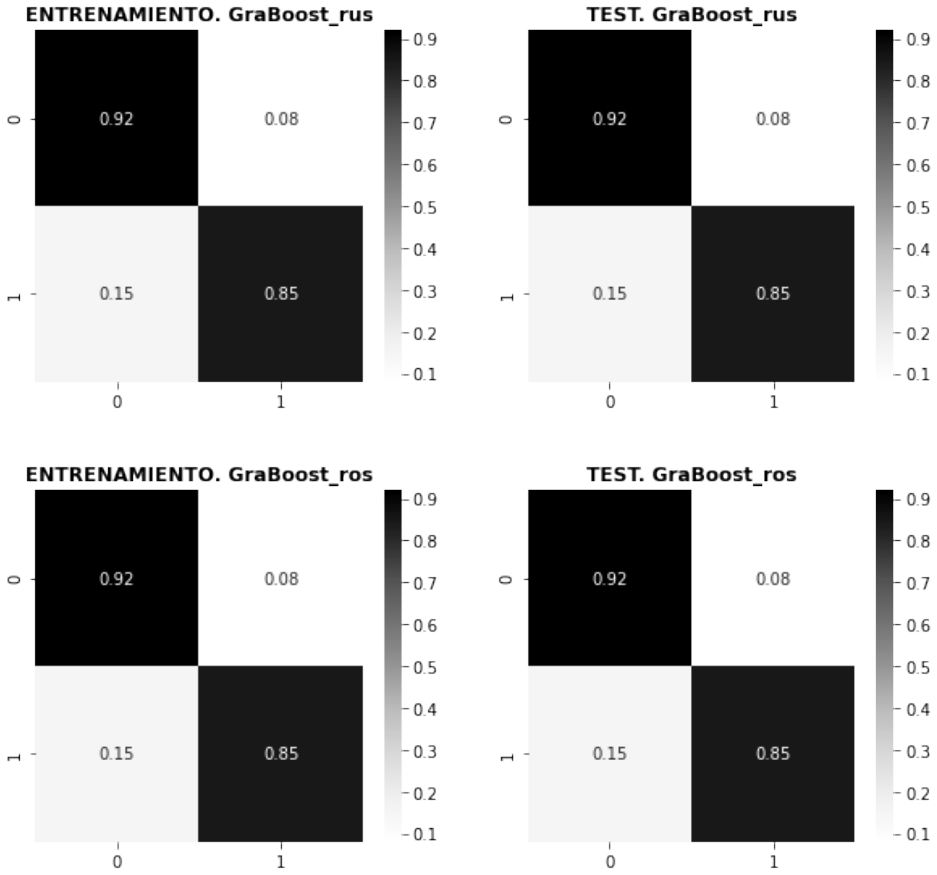


El comportamiento de los algoritmos de *undersampling* replica lo ya mostrados en las dos técnicas anteriores. El mejor comportamiento de una técnica entre los dos

modelos de eliminación de instancias recurrentes *ENN* y *Tomek Links* vuelve a mostrarlo *ENN* con una ganancia en la sensibilidad del 2% mejorando muy levemente el valor F1 (del 0,792 al 0,793; véase **Tabla 16**). Por su parte, haciendo uso de *Tomek Links* el valor F1 aumenta ligeramente hasta 0,794 observándose una mejora también casi inapreciable en la precisión.

Respecto a los métodos de balanceo no heurísticos se vuelven a repetir comportamientos que ya se habían observado en técnicas anteriores. En este caso, el comportamiento se parece más a lo que ocurría con la regresión logística. Los dos métodos de *resampling* aleatorio, *random oversampling* y *random undersampling,* ofrecen los mejores rendimientos en la clasificación entre los clientes fugados.

Figura 34: Matrices de clasificación de los modelos de *gradient boosting* previa aplicación de técnicas de *resampling* no heurísticas (RUS y ROS).

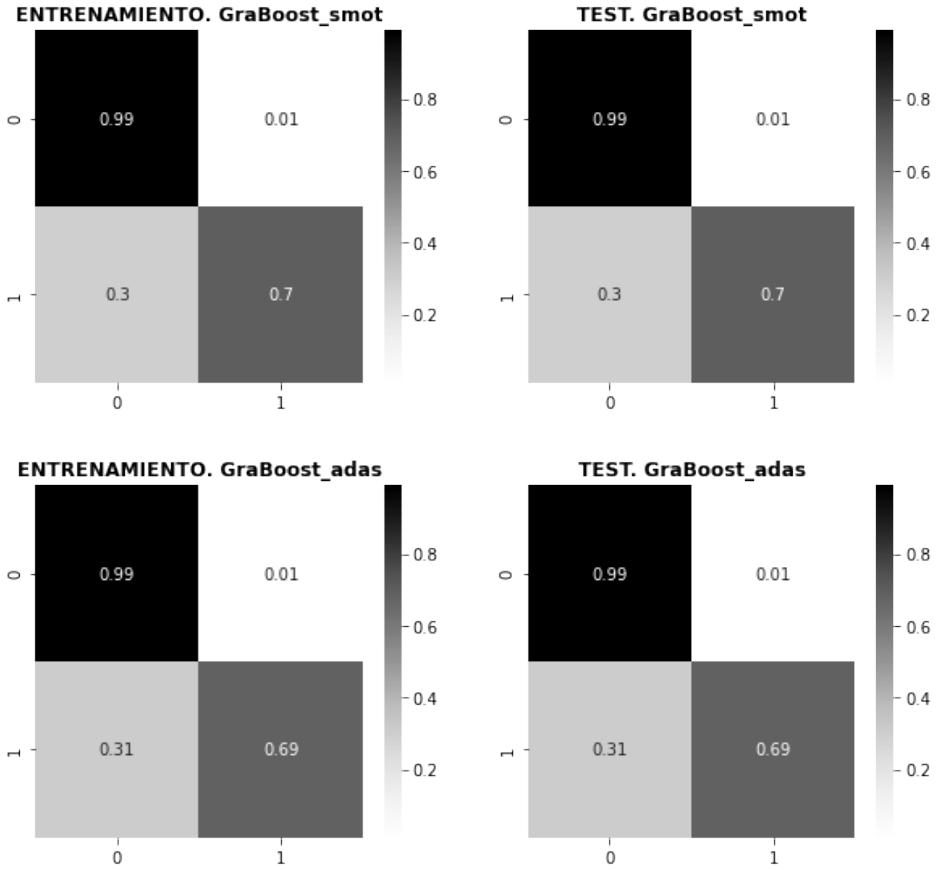


Como se observa en la **Figura 34** la mejora en la clasificación se vincula al aumento en la sensibilidad de los modelos, que se elevan hasta el 0,848 en ambos casos (véase **Tabla 16**), con una merma en el ajuste en la clase mayoritaria que desciende hasta el 92%. El mejor modelo de las dos soluciones no heurísticas es el ROS que alcanza una exactitud del 0,913. Al contrario de lo que sucedía, con los bosques aleatorios en el trabajo con clasificadores débiles, el *oversampling* aleatorio sí contribuye a la

mejora de la predicción de las fugas de clientes. De manera recurrente, hemos visto que los balanceos aleatorios han mostrado las mejores contribuciones en este sentido pudiendo destacar que el *random undersampling* es la más estable para todas las técnicas.

En lo que se refiere a los *oversampling* sintéticos volvemos a certificar peores rendimientos que el resto de técnicas ya vistas en cuanto a la detección de fugas se refiere.

Figura 35: Matrices de clasificación de los modelos de regresión logística previa aplicación de *oversampling* sintético con SMOTE y ADASYN.



Como vemos en la **Figura 35** no solo no se mejora el rendimiento del modelo base, sino que en el caso del ADASYN se empeora en precisión del modelo. Algo que no se había mostrado hasta ahora. Como se aprecia en la **Tabla 16** los valores más bajos para la sensibilidad son los ofrecidos por ADASYN y el modelo híbrido de *SMOTE+Tomek Links.*

Tabla 16: Métricas de rendimiento de los trabajos de clasificación con bosques aleatorios usando los distintos métodos de resampling.

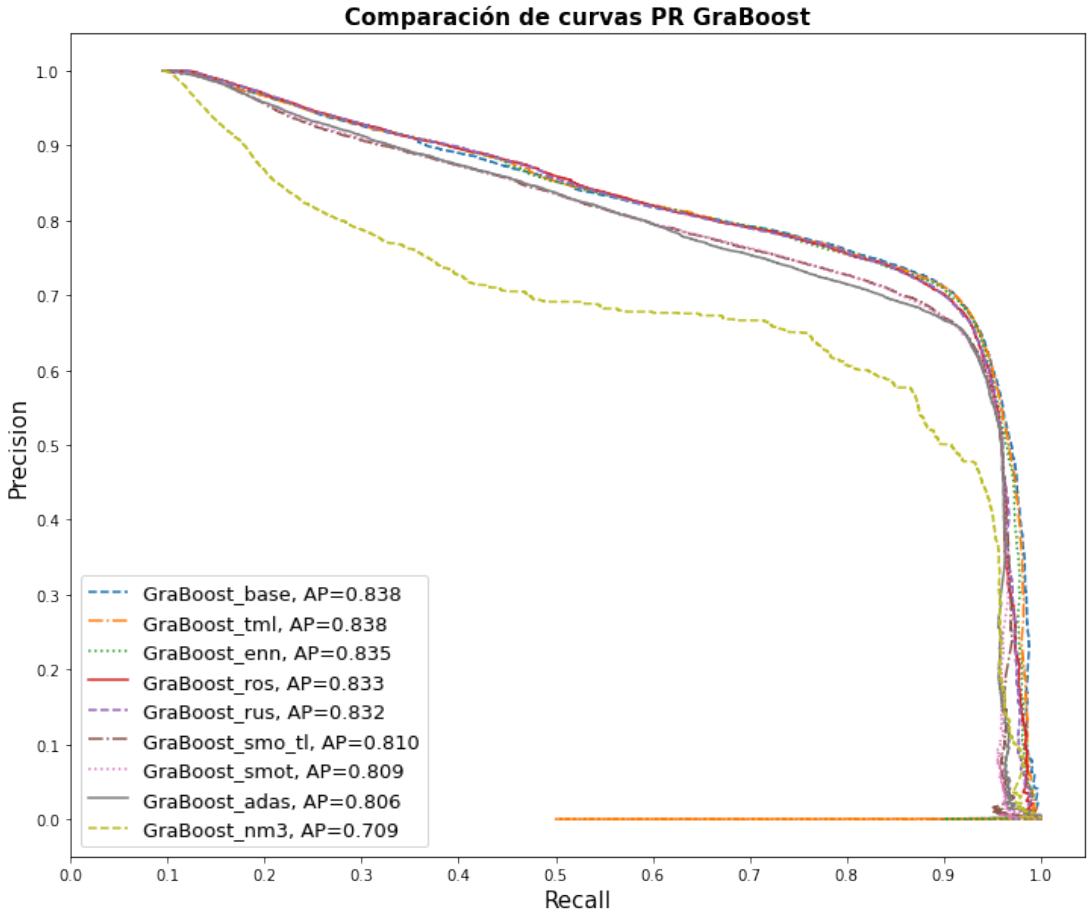
Modelo	Accuracy	Sensibilidad	Precisión	F1-score	AUC	AP	Log-loss
GraBoost_base	**0,965**	0,700	**0,912**	0,792	0,952	**0,838**	**0,120**
GraBoost_rus	0,911	**0,848**	0,519	0,644	**0,953**	0,832	0,244
GraBoost_enn	0,964	0,720	0,882	**0,793**	0,953	0,835	0,121
GraBoost_tml	**0,965**	0,706	**0,907**	**0,794**	0,952	**0,838**	**0,120**
GraBoost_nm3	0,732	0,828	0,237	0,368	0,885	0,709	0,609
GraBoost_ros	0,913	**0,848**	0,523	0,647	**0,954**	0,833	0,244
GraBoost_smot	0,960	0,702	0,853	0,770	0,944	0,809	0,157
GraBoost_adas	0,960	0,689	0,860	0,765	0,944	0,806	0,158
GraBoost_smo_tl	0,961	0,699	0,860	0,771	0,943	0,810	0,157

Avanzando en el análisis de las métricas mostradas en la **Tabla 16** se certifica que, en lo que respecta a pérdida logística de las predicciones, *gradient boosting* es la técnica con resultados más ajustados comparativamente para todos los modelos. En cuanto a los valores de precisión media (AP) de aquellos modelos que sacrifican precisión por ganancia en la sensibilidad solo se desciende del 0,8 para la solución *NearMiss*. En relación a la sensibilidad, para las dos soluciones no heurísticas, el ROS contiene un poco mejor esa pérdida, algo que se refleja también en el F1-score.

De forma general, atendiendo a la tabla se vuelven a mostrar diferencias mínimas en cuanto al AUC como indicador global del modelo, pero que de alguna manera certifican la consideración del ROS y el RUS como las mejores soluciones en combinación con el *gradient boosting.* Para el caso concreto que nos ocupa se vuelve a confirmar que estas soluciones implican una menor precisión de las predicciones probabilísticas con una pérdida logística (*log-loss*) mayor que para el resto de soluciones.

En la **Figura 36** se aprecia con mayor claridad el comportamiento de los modelos asociados a las técnicas de *undersampling,* tanto el aleatorio como los de eliminación de instancias con *Tomek Links* y *ENN,* y el modelo base, como los que mejor se comportan respecto a la predicción en la clase de los no fugados. Es en los valores más altos de la sensibilidad donde se aprecia que las líneas que corresponden al RUS y el ROS se alejan del primer grupo de modelos. Se certifica en esta gráfica el peor rendimiento general al usar algoritmos sintéticos de *resampling* con una merma apreciable de la precisión.

Figura 36: Comparativa de curvas PR de las técnicas de *resampling* combinadas con *gradient boosting*.



5. Conclusiones.

Ejecutadas y analizadas todas las técnicas aplicadas al problema de fuga de clientes planteado, se está en disposición de afirmar que la selección de una u otra técnica ofrece resultados dispares. En el caso particular de lo visto en este trabajo las distintas combinaciones de técnicas de *resampling* y clasificadores desvelan importantes diferencias en el rendimiento a tener en cuenta en la respuesta final para la detección de las fugas de los clientes.

En primer lugar y atendiendo a las distintas técnicas de clasificación aplicadas y con los datos empleados, la técnica que ofrece un mejor rendimiento global en el resultado de clasificación final es los bosques aleatorios (*random forest*). Esta técnica muestra un claro sobreajuste en los datos de entrenamiento, pero destaca en la capacidad predictiva, siendo el modelo que mejor resultados ofrece sin combinación con técnicas de *resampling* con un AUC de 0,957 y alcanzando un nivel de sensibilidad de 0,715. Ofrece el mejor resultado de los clasificadores sin

resampling empleados y de forma sistemática mejora la clasificación en combinación con los distintos algoritmos de *resampling* en comparación con los dos clasificadores restantes. El *gradient boosting* ofrece también buenos resultados en las labores de clasificación mostrando un mejor comportamiento sobre los datos de entrenamiento conteniendo claramente la tendencia al sobreajuste mostrada por los bosques aleatorios. Como modelo teórico, la regresión logística tipo LASSO, ofrece resultados considerablemente buenos en comparación con las otras dos técnicas. Este tipo de regresión se presenta como una solución con un comportamiento muy efectivo en un problema con un alto número de variables independientes, agilizando el proceso de selección de predictores. No obstante, no se alcanzan los buenos niveles de predicción del modelo de los otros dos clasificadores.

Empleando de forma combinada los clasificadores y las técnicas de *resampling* propuestas, las diferencias se hacen menos notables que en el empleo de los modelos de base. Los tres clasificadores ofrecen soluciones notablemente satisfactorias desde el punto de vista predictivo y fundamentalmente en relación a la mejora de la precisión de los modelos. Uno de los elementos importantes en este trabajo es valorar qué tipo de técnicas permiten una mejora de la sensibilidad en detrimento de la precisión para obtener un mejor ajuste en la clase minoritaria y de interés (los clientes fugados). En este caso concreto, la detección del mayor número de fugas de clientes, sería la motivación principal, pero obviamente, siempre y cuando las pérdidas en la clase mayoritaria sean asumibles. Es decir, al investigador se le plantea valorar donde ubicar el punto de equilibrio entre mejorar la detección de las fugas asumiendo que se va a obtener un mayor número de alertas falsas. Sistemáticamente se observa que el empleo de técnicas de balanceo de datos que permiten mejoras de ajuste en la clase minoritaria, implica (lógicamente) una merma en la precisión. Si se asume que es deseable un equilibrio predictivo en ambas clases, de forma clara los métodos que ofrecen un mejor rendimiento en este sentido, son los métodos no heurísticos RUS y ROS.

Tabla 17: Métricas de rendimiento de los métodos equilibrados entre precisión y sensibilidad.

Method	Accuracy	Sensibilidad	Precisión	AUC	AP	Log-loss
RandomForest_rus	0,915	0,866	0,531	0,960	0,840	0,268
GraBoost_rus	0,911	0,848	0,519	0,953	0,832	0,244
GraBoost_ros	0,913	0,848	0,523	0,954	0,833	0,244
LogReg_ros	0,900	0,816	0,483	0,929	0,763	0,298
LogReg_rus	0,902	0,814	0,488	0,929	0,763	0,297

Asumiendo que sea deseable un equilibrio entre precisión y sensibilidad, en la **Tabla 17** se muestran los modelos que mejor ajuste ofrecen en ese sentido. Como ya se avanzaba probablemente los mejores resultados para el problema propuesto son los mostrados por los bosques aleatorios en su combinación con el *undersampling*

aleatorio, destacando por el nivel de sensibilidad más alto del estudio y la menor pérdida de precisión asociada a esa ganancia. Dentro de esta "regla" de equilibrio entre correcta clasificación de las fugas y un nivel de falsa alertación lo más bajo posible, las aplicaciones de *gradient boosting* y regresión logística en combinación con cualquiera de los dos métodos no heurísticos muestran también buenos resultados. Es por ello, que globalmente se concluya que para el problema planteado los métodos de *resampling* que aseguran mejores ajustes en ambas clases son llamativamente los métodos de balanceo de datos no controlados (aleatorios).

En lo que se refiere a los métodos de eliminación de instancias recurrentes de forma sistemática se constatan leves mejoras en la sensibilidad sin incurrir en pérdidas logísticas importantes. Lo que nos indica este comportamiento, es que las contribuciones de este tipo de técnicas de balanceo permiten una leve mejora de clasificación conteniendo mejor la merma en la precisión. En cualquier caso, este tipo de soluciones parecen insuficientes si lo que se pretende es una mejora notable en la predicción de clientes fugados.

Las soluciones *NearMiss* se ubican en otro extremo, en el que la mejora de la precisión es clara en todas sus aplicaciones, pero ofrece soluciones que quedan invalidadas por la tremenda pérdida que se produce en la predicción de los clientes no fugados. No parece viable en ningún caso asumir pérdidas tan grandes para ajustar mejor la detección de fugas reales. De manera similar los algoritmos de *oversampling* sintéticos no parecen ofrecer soluciones al problema. En casi ningún caso se ofrecen mejoras destacables. Parece claro que, al menos para el uso de caso propuesto, la replicación de instancias no ayuda al proceso de decisión del clasificador. Generalmente se han observado comportamientos similares a los ofrecidos por *Tomek links* y *ENN* pero con contribuciones menores en cuanto a mejoras en la detección de la fuga de clientes.

Llegados a este punto las conclusiones generales a las que se llega es que la distribución de los grupos es determinante a la hora de elegir una u otra técnica de *resampling*. Para los datos manejados en este estudio parece claro que la ocupación del espacio con nuevas instancias no ofrece mejoras sustanciales en relación a los resultados de clasificación finales. La eliminación de instancias recurrentes, por su parte, sí que permite ligeras contribuciones en este sentido, pero sin ofrecer ganancias excesivamente altas. En un contexto como el propuesto, parece que son los métodos no heurísticos los que desvelan diferencias de rendimiento muy marcadas que permiten contribuciones muy notables en la predicción de la fuga de clientes.

En lo que se refiere a los elementos que no se han tratado de forma directa en este trabajo, aparecen como determinantes de cara a la elección de uno u otro método de *resampling* cuestiones relativas a la propia composición de la muestra. Como futuras

líneas de interés en este sentido resultaría relevante el análisis de la influencia que tiene en las diferencias de rendimiento el tamaño de la muestra, el nivel de desbalanceo entre las clases o la distribución espacial de los casos, por ejemplo, usando distintos grupos de variables predictoras. Una cuestión que se desvela también en este tipo de problemas, sería el cálculo de costes asociados a las predicciones. Es decir, para una valoración correcta del valor de las predicciones de nuestros modelos sería de interés conocer el coste asociado a no detectar fugas reales de clientes, así como el coste asociado a general una fuga falsa. El conocimiento de estos costes, permitirían valorar rigurosamente qué valores de la sensibilidad y la precisión son los mejores para los intereses de la compañía, empresa o cliente.

Por otro lado, y más allá de las cuestiones relacionadas con el empleo de una u otra técnica, se destaca en este trabajo que la obtención de resultados de calidad en las tareas de clasificación está íntimamente relacionada con el procesado de datos previo, la ordenación de la información ofrecida y la creación de nuevas variables.

6. Bibliografía.

Afrin, K. *et al.* (2018) 'Balanced Random Survival Forests for Extremely Unbalanced, Right Censored Data', *arXiv preprint arXiv:1803.09177*, pp. 1–27. Available at: http://arxiv.org/abs/1803.09177.

Aramburu La Torre, A. (2020) *Precisión diagnóstica de pruebas rápidas de detección de anticuerpos para Precisión diagnóstica de pruebas rápidas de detección de anticuerpos*. 01–2020.

Backiel, A. *et al.* (2015) 'Combining local and social network classifiers to improve churn prediction', *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2015*, pp. 651–658. doi: 10.1145/2808797.2808850.

Batista, G. E., Prati, R. C. and Monard, M. (2004) 'A Study of the Behavior of Several Methods for Balancing machine Learning.', *ACM SIGKDD Explorations Newsletter*, 6(1), pp. 20–29. doi: 10.1145/1007730.1007735.

Boyd, K., Eng, K. H. and Page, C. D. (2013) 'Area under the Precision-Recall Curve : Point Estimates and Confidence Intervals', in Blockeel, H. et al. (eds) *Machine Learning and Knowledge Discovery in Databases (European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III)*. Berlin: Springer-Verlag, pp. 451–466. doi: 10.1007/978-3-642-40994-3_29.

Burez, J. and Van den Poel, D. (2009) 'Handling class imbalance in customer churn prediction', *Expert Systems with Applications*. Elsevier Ltd, 36(3 PART 1), pp. 4626–4636. doi: 10.1016/j.eswa.2008.05.027.

De Caigny, A., Coussement, K. and De Bock, K. W. (2018) 'A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees', *European Journal of Operational Research*. Elsevier B.V., 269(2), pp. 760–772. doi: 10.1016/j.ejor.2018.02.009.

Chawla, N. V. *et al.* (2002) 'SMOTE: Synthetic Minority Over-sampling Technique Nitesh', *Journal of Artificial Intelligence Research*, 2009(Sept. 28), pp. 321–357. doi: 10.1613/jair.953.

Davis, J. and Goadrich, M. (2006) 'The Relationship Between Precision-Recall and ROC Curves', in Cohen, W. W. ; and W Moore, A. (eds) *Proceedings of the 23 rd International Conference on Machine Learning*. Pittsburgh.

Deng, W. *et al.* (2019) 'Sampling method based on improved C4.5 decision tree and its application in prediction of telecom customer churn', *International Journal of Information Technology and Management*, 18(1), pp. 93–109. doi: 10.1504/IJITM.2019.097887.

Dirick, L., Claeskens, G. and Baesens, B. (2017) 'Time to default in credit scoring using survival analysis: A benchmark study', *Journal of the Operational Research Society*. Palgrave Macmillan UK, 68(6), pp. 652–665. doi: 10.1057/s41274-016-0128-9.

Farquad, M. A. H., Ravi, V. and Raju, S. B. (2014) 'Churn prediction using comprehensible support vector machine: An analytical CRM application', *Applied Soft Computing Journal*. Elsevier B.V., 19, pp. 31–40. doi: 10.1016/j.asoc.2014.01.031.

Fernández-Navarro, F., Hervás-Martínez, C. and Antonio Gutiérrez, P. (2011) 'A dynamic over-sampling procedure based on sensitivity for multi-class problems', *Pattern Recognition*, 44(8), pp. 1821–1833. doi: 10.1016/j.patcog.2011.02.019.

Fernández, A. *et al.* (2018) *Learning from Imbalanced Data Sets*, *Learning from Imbalanced Data Sets*. Cham: Springer International Publishing. doi: 10.1007/978-3-319-98074-4.

Figini, S. and Maggi, M. (2014) 'Performance of credit risk prediction models via proper loss functions', *Department of Economics and Management Working Paper Series (Pavia University).* Pavia, 64(January).

Gordini, N. and Veglio, V. (2017) 'Customers churn prediction and marketing retention strategies. An application of support vector machines based on the AUC parameter-selection technique in B2B e-commerce industry', *Industrial Marketing Management*. Elsevier Inc., 62, pp. 100–107. doi: 10.1016/j.indmarman.2016.08.003.

Hand, D. J. (2012) 'Assessing the Performance of Classification Methods', *International Statistical Review*, 80(3), pp. 400–414. doi: 10.1111/j.1751-5823.2012.00183.x.

Hart, P. E. (1968) 'The condensed nearest neighbor rule', *IEEE Transactions on*

Information Theory, Mayo, pp. 515–517. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.294.6968&rep=rep1&type=pdf.

He, H. *et al.* (2008) 'ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning', in *IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. Hong Kong, pp. 1322–1328. doi: 10.1109/IJCNN.2008.4633969.

He, H. and Ma, Y. (2013) *Imbalanced Learning. Foundations, Algorithms, and Applications*. Edited by H. He and Y. Ma. Wiley. doi: 10.1002/9781118646106.

Höppner, S. *et al.* (2020) 'Profit driven decision trees for churn prediction', *European Journal of Operational Research*, 284(3), pp. 920–933. doi: 10.1016/j.ejor.2018.11.072.

Jain, H., Khunteta, A. and Srivastava, S. (2020) 'Churn Prediction in Telecommunication using Logistic Regression and Logit Boost', *Procedia Computer Science*. Elsevier B.V., 167(2019), pp. 101–112. doi: 10.1016/j.procs.2020.03.187.

Japkowicz, N. (2000) 'The Class Imbalance Problem : Signi cance and Strategies', in *Proceedings of the 2000 International Conference on Artificial Intelligence (ICAI)*. Las Vegas, pp. 111–117.

Kumar, S. and Kumar, M. (2019) 'Predicting Customer Churn Using Artificial Neural Network', in *Engineering Applications of Neural Networks*. Springer International Publishing, pp. 299–305. doi: 10.1007/978-3-030-20257-6.

Kvamme, H., Borgan, Ø. and Scheel, I. (2019) 'Time-to-event prediction with neural networks and cox regression', *Journal of Machine Learning Research*, 20, pp. 1–30.

Lin, C. S., Tzeng, G. H. and Chin, Y. C. (2011) 'Combined rough set theory and flow network graph to predict customer churn in credit card accounts', *Expert Systems with Applications*. Elsevier Ltd, 38(1), pp. 8–15. doi: 10.1016/j.eswa.2010.05.039.

Loyola-González, O., Martínez-trinidad, J. F. and Carrasco-ochoa, J. A. (2016) 'Study of the impact of resampling methods for contrast pattern based classi fi ers in imbalanced databases', *Neurocomputing*. Elsevier, 175(Part B), pp. 935–947. doi: 10.1016/j.neucom.2015.04.120.

Lu, N. *et al.* (2014) 'A customer churn prediction model in telecom industry using boosting', *IEEE Transactions on Industrial Informatics*, 10(2), pp. 1659–1665. doi: 10.1109/TII.2012.2224355.

Maldonado, S., López, J. and Vairetti, C. (2020) 'Profit-based churn prediction based on Minimax Probability Machines', *European Journal of Operational Research*, 284(1), pp. 273–284. doi: 10.1016/j.ejor.2019.12.007.

Mitrović, S. *et al.* (2017) 'Churn Prediction Using Dynamic RFM-Augmented Node2vec', in Guidotti, R. et al. (eds) *First International Workshop, PAP 2017 . Personal Analytics and Privacy. An Individual and Collective Perspective*. Skopje, pp. 122–138. doi: 10.1007/978-3-319-71970-2_11.

Mitrović, S. *et al.* (2019) 'tcc2vec: RFM-informed representation learning on call graphs for churn prediction', *Information Sciences*. Elsevier Inc., (xxxx). doi: 10.1016/j.ins.2019.02.044.

Nguyen, G. H., Bouzerdoum, A. and Phung, S. L. (2009) 'Learning Pattern Classification Tasks with Imbalanced Data Sets', in Yin, P.-Y. (ed.) *Pattern Recognition*. IntechOpen, pp. 193–208. doi: 10.5772/7544.

Salve, A. De, Mori, P. and Ricci, L. (2017) 'Personal Analytics and Privacy. An Individual and Collective Perspective', in Guidotti, R. et al. (eds) *First International Workshop, PAP 2017 . Personal Analytics and Privacy. An Individual and Collective Perspective*. Skopje, pp. 51–63. doi: 10.1007/978-3-319-71970-2.

Sivasankar, E. and Vijaya, J. (2019) 'Hybrid PPFCM-ANN model: an efficient system for customer churn prediction through probabilistic possibilistic fuzzy clustering and artificial neural network', *Neural Computing and Applications*. Springer London, 31(11), pp. 7181–7200. doi: 10.1007/s00521-018-3548-4.

Sun, Y., Wong, A. and Kamel, M. S. (2009) 'Classification of imbalanced data : a review CLASSIFICATION OF IMBALANCED DATA : A REVIEW', *International Journal of Pattern Recognition and Artificial Intelligence*, 23(4), pp. 687–717. doi: 10.1142/S0218001409007326.

Thai-Nghe, N., Gantner, Z. and Schmidt-Thieme, L. (2010) 'Cost-sensitive learning methods for imbalanced data', *Proceedings of the International Joint Conference on Neural Networks*. doi: 10.1109/IJCNN.2010.5596486.

Tomek, I. (1976a) 'An Experiment with the Edited Nearest-Neighbor Rule', *IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS.*, 1976(Junio), pp. 448–452.

Tomek, I. (1976b) 'Two modifications of CNN', *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, Noviembre, pp. 769–772.

Ullah, I. *et al.* (2019) 'A Churn Prediction Model Using Random Forest: Analysis of Machine Learning Techniques for Churn Prediction and Factor Identification in Telecom Sector', *IEEE Access*. IEEE, 7, pp. 60134–60149. doi: 10.1109/ACCESS.2019.2914999.

Verbeke, W. *et al.* (2012) 'New insights into churn prediction in the telecommunication sector : A profit driven data mining approach New insights into churn prediction in the telecommunication sector : A profit driven data mining approach', *European Journal of Operational Research*. Elsevier B.V., 218(1), pp. 211–229. doi: 10.1016/j.ejor.2011.09.031.

Wilson, D. L. (1972) 'Asymptotic Properties of Nearest Neighbor Rules Using Edited Data', *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, 2(Julio), pp. 408–421.

Wong, K. K. K. (2011) 'Using Cox regression to model customer time to churn in the wireless telecommunications industry', *Journal of Targeting, Measurement and Analysis for Marketing*, 19(1), pp. 37–43. doi: 10.1057/jt.2011.1.

Zang, J. and Mani, I. (2003) *Knn approach to unbalanced data distributions: a case study involving information extraction*. 2. Washington.

Zhang, D. *et al.* (2019) 'Imbalanced Data Classification Algorithm Based on Integrated Sampling and Ensemble Learning', in Pan, J.-S. et al. (eds) *ICGEC 2018: Genetic and Evolutionary Computing*. Springer Singapore, pp. 461–470. doi: 10.1007/978-981-13-5841-8.

Zhu, B. *et al.* (2018) 'Benchmarking sampling techniques for imbalance learning in churn prediction', *Journal of the Operational Research Society*, 69(1), pp. 49–65. doi: 10.1057/s41274-016-0176-1.

Zhu, B., Baesens, B. and vanden Broucke, S. K. L. M. (2017) 'An empirical comparison of techniques for the class imbalance problem in churn prediction', *Information Sciences*. Elsevier Inc., 408(June), pp. 84–99. doi: 10.1016/j.ins.2017.04.015.

7. Anexos.

Anexo1: Índice de figuras y tablas incluidas en el trabajo.

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

Anexo 2: Código *Pipeline 1 - Construcción* conjunto de datos.

```
import pandas as pd
import numpy as np
import glob
import time

def read_file(data_path, table_name):
    # Lee y carga los datos en memoria

    t1 = time.time()

    files = glob.glob(data_path + "*.csv")
    file_name = [x for x in files if x.endswith(table_name + ".csv")][0]
    df = pd.read_csv(file_name)

    t2 = time.time()
    dif = round(t2-t1, 2)
    unit = "seconds"
    if(dif > 60):
        unit = "minutes"
        dif = round(dif/60,2)

    print("The dataset " + table_name + " has " + str(df.shape[0]) + " rows and " + str(df.shape[1]) + " columns")
    print("Execution time: " + str(dif) + " " + unit)
    return df


# Agregación de la información de uso de user_logs

def user_log_aggregation(df):

    """ Aggregation of user_log information. Summary of usage information for each subscriber during the last observed month.

    Parameters:
    df (pandas.dataframe): a pandas dataframe with the usage information day by day.
```

```
    Returns:
    pandas.dataframe: Pandas dataframe with the usage summary for each subscriber. """

    t1 = time.time()
    df_agg = df.groupby('msno').agg(num_25 = pd.NamedAgg('num_25','sum'),
                                                num_50 = pd.NamedAgg('num_50','sum'),
                                                num_75 = pd.NamedAgg('num_75','sum'),
                                                num_985 = pd.NamedAgg('num_985','sum'),
                                                num_100 = pd.NamedAgg('num_100','sum'),
                                                num_unq = pd.NamedAgg('num_unq','sum'),
                                                total_secs = pd.NamedAgg('total_secs','sum'),
                                                n_activity = pd.NamedAgg('date','count')).reset_index()
    t2 = time.time()
    dif = round(t2-t1, 2)
    unit = "seconds"
    if(dif > 60):
        unit = "minutes"
        dif = round(dif/60,2)

    print("The new dataset 'user_logs_grouped' has " + str(df_agg.shape[0]) + " rows and " + str(df_agg.shape[1]) + " columns")
    print("Execution time: " + str(dif) + " " + unit)
    return df_agg


# Ordenación y eliminación de filas de "transactions".

def transactions_clean(df, df_train):

    """ Select only those transactions from users included in the churn study. Remove useless transactions and
    sorts the dataset.

    Parameters:
    df (pandas.dataframe): a raw pandas dataframe with the user transactions.

    Returns:
    pandas.dataframe: Pandas dataframe clean and sorted. """

    t1 = time.time()
```

```
    # Rows selection
    print("Selecting study rows...")
    df_clean = df.drop(['Unnamed: 0'],axis=1)
    df_clean = df_clean[df_clean['msno'].isin(df_train['msno'])]

    # Setting dates
    print("Setting dates...")
    df_clean['transaction_date'] = pd.to_datetime(df_clean['transaction_date'].astype(str), format='%Y%m%d')
    df_clean['membership_expire_date'] = pd.to_datetime(df_clean['membership_expire_date'].astype(str), format='%Y%m%d')

    # Sort df by msno, transaction date & membership expiration date
    print("Sorting by users ids and transactions dates...")
    df_clean = df_clean.sort_values(['msno', 'transaction_date', 'membership_expire_date'])

    # Removing useless transactions
    print("Removing useless transactions...")
    index = df_clean[(df_clean['payment_plan_days'] == 0) & (df_clean['plan_list_price'] == 0)].index
    df_clean.drop(index, inplace = True)
    df_clean = df_clean.groupby(['msno','transaction_date']).tail(1).reset_index()
    df_clean.drop(['index'],axis = 1)

    t2 = time.time()
    dif = round(t2-t1, 2)
    unit = "seconds"
    if(dif > 60):
        unit = "minutes"
        dif = round(dif/60,2)

    print("The clean and sort dataset has " + str(df_clean.shape[0]) + " rows and " + str(df_clean.shape[1]) + " columns.")
    print("Execution time: " + str(dif) + " " + unit)
    return df_clean


# New variables creation for Churn study

def transactions_setup (df_clean):

    """ Creates new features for the churn study. Past churns, churns dates, renew dates, discounts,
```

```
    days between churn and renew dates, real expiration dates.

    Parameters:
    df_clean (pandas.dataframe): a clean and sorted pandas dataframe with the valid transactions.

    Returns:
    pandas.dataframe: Pandas dataframe with the new features created. """

    t1 = time.time()

    # Times between consecutive transactions
    print("Calculating time between transactions...")
    df_trans = (df_clean.assign(next_trans_date=(lambda x: x.groupby('msno')['transaction_date'].shift(-1))))
    df_trans['diff_tr'] = df_trans['next_trans_date'] - df_trans['transaction_date']
    df_trans['diff_tr'] = pd.to_numeric(df_trans['diff_tr'].dt.days, downcast='integer')

    # Past churns detection
    print("Detecting past churns...")
    df_trans['churn_past'] = '0'
    df_trans.loc[df_trans['diff_tr'] > df_trans['payment_plan_days'] + 30, 'churn_past'] = 1
    df_trans.loc[df_trans['diff_tr'] < df_trans['payment_plan_days'] + 30, 'churn_past'] = 0

    # Set churn past dates
    print("Giving dates to the past churns...")
    df_trans['churn_past_date'] = float('NaN')
    df_trans.loc[df_trans['churn_past'] == 1, 'churn_past_date'] = df_trans['transaction_date']
    df_trans['churn_past_date'] =  pd.to_datetime(df_trans['churn_past_date']) +
pd.to_timedelta(df_trans['payment_plan_days'],unit='d')

    df_trans = df_trans.drop(['next_trans_date', 'diff_tr'], axis = 1)
    df_trans['churn_past'] = df_trans['churn_past'].astype(str).astype(int)

    # Set renew dates before churn
    print("Giving dates to the renew services...")
    df_trans['renew_date'] = float('NaN')
    df_trans.loc[df_trans['churn_past'].shift(1) == 1, 'renew_date'] = df_trans['transaction_date']
    df_trans['renew_date'] =  pd.to_datetime(df_trans['renew_date'])

    # Set real expiration dates
```

```
    print("Calculating real expiration dates...")
    df_trans['expire_plan_date'] = df_trans['transaction_date'] + pd.to_timedelta(df_trans['payment_plan_days'],unit='d')

    # Discounts calculation
    print("Calculating discounts...")
    df_trans['discount'] = df_trans['plan_list_price'] - df_trans['actual_amount_paid']

    # Days between churn and following renew
    print("Calculating past suscription days...")
    df_trans['past_days'] = df_trans['churn_past_date'] - df_trans['renew_date']

    t2 = time.time()
    dif = round(t2-t1, 2)
    unit = "seconds"
    if(dif > 60):
        unit = "minutes"
        dif = round(dif/60,2)

    print("The new dataset has " + str(df_trans.shape[0]) + " rows and " + str(df_trans.shape[1]) + " columns.")
    print("Execution time: " + str(dif) + " " + unit)
    return df_trans


# Transactions aggregations with integrated functionsclean

def transactions_int_aggs (df_trans):

    """ Aggregation of transactions information. Summary of user transactions with Python integrated functions.
    Gives aggregated information of 14 variables to each user.

    Parameters:
    df_trans (pandas.dataframe): transactions dataframe with new features created.

    Returns:
    pandas.dataframe: 14 columns Pandas dataframe with transactions summary for each subscriber. """

    print("Executing aggregations with integrated functions...")

    t1 = time.time()
```

```
    df_agg_1 = df_trans.groupby(['msno']).agg(total_payment_days = pd.NamedAgg('payment_plan_days','sum'),
                                              total_amount_paid = pd.NamedAgg('actual_amount_paid','sum'),
                                              n_auto_renew = pd.NamedAgg('is_auto_renew','sum'),
                                              n_cancel = pd.NamedAgg('is_cancel','sum'),
                                              n_churn_past = pd.NamedAgg('churn_past','sum'),
                                              discount_amount = pd.NamedAgg('discount','sum'),
                                              last_plan_paid = pd.NamedAgg('payment_plan_days',lambda x: x.iloc[-1]),
                                              last_payment_method = pd.NamedAgg('payment_method_id',lambda x: x.iloc[-1]),
                                              last_is_cancel = pd.NamedAgg('is_cancel',lambda x: x.iloc[-1]),
                                              first_transaction = pd.NamedAgg('transaction_date','min'),
                                              expire_plan_date = pd.NamedAgg('expire_plan_date','max'),
                                              membership_expire_date = pd.NamedAgg('membership_expire_date','max'),
                                              last_renew_date = pd.NamedAgg('renew_date','max'),
                                              n_transactions = pd.NamedAgg('membership_expire_date','count')).reset_index()

    t2 = time.time()
    dif = round(t2-t1, 2)
    unit = "seconds"
    if(dif > 60):
        unit = "minutes"
        dif = round(dif/60,2)

    print("Aggregation complete.")
    print("Execution time: " + str(dif) + " " + unit)
    return df_agg_1


# CUSTOM FUNCTIONS FOR AGGREGATIONS.

# Number of plan changes function.
def n_changes(serie):

    """ Gives the number of times a user changes the subscription plan.

    Parameters:
    serie (pandas.dataframe): a list with the consecutive plans in each transaction.

    Returns:
```

```
    integer: number of plan changes. """

    return serie.rolling(2).apply(lambda x: x[0] != x[-1], raw=True).sum().astype(int)

# Number in past subscriptions days.
def n_past_days(df):

    """ Gives the number of days a user was a subscriptor before the actual subscription period.

    Parameters:
    df_trans (pandas.dataframe): transactions dataframe.

    Returns:
    timedelta object: number of days of past subscriptions. """

    return (df.churn_past_date.min() - df.transaction_date.min()) + df.past_days.sum()


# Transactions aggregations with custom functions

def transactions_custom_aggs (df_trans):

    """ Aggregation of transactions information. Summary of user transactions with custom created functions.
    Gives aggregated information of 3 variables to each user.

    Parameters:
    df_trans (pandas.dataframe): transactions dataframe with new features created.

    Returns:
    pandas.dataframe: 4 columns Pandas dataframe with transactions summary for each subscriber. """

    print("Executing aggregations with custom functions...")

    t1 = time.time()

    print('Grouping data...')
    df_grouped = df_trans.groupby(['msno'])

    print('Aggregating data...')
```

```
    # Plan changes
    df_changes = df_grouped.agg(n_plan_changes = pd.NamedAgg('plan_list_price', n_changes)).reset_index()
    print('Number of plan changes complete.')

    # Subscription past days
    df_past = df_grouped.apply(n_past_days).reset_index(name = "past_subscript_days")
    print('Past subscription days complete.')

    # Most common plan
    df_plan = df_grouped.agg(usual_plan = pd.NamedAgg('payment_plan_days', lambda x:x.value_counts().index[0])).reset_index()
    print('Most frequent plan complete.')

    # Merging grouped variables "msno"
    print('Joining columns...')
    df_agg_2 = df_changes.merge(df_past,on='msno').merge(df_plan,on='msno')

    t2 = time.time()
    dif = round(t2-t1, 2)
    unit = "seconds"
    if(dif > 60):
        unit = "minutes"
        dif = round(dif/60,2)

    print("Aggregation complete.")
    print("Execution time: " + str(dif) + " " + unit)
    return df_agg_2


# Dataset preparation. KKBOX dataset final version.

def kkbox_dataset_setup (df_merged, date_end):

    """ Prepares and cleans the merged dataset. Fix wrong expiration dates, calculates life times,
    removes date & time columns and solves NaN conflicts

    Parameters:
    df_merged (pandas.dataframe): merged dataset with user logs, transactions, churn and members information included.
```

```
    Returns:
    pandas.dataframe: Clean and prepared dataset for data modelling. """


    date_end = pd.to_datetime(date_end)

    # Init actual subscription dates extraction
    print("Setting dates...")

    df_merged['actual_transaction_date'] = ''
    df_merged.loc[df_merged.last_renew_date.isnull(), 'actual_transaction_date'] = df_merged['first_transaction']
    df_merged.loc[df_merged.last_renew_date.notnull(), 'actual_transaction_date'] = df_merged['last_renew_date']
    df_merged['actual_transaction_date'] =  pd.to_datetime(df_merged['actual_transaction_date'])
    df_merged = df_merged.drop(['last_renew_date'],axis=1)

    # Setting "registration_init_time"

    df_merged['registration_init_time'] = df_merged['registration_init_time'].astype(pd.Int64Dtype())
    df_merged['registration_init_time'] = df_merged['registration_init_time'].astype(str)
    df_merged['registration_init_time'] = pd.to_datetime(df_merged['registration_init_time'], format='%Y%m%d', errors = 'coerce')

    # Change fake NaN values with 0.
    # No usage during the last month (not NA, usage = 0).
    print("Removing fake NAs in service usage...")

    df_merged[['num_25', 'num_50', 'num_75', 'num_985', 'num_100', 'num_unq', 'total_secs', 'n_activity']] = df_merged[['num_25',
'num_50', 'num_75', 'num_985', 'num_100', 'num_unq', 'total_secs', 'n_activity']].fillna(0)

    # Setting real expiration dates.
    print("Setting expiration dates...")

    df_merged.loc[df_merged['expire_plan_date'] < df_merged.membership_expire_date.min(), 'expire_plan_date'] =
    df_merged['membership_expire_date']
    df_merged.loc[df_merged['membership_expire_date'] > df_merged.expire_plan_date.max(), 'expire_plan_date'] =
    df_merged['expire_plan_date']

    # Life times calculations:
    print("Calculating life times...")
```

```
    # Days from registration.
    df_merged['days_registration'] = (date_end - df_merged['registration_init_time']).dt.days
    # Days from first transaction.
    df_merged['days_first_trans'] = (date_end - df_merged['first_transaction']).dt.days
    # Days from starting actual valid transaction to expiration (life time) 'tenure_expiration'.
    df_merged['tenure_expiration'] = (df_merged['expire_plan_date'] - df_merged['actual_transaction_date']).dt.days
    # Days from starting actual valid transaction to end of observation date (life time) 'ternure_end_obs'.
    df_merged['ternure_end_obs'] = (date_end - df_merged['actual_transaction_date']).dt.days

    # Columns preparation
    print("Removing date columns and giving numeric formats...")

    # Subscriptions past days if NaN = 0. No subscription past days.
    df_merged['past_subscript_days'] = (df_merged['past_subscript_days']).dt.days
    df_merged['past_subscript_days'] = df_merged['past_subscript_days'].fillna(0)

    df_merged = df_merged.select_dtypes(exclude=['datetime64'])

    # Solving NaN problems in columns.
    print("Removing columns over 30% of NaN...")

    cols_to_drop = []
    for var in df_merged:
        if (df_merged[var].dropna().shape[0]/df_merged[var].shape[0]) < 0.7:
            cols_to_drop.append(var)

    df_final = df_merged.drop(cols_to_drop,axis=1)


    print("_____________________________________________________________\n\nTHE KKBOX DATASET HAS BEEN GENERATED.\nThe final
dataset has: " +
          str(df_final.shape[0]) + " rows and " + str(df_final.shape[1]) + " columns.")
    return df_final


import time
import src.data.pipeline1_aux as p1
```

```
def kkbox_df_build (df_train, df_uslg, df_trans, df_memb, date_end):
    ti = time.time()

    # user_logs aggregations:
    df_uslg = p1.user_log_aggregation(df_uslg)
    print("AGGREGATIONS IN 'user_logs' COMPLETED\n")

    # transactions clean and sort
    df_trans = p1.transactions_clean(df_trans, df_train)

    # transactions setup
    df_trans = p1.transactions_setup(df_trans)
    print("TRANSFORMATIONS IN 'transactions' COMPLETED\n")

    # transactions aggregations
    df_agg_1 = p1.transactions_int_aggs (df_trans_setup)
    df_agg_2 = p1.transactions_custom_aggs (df_trans_setup)

    df_trans_agg = df_agg_1.merge(df_agg_2,on='msno')
    print("AGGREGATIONS IN 'transactions' COMPLETED\n")

    del df_agg_1
    del df_agg_2

    # merge dataset
    df_merged = df_train.merge(df_trans_agg, how='left', on = 'msno').merge(df_uslg_gr, how='left', on = 'msno').merge(df_memb,
how='left', on = 'msno')
    print("DATASETS MERGER COMPLETED\n")

    del df_trans_agg

    # kkbox final dataset preparation
    df_kkbox_final = p1.kkbox_dataset_setup (df_merged, date_end)

    tf = time.time()
    dif = round(tf-ti, 2)
    unit = "seconds"
    if(dif > 60):
        unit = "minutes"
```

```
        dif = round(dif/60,2)

    print("TOTAL EXECUTION TIME: " + str(dif) + " " + unit)

    return df_kkbox_final
```

Anexo 3: Código *Pipeline 2* – Preparación del conjunto de datos.

```
import pandas as pd
import numpy as np


# Fórmula para la conversión de variables numéricas a categóricas.

def num_to_cat(df):
    """ Converts numerical columns to categorical.

    Parameters:
    df: the dataframe with the columns to convert
    a
    Returns:
    dataframe with the correct formats """

    vars_to_cat = ['last_plan_paid', 'usual_plan', 'last_payment_method', 'city', 'registered_via']

    for c in vars_to_cat:
        df[c] = df[c].astype('object')


    print (len(vars_to_cat), "columns have been converted to categorical.")
    return df


# Eliminación de variables con un alto número de categorías.

def drop_categorical(df, threshold = 0.8):
```

```
    """ Removes columns with a high number of categories over the dimension of the dataset.

    Parameters:
    df (pandas.dataframe): The entire KKBox dataframe.
    threshold: weight of the number of categories over the dimension of the dataset.

    Returns:
    df: dropped dataset """

    cat_cols = df.select_dtypes('object').columns
    cat_cols = cat_cols[cat_cols != 'msno']

    vars_to_drop = []
    for c in cat_cols:
        nlevels = len(df[c].unique())
        if nlevels >= threshold*df.shape[0]:
            vars_to_drop.append(c)

    if len(vars_to_drop) > 0:
        df = df.drop(vars_to_drop, axis = 1)

    print(len(vars_to_drop), "columns have been dropped")
    return df


# Función que agrupa categorías de baja aparición en variables categóricas.

def group_categories(df, threshold = 0.85):

    """ Groups low occurrence categories in a single one in the categorical columns.

    Parameters:
    df (pandas.dataframe): The entire KKBox dataframe..
    threshold: cumulative percentage value over which, the categories will be grouped

    Returns:
    df: pandas dataframe with grouped categories in categorical columns """

    cat_cols = df.select_dtypes('object').columns
```

```
    cat_cols = cat_cols[cat_cols != 'msno']

    for c in cat_cols:

        df_levels = df[c].value_counts().rename_axis('unique_values').reset_index(name='counts')
        df_levels['cum_counts'] = df_levels['counts'].cumsum()
        df_levels['cum_perc'] = round(df_levels['cum_counts']/sum(df_levels['counts']), 2)

        df_levels_to_recode = df_levels[df_levels['cum_perc'] >= threshold]
        levels_to_recode = df_levels_to_recode['unique_values'][1:df_levels_to_recode.shape[0]]

        df[c] = np.where(df[c].isin(levels_to_recode), "other", df[c])

        print(len(levels_to_recode), "levels have been grouped for the variable", c)
        print("Final levels: ", df[c].unique(), "\n------------")

    return df


# Función que aborda la codificación One Hot.

def one_hot_encoding(df):

    """Solves the One Hot Encoding for categorical columns inclueded in the KKBox dataset.

    Parameters:
    df(pandas.dataframe): The entire KKBox dataframe with low occurrence categories transformations.

    Returns:
    pandas.dataframe: the new one hot encoded dataframe. """

    cols = list(df.loc[:,'total_payment_days':].select_dtypes(include=['object']))
    df_ohe = pd.get_dummies(data = df, columns = cols)

    print ("One Hot encoding has been done. The dataset has increased from",df.shape[1], "to", df_ohe.shape[1], "columns.")
    return df_ohe


# Función que elimina variables con una baja dispersión.
```

```
def drop_low_variation(df, threshold = 0.05):

    """ Drops low dispersion variables from de dataset.

    Parameters:

    df (pandas.dataframe): kkbox final dataset.
    threshold: standard deviation ponderated by mean (std(x)/mean(x)) over which, variables will be dropped.

    Returns:
    pandas.dataframe: the new droped dataframe. """

    df_desc = df.describe().loc['std'].rename_axis('variable').reset_index(name='std')
    df_desc['mean'] = df.describe().loc['mean'].to_list()

    df_desc['cv'] = round(df_desc['std']/df_desc['mean'], 2)
    vars_to_drop = df_desc[df_desc['cv'] <= threshold]['variable']

    df = df.drop(vars_to_drop, axis = 1)

    print(len(vars_to_drop), "variables have been removed due to low dispersion.")

    return df

import src.data.pipeline2_aux as p2
import pandas as pd

def kkbox_df_preparation (df, threshold_drop_cat = 0.8 , threshold_group_cat = 0.85, threshold_low_var = 0.05):

    """ Returns the preparated dataframe to implement models. The pipeline removes columns with high number of categories,
    groups low occurrence categories in a single one, does the one hot encoding and removes low variation columns.

    Parameters:

    df (pandas.dataframe): kkbox final dataset (from interim folder).
    threshold_drop_cat: weight of the number of categories over the dimension of the dataset (0.8 if is omitted).
    threshold_group_cat: cumulative percentage value over which, the categories will be grouped (0.85 if is omitted).
```

```
    threshold_low_var: standard deviation ponderated by mean (std(x)/mean(x)) over which, variables will be dropped (0.05 if is
omitted).

    Returns:

    df: KKBox pandas dataframe prepared for modelling. """

    pd.options.mode.chained_assignment = None

    # Drop NaN
    df = df.dropna()

    # Categorical columns conversion
    df = p2.num_to_cat(df)

    # Drop categrical columns with a high number of categories
    df = p2.drop_categorical(df, threshold_drop_cat)

    # Group low occurrence categories
    df = p2.group_categories(df, threshold_group_cat)

    # Solve the one hot encoding.
    df = p2.one_hot_encoding(df)

    # Drop low variation columns
    df = p2.drop_low_variation(df, threshold_low_var)

    print("The KKBox dataset is prepared for modelling.\n---------------")
    print("The new dataset has " + str(df.shape[0]) + " rows and " + str(df.shape[1]) + " columns.")


    return df
```

Anexo 4: Código para la extracción de resultados.

```
# Models
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier

# Resampling techniques
from imblearn.under_sampling import RandomUnderSampler
from imblearn.under_sampling import CondensedNearestNeighbour
from imblearn.under_sampling import EditedNearestNeighbours
from imblearn.under_sampling import RepeatedEditedNearestNeighbours
from imblearn.under_sampling import TomekLinks
from imblearn.under_sampling import NearMiss
from imblearn.over_sampling import SMOTE
from imblearn.over_sampling import ADASYN
from imblearn.over_sampling import RandomOverSampler
from imblearn.combine import SMOTETomek

# Performance metrics
from sklearn.metrics import precision_recall_fscore_support
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import plot_roc_curve
from sklearn.metrics import log_loss
from sklearn.metrics import auc
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import average_precision_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

# Visualization
import matplotlib.pyplot as plt
import seaborn as sns
import random
import itertools
```

```
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt
from itertools import cycle
lines = ["-","--","-.",":"]
linecycler = cycle(lines)

# Data split TRAIN & TEST
y = kkbox_final.is_churn
X = kkbox_final.drop(['msno','is_churn'], axis = 1)
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.2)

# Definición de las técnicas de resampling
rus = RandomUnderSampler()
enn = EditedNearestNeighbours()
tml = TomekLinks()
nm3 = NearMiss(version = 3)
ros = RandomOverSampler()
smo = SMOTE()
ada = ADASYN()
smt = SMOTETomek(random_state=semilla_resampling)

# algorithms list
sampler = [rus, enn, tml, nm3, ros, smo, ada, smt]
# algorithms labels list
alias = ['rus', 'enn', 'tml', 'nm3', 'ros', 'smot', 'adas', 'smo_tl']
# algorithms names list
method = ['Random Under Sampling', 'Edited nearest neighbours' , 'Tomek Links', 'Near Miss 3', 'Random Over Sampling', 'SMOTE',
'ADASYN', 'SMOTE&TomekLinks']
# methods dataframe
method_df = pd.DataFrame({'method_name':method, 'alias':alias,'sampler':sampler})


def classification_results (X_train,X_test,y_train,y_test, classifier):

    '''Gives the performance results of a logistic regression after applying different data balancing methods.

    Parameters:
    normalize = 'true'
    X_train: splnormalize = 'true'it data containing the predictors from the training set.
```

```
    X_test: split data containing the predictors from the test set.
    y_train: split data containing variable to predict from the training set.
    y_test: split data containing variable to predict from the training set.
    classifier: classification method.

    Returns:
    pandas.dataframe: Pandas dataframe with the ordered results of the implementations.'''


    # Results dataframe
    performance_metrics_df = pd.DataFrame(columns=['Method', 'Accuracy', 'Precision', 'Recall', 'F1-score', 'AUC','AP', 'Log-
loss'])

    # ROC curves results dataframe
    result_table = pd.DataFrame(columns=['method', 'fpr','tpr','auc'])

    # PR Curves results dataframe
    result_table_prc = pd.DataFrame(columns=['method', 'precision','recall','auc_pr'])

    # Base model
    base_model = classifier.fit(X_train, y_train)

    # Base model predictions
    y_pred_train_base = base_model.predict(X_train)
    y_pred_train_proba_base = base_model.predict_proba(X_train)[::,1]

    y_pred_test_base = base_model.predict(X_test)
    y_pred_test_proba_base = base_model.predict_proba(X_test)[::,1]

    # Base model metrics
    technique_name_base = classifier_label + '_base'
    metrics_base = []
    metrics_base.append(accuracy_score(y_pred_test_base, y_test)) # accuracy
    metrics_base = metrics_base + list(precision_recall_fscore_support(y_pred_test_base, y_test, average='binary')[0:3]) #
precision, recall & F1
    metrics_base.append(roc_auc_score(y_test, y_pred_test_proba_base)) # AUC
    metrics_base.append(average_precision_score (y_test, y_pred_test_proba_base, average = 'samples')) # Average Precision
    metrics_base.append(log_loss(y_test,y_pred_test_proba_base)) # LogLoss
    metrics_base = [round(num, 5) for num in metrics_base]
```

```
    metrics_base.insert(0,technique_name_base)
    metrics_base = pd.Series(metrics_base, index = performance_metrics_df.columns)
    performance_metrics_df = performance_metrics_df.append(metrics_base,ignore_index=True)

    # Base model confusion matrix.
    cm_train_base = confusion_matrix(y_train, y_pred_train_base, normalize = 'true')
    cm_test_base = confusion_matrix(y_test, y_pred_test_base, normalize = 'true')

    fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(10,4))
    sns.heatmap(np.round(cm_train_base,2), annot=True, fmt = 'g',cmap = plt.cm.Greys,ax=axes[0]).set_title('ENTRENAMIENTO. '+
classifier_label +'_base', fontweight = 'bold')
    sns.heatmap(np.round(cm_test_base,2), annot=True, fmt = 'g',cmap = plt.cm.Greys,ax=axes[1]).set_title('TEST. '+
classifier_label +'_base', fontweight = 'bold')

    print(classifier_label +'_base' + ' metrics extracted...')


    # RESAMPLING METRICS EXTRACTION

    for alias, sampler in zip(method_df.iloc[:, 1], method_df.iloc[:, 2]):
        # Resampling
        X_train_rs, y_train_rs = sampler.fit_sample(X_train, y_train)

        # Modelo
        model = classifier.fit(X_train_rs, y_train_rs)

        # Predicciones de clase y probabilísticas
        y_pred_train = model.predict(X_train)
        y_pred_train_proba = model.predict_proba(X_train)[::,1]

        y_pred_test = model.predict(X_test)
        y_pred_test_proba = model.predict_proba(X_test)[::,1]

        #Insertar resultados
        technique_name = classifier_label +'_'+ alias
        metrics = []
        metrics.append(accuracy_score(y_pred_test, y_test)) # accuracy
        metrics = metrics + list(precision_recall_fscore_support(y_pred_test, y_test, average='binary')[0:3]) # precision, recall
& F1
```

```
        metrics.append(roc_auc_score(y_test, y_pred_test_proba)) # AUC
        metrics.append(average_precision_score (y_test, y_pred_test_proba, average = 'samples')) # Average Precision
        metrics.append(log_loss(y_test,y_pred_test_proba)) # LogLoss
        metrics = [round(num, 5) for num in metrics]
        metrics.insert(0,technique_name)
        metrics = pd.Series(metrics, index = performance_metrics_df.columns)
        performance_metrics_df = performance_metrics_df.append(metrics,ignore_index=True)

        # Confusion matrixes.
        cm_train = confusion_matrix(y_train, y_pred_train, normalize = 'true')
        cm_test = confusion_matrix(y_test, y_pred_test, normalize = 'true')

        fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(10,4))
        sns.heatmap(np.round(cm_train,2), annot=True, fmt = 'g',cmap = plt.cm.Greys,ax=axes[0]).set_title('ENTRENAMIENTO. '+
classifier_label +'_'+ alias, fontweight = 'bold')
        sns.heatmap(np.round(cm_test,2), annot=True, fmt = 'g',cmap = plt.cm.Greys,ax=axes[1]).set_title('TEST. '+
classifier_label +'_'+ alias, fontweight = 'bold')

        # ROC curves dataframes for visualizations
        fpr, tpr, _ = roc_curve(y_test, y_pred_test_proba)
        auc = roc_auc_score(y_test, y_pred_test_proba)

        result_table = result_table.append({'method': classifier_label +'_'+ alias,
                                            'fpr':fpr,
                                            'tpr':tpr,
                                            'auc':auc}, ignore_index=True)



        # PR curves dataframes for visualizations
        precision, recall, _ = precision_recall_curve(y_test, y_pred_test_proba)
        AP = average_precision_score (y_test, y_pred_test_proba, average = 'samples')

        result_table_prc = result_table_prc.append({'method': classifier_label +'_'+alias,
                                        'precision':precision,
                                        'recall':recall,
                                        'auc_pr': AP}, ignore_index=True)
```

```
        print(classifier_label +'_'+ alias + ' metrics extracted...')


    # ROC base model results.

    fpr, tpr, _ = roc_curve(y_test, y_pred_test_proba_base)
    auc = roc_auc_score(y_test, y_pred_test_proba_base)

    result_table = result_table.append({'method': classifier_label +'_base',
                                        'fpr':fpr,
                                        'tpr':tpr,
                                        'auc':auc}, ignore_index=True)

    result_table = result_table.sort_values(by=['auc'], ascending=False)

    # PR base model results.

    precision, recall, _ = precision_recall_curve(y_test, y_pred_test_proba_base)
    AP = average_precision_score(y_test, y_pred_test_proba_base, average = 'samples')

    result_table_prc = result_table_prc.append({'method': classifier_label +'_base',
                                        'precision':precision,
                                        'recall':recall,
                                        'auc_pr':AP}, ignore_index=True)

    result_table_prc = result_table_prc.sort_values(by=['auc_pr'], ascending=False)

    # ROC curves visualizations.

    fig = plt.figure(figsize=(12,10))

    for i, linestyle in zip(result_table.index, linecycler):
        plt.plot(result_table.loc[i]['fpr'],
                 result_table.loc[i]['tpr'],
                 linestyle=linestyle,
                 label="{}, AUC={:.3f}".format(result_table.loc[i]['method'], result_table.loc[i]['auc']))

    plt.plot([0,1], [0,1], color='grey', linestyle='--')
```

```
    plt.xticks(np.arange(0., 1.1, step=0.1))
    plt.xlabel("1-especificidad", fontsize=15)

    plt.yticks(np.arange(0., 1.1, step=0.1))
    plt.ylabel("Sensibilidad", fontsize=15)

    plt.title('Comparación de curvas ROC '+classifier_label, fontweight='bold', fontsize=15)
    plt.legend(prop={'size':13}, loc='lower right')

    plt.show()

    # ROC curves visualizations.

    fig = plt.figure(figsize=(12,10))

    for i, linestyle in zip(result_table_prc.index, linecycler):
        plt.plot(result_table_prc.loc[i]['precision'],
                 result_table_prc.loc[i]['recall'],
                 linestyle=linestyle,
                 label="{}, AP={:.3f}".format(result_table_prc.loc[i]['method'], result_table_prc.loc[i]['auc_pr']))


    plt.xticks(np.arange(0., 1.1, step=0.1))
    plt.xlabel("Recall", fontsize=15)

    plt.yticks(np.arange(0., 1.1, step=0.1))
    plt.ylabel("Precision", fontsize=15)

    plt.title('Comparación de curvas PR '+classifier_label, fontweight='bold', fontsize=15)
    plt.legend(prop={'size':13}, loc='lower left')

    return performance_metrics_df
```