CUNEF UNIVERSIDAD

MÁSTER EN CIENCIA DE DATOS

Trabajo Fin de MÁSTER

EFECTO DE LAS TÉCNICAS DE BALANCEO EN PROBLEMAS DE CLASIFICACIÓN DESEQUILIBRADOS

Over y Undersampling aplicado a churn en empresas de telecomunicación

Autor: Blanco García, Ángel

Tutora: Arévalo Barco, Irina

Madrid, junio de 2024

Índice

1. RESUMEN

El aprendizaje automático es uno de los pilares fundamentales de la Ciencia de Datos en la actualidad. Son cada vez más las organizaciones que lo utilizan como herramienta para extraer valor de los datos que poseen y poder elaborar planes de actuación comerciales en base a fundamentos respaldados por la estadística. Los modelos de machine learning ayudan a los analistas porque pueden procesar grandes cantidades de datos y detectar patrones complejos con mayor precisión y eficiencia que los métodos tradicionales. La implementación de estos modelos lleva a la automatización de tareas repetitivas y, dicha automatización mejora la eficiencia operativa porque reduce la posibilidad de errores humanos.

A menudo, esta disciplina se utiliza para identificar grupos de datos con características similares y así poder predecir, con información nueva, a que grupo pertenecerán dichos datos.

Sin embargo, el proceso de aplicación de estos modelos no es una tarea liviana puesto que, en muchos casos, la calidad de los datos disponibles no es suficientemente buena o existe una desigualdad entre la información que se posee sobre uno de los grupos con respecto al resto, lo que frecuentemente lleva a los modelos a sesgar su clasificación hacia el perfil más conocido. Es en estos casos en los que se recurre a la aplicación de técnicas de aumento de los datos de la clase infrarrepresentada o reducción de los de la clase mayoritaria para mejorar la precisión de estos modelos.

El objetivo de este trabajo es analizar los efectos de las diferentes técnicas de balanceo de clases sobre el latente problema del abandono de clientes y proporcionar una plantilla para todo aquel que desee seguir investigando o aportar otro enfoque al estudio.

Términos clave: undersampling, oversampling, churn, telecomunicaciones, machine learning, aprendizaje automático, abandono de clientes.

2. INTRODUCCIÓN

El churn, o abandono de clientes, es un fenómeno crítico que afecta a las empresas, particularmente a aquellas con modelos de negocio basados en suscripciones. Un estudio realizado por la empresa CustomerGauge sobre la retención y abandono de clientes a nivel global, estimó que las cifras de abandono promedio para empresas B2B es del 23%, mientras que para empresas B2C, de alrededor del 31%, en el año 2023 (CustomerGauge, 2024).

Si bien la fuga de clientes parece afectar de formas diferentes según la industria, los datos muestran que afecta a cualquier tipo de empresa, por lo que es un problema real para tener en cuenta por las organizaciones.

El mercado de operadores móviles en Europa es altamente competitivo y los datos de churn, según un estudio de la consultoría internacional Oliver Wyman (Oliver Wyman, 2023), muestran que un 44% de los consumidores están dispuestos a cambiar de proveedor de servicios de telefonía móvil, y más del 40% tiene la intención de hacerlo en el próximo año.

En cuanto a la intención de churn en el sector de proveedores de servicios móviles en España, se alinea con la media de la UE, situándose en el 46% (Oliver Wyman, 2023). Por lo que los datos demuestran que se trata de un problema latente en este sector.

Ante tal adversidad, el primer paso a llevar a cabo por las empresas es localizar el causante de esta y, aunque en ocasiones es sencillo encontrar el por qué, existen casos en los que resulta difícil hallar un desencadenante concreto. Para todos los casos, pero sobre todo para estos últimos, el aprendizaje automático sirve como herramienta para tratar de anticiparse al abandono de clientes.

La idea es aplicar modelos de clasificación para poder predecir la probabilidad de abandono de un cliente en un momento dado y aplicar las pertinentes medidas para intentar evitar que se marche. Sin embargo, este es uno de los casos en los que la aplicación de estos modelos no siempre es una tarea sencilla, puesto que normalmente se tienen muchos más datos sobre clientes continuos que sobre los que han hecho churn. A este tipo de problemas de clasificación con un conjunto de datos en el que las clases a clasificar no están representadas equitativamente, se les conoce como problemas de clasificación desbalanceados.

Este desbalanceo puede inducir a los modelos predictivos a estar sesgados hacia la clase sobre la que se disponen una mayor cantidad de datos, es decir, en este caso, el modelo tendría una tendencia mayor a predecir que los clientes continuarán en lugar de abandonar la empresa, lo que supondría una disminución en la precisión de la clasificación en las categorías minoritarias. Además, también hace que las métricas utilizadas para medir la efectividad del modelo pierdan fiabilidad. Por ejemplo, si los datos tienen un alto porcentaje de individuos de la clase negativa, el modelo predice todo como negativo y la métrica de accuracy será muy alta, aunque esto no signifique que realmente sea un buen modelo (Na8, 2019).

Este trabajo de fin de máster se centra en abordar esta problemática mediante la exploración de tres métodos fundamentales: el oversampling, el undersampling y la combinación de ambos.

El oversampling, que busca incrementar la representación de las clases minoritarias, y el undersampling, que reduce la presencia de las clases mayoritarias, son abordados en este estudio como métodos potenciales para equilibrar los conjuntos de datos. A través de un análisis detallado, este estudio evalúa la eficacia de estas técnicas en el contexto específico del churn.

En el tercer apartado, se definen teóricamente las principales técnicas de oversampling, undersampling y combinaciones de ambas. Después, se analizan de forma práctica sus efectos sobre los datos de churn en teleco y, finalmente, se extraen conclusiones sobre el estudio realizado.

3. ANÁLISIS TEÓRICO DE LAS TÉCNICAS DE BALANCEO

En este capítulo vamos a desarrollar teóricamente las principales técnicas de resampleo existentes, así como algunas de las combinaciones de estas.

3.1. TÉCNICAS DE OVERSAMPLING

Primeramente, el análisis de las técnicas de oversampling. Estas técnicas consisten en el incremento del número de ejemplos en la clase minoritaria para igualar el número de ejemplos en la clase mayoritaria. Esto se puede hacer duplicando ejemplos de la clase minoritaria o generando nuevos ejemplos sintéticos.

El esquema de la figura 1, muestra visualmente el objetivo de estas técnicas.

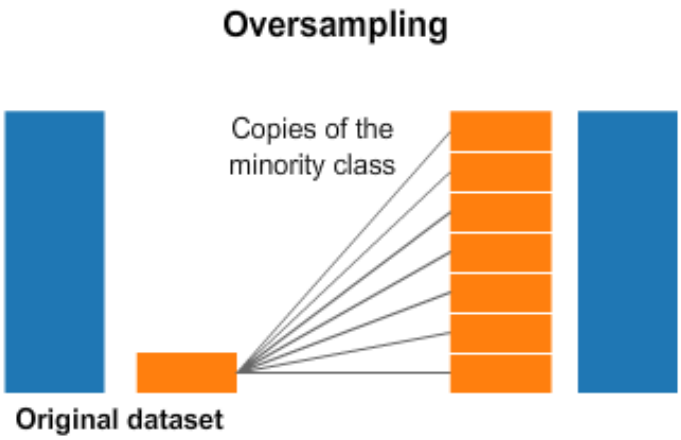


Figura 1: “Undersampling and oversampling: An old and a new approach” (Medium, 2021)

Random Oversampler

A diferencia de métodos más sofisticados como SMOTE o ADASYN, el Random Oversampler incrementa la cantidad de ejemplos en la clase minoritaria mediante la duplicación aleatoria de ejemplos existentes.

Primero, se seleccionan ejemplos de la clase minoritaria de manera aleatoria. Estos ejemplos se duplican para aumentar su presencia en el conjunto de datos y después se insertan de nuevo en él (G Menardi, N. Torelli, 2014).

Al ser una técnica de duplicación directa, el Random Oversampler puede aumentar el riesgo de sobreajuste[1], ya que no introduce nueva información al modelo. Los ejemplos duplicados son exactamente iguales a los originales, lo que significa que el modelo podría aprender a reconocer esos ejemplos específicos en lugar de generalizar a partir de patrones más amplios (G Menardi, N. Torelli, 2014).

Por esta razón, aunque el Random Oversampler es una herramienta rápida y fácil de usar, a menudo se prefiere utilizar SMOTE o ADASYN por proporcionar una mayor diversidad en el conjunto de datos.

SMOTE

Su nombre significa Synthetic Minority Over-sampling Technique (Técnica de Oversampling de Minoría Sintética). A diferencia del oversampling tradicional, que simplemente duplica ejemplos de la clase minoritaria, con SMOTE genera ejemplos sintéticos nuevos y más diversos.

La técnica funciona de la siguiente manera (Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P., 2002):

Para un ejemplo de la clase minoritaria, selecciona uno de sus k vecinos más cercanos también de la clase minoritaria. La selección de estos vecinos se realiza mediante un algoritmo de vecinos más cercanos (k-NN).

[1] El sobreajuste es un comportamiento de aprendizaje automático no deseado que se produce cuando el modelo de aprendizaje automático proporciona predicciones precisas para los datos de entrenamiento, pero no para los datos nuevos. Cuando los científicos de datos utilizan modelos de aprendizaje automático para hacer predicciones, primero entrenan el modelo en un conjunto de datos conocido. Luego, basándose en esta información, el modelo intenta predecir los resultados para los nuevos conjuntos de datos. Un modelo sobre ajustado puede proporcionar predicciones inexactas y no puede funcionar bien para todos los tipos de datos nuevos (Amazon Web Services, 2024).

A continuación, se crea un ejemplo sintético tomando la diferencia entre el ejemplo de la clase minoritaria y su vecino seleccionado. Esta diferencia se multiplica por un número aleatorio entre 0 y 1, y se suma al ejemplo original.

La fórmula para generar un nuevo ejemplo sintético sería:

$$\mathbf{x}_{\text{sintético}} = \mathbf{x}_{\text{minoritario}} + (\mathbf{x}_{\text{vecino}} - \mathbf{x}_{\text{minoritario}}) \times \text{número aleatorio}$$

Donde:

- x minoritario: es un ejemplo existente de la clase minoritaria.
- x vecino*:* es un ejemplo de la clase minoritaria que es uno de los (*k*) vecinos más cercanos de *x minoritario.*
- número aleatorio: es un número aleatorio entre 0 y 1.

Este proceso se repite hasta que la clase minoritaria está adecuadamente representada, equilibrando así la distribución de clases en el conjunto de datos.

La ventaja que tiene usar este método es que, al crear datos sintéticos en vez de replicar los existentes, introduce variabilidad. Esto ayuda a los modelos a generalizar mejor y a ser más robustos frente a datos no vistos.

Un esquema que resume su funcionamiento sería el siguiente:

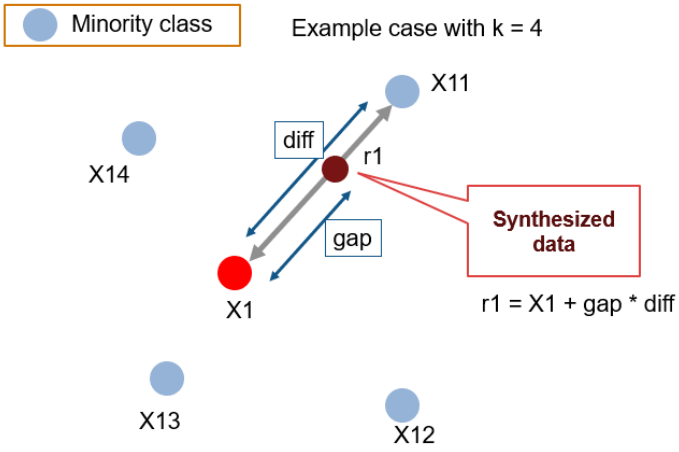


Figura 2: "Overcoming Class Imbalance using SMOTE Techniques" (Analytics Vidhya, 2020)

ADASYN

Al igual que SMOTE, ADASYN (Adaptive Synthetic Sampling) genera ejemplos sintéticos de la clase minoritaria, pero con adaptando la cantidad de ejemplos sintéticos generados para cada ejemplo de la clase minoritaria en función de su nivel de dificultad de aprendizaje.

La técnica ADASYN se basa en los siguientes pasos (He, Haibo, Yang Bai, Edwardo A. Garcia, and Shutao Li, 2008):

1. Calcular la Distribución de Peso: para cada ejemplo de la clase minoritaria, se calcula la distribución de peso que indica cuántos ejemplos sintéticos se necesitan. Esto se hace contando el número de ejemplos de la clase mayoritaria en el vecindario de cada ejemplo de la clase minoritaria.

2. Determinar la Cantidad de Ejemplos Sintéticos: los ejemplos de la clase minoritaria que están rodeados por más ejemplos de la clase mayoritaria recibirán más ejemplos sintéticos, ya que se consideran más difíciles de aprender.

3. Generar Ejemplos Sintéticos: utilizando un algoritmo similar al de SMOTE, se generan los ejemplos sintéticos. La diferencia es que la cantidad de ejemplos sintéticos para cada ejemplo de la clase minoritaria no es uniforme y depende de la distribución de peso calculada previamente.

La fórmula que sigue es similar a la de SMOTE, pero incorporando la distribución de peso:

$$\mathbf{x}_{\text{sintético}} = \mathbf{x}_{\text{minoritario}} + (\mathbf{x}_{\text{vecino}} - \mathbf{x}_{\text{minoritario}}) \times \text{número aleatorio} \times \delta$$

Donde:

- *x minoritario*: es un ejemplo existente de la clase minoritaria.
- *x vecino*: es un ejemplo de la clase minoritaria que es uno de los (k) vecinos más cercanos de *x minoritario*.
- *número aleatorio*: es un número aleatorio entre 0 y 1.
- *delta*: es la distribución de peso que refleja la dificultad de aprendizaje del ejemplo minoritario.

Su objetivo es crear un conjunto de datos más equilibrado, enfocándose en las áreas donde la clase minoritaria es más difícil de aprender. En otras palabras, ajusta el número de ejemplos sintéticos que se generan para cada ejemplo minoritario basándose en la cantidad de ejemplos mayoritarios que hay en su vecindad inmediata.

Borderline-SMOTE

Es una variante de SMOTE diseñada específicamente para tratar con los ejemplos minoritarios que son más difíciles de clasificar porque están en o cerca de la región donde se superponen las clases, lo que a menudo resulta en una mayor tasa de error de clasificación (H. Han, W. Wen-Yuan, M. Bing-Huan, 2005).

Primero, se identifican los ejemplos de la clase minoritaria que están en la "frontera". Esto se hace utilizando un algoritmo de KNN para encontrar aquellos ejemplos minoritarios cuyos vecinos incluyen una cantidad significativa de ejemplos de la clase mayoritaria.

Para cada ejemplo minoritario en la frontera, se seleccionan uno o más de sus vecinos minoritarios más cercanos.

Se generan ejemplos sintéticos a lo largo de la línea que conecta el ejemplo minoritario en la frontera con sus vecinos minoritarios seleccionados. Esto se hace de manera similar a SMOTE, interpolando entre el ejemplo minoritario y sus vecinos.

La idea detrás de Borderline-SMOTE es que, al crear más ejemplos minoritarios cerca de la frontera, el clasificador puede aprender una mejor frontera de decisión entre las clases.

Dicho de manera más sencilla, al enfocarse en la región de superposición, Borderline-SMOTE intenta mejorar la generalización del modelo en las áreas donde es más probable que ocurran errores.

También existe el riesgo de generar sobreajuste si la frontera entre las clases es muy compleja o ruidosa. Por lo tanto, es crucial utilizarla junto con una validación cruzada adecuada y otras técnicas de regularización para asegurar que el modelo resultante sea robusto y generalizable.

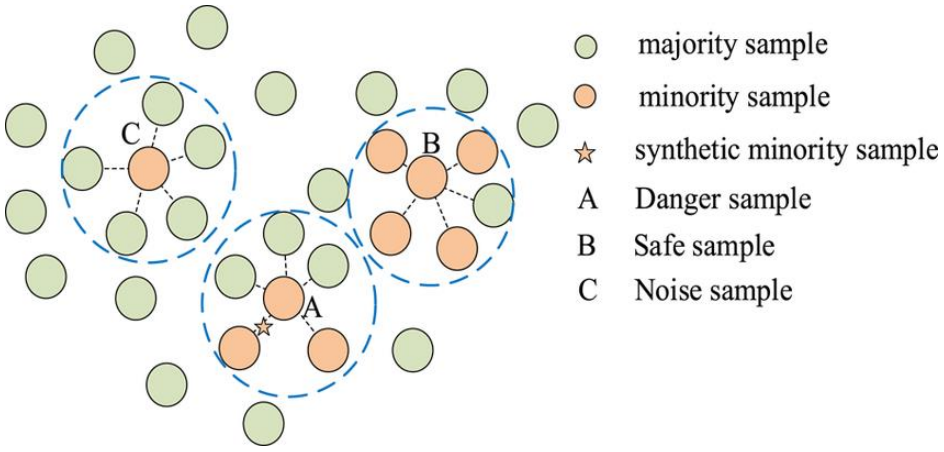


Figura 3: "Diagrama esquemático del principio del algoritmo Borderline-SMOTE" (Pei, Xin & Mei, Fei & Gu, Jiaqi., 2022).

SVMSMOTE

Esta técnica se basa en el uso de Support Vector Machines (SVM) para identificar los ejemplos de la clase minoritaria que están cerca de la frontera de decisión con la clase mayoritaria, como Borderline-SMOTE (H. M. Nguyen, E. W. Cooper, K. Kamei, 2009).

Genera datos sintéticos para estos ejemplos minoritarios seleccionados. La generación se realiza interpolando entre los ejemplos minoritarios y sus vecinos más cercanos dentro de la misma clase, similar a cómo lo hace SMOTE, pero con la guía de la frontera de decisión proporcionada por el SVM.

La ventaja de SVMSMOTE sobre otras técnicas de oversampling es que, al centrarse en los ejemplos cerca de la frontera de decisión, puede mejorar la capacidad del modelo para clasificar correctamente los ejemplos minoritarios que son más difíciles de distinguir. Esto puede resultar en un mejor rendimiento del modelo en términos de precisión y recall[2] para la clase minoritaria.

SVMSMOTE es particularmente útil en situaciones donde la frontera entre clases es compleja y no está claramente definida. Al enfocarse en la región de superposición entre clases, ayuda a construir un clasificador que puede manejar mejor la ambigüedad y mejorar la generalización del modelo.

KMeansSMOTE

Esta técnica integra la robustez del algoritmo K-Means con la generación de ejemplos sintéticos de SMOTE para abordar el desequilibrio de clases en conjuntos de datos. Es particularmente efectiva cuando se trata de conjuntos de datos con múltiples subgrupos dentro de la clase minoritaria, ya que permite una mejor captura de la estructura interna de los datos (Georgios Douzas, Fernando Bacao, Felix Last, 2018).

En lugar de aplicar SMOTE a todo el conjunto de datos minoritario, KMeansSMOTE comienza segmentando la clase minoritaria en varios clústeres utilizando el algoritmo K-Means. Este paso es crucial porque reconoce que no todos los ejemplos minoritarios son iguales y que pueden existir diferentes subgrupos con sus propias características distintivas.

Una vez que los clústeres están formados, SMOTE se aplica dentro de cada clúster, generando ejemplos sintéticos que son coherentes con la distribución de ese subgrupo

---

[2] Precisión: Métrica que se enfoca en la corrección de las predicciones positivas. Ejemplo: “¿De todos los correos electrónicos marcados como spam, qué proporción eran realmente spam?” (Huilgol, P., 2024).

Recall: Métrica que enfatiza en capturar todas las instancias relevantes. Ejemplo: “¿De todos los correos electrónicos de spam reales, qué proporción identificó correctamente el sistema?” (Huilgol, P., 2024).

específico. Esto asegura que la nueva información sintética refleje más fielmente la diversidad dentro de la clase minoritaria.

KMeansSMOTE es eficaz porque no solo equilibra las clases, sino que también enriquece el conjunto de datos con ejemplos que son representativos de las variaciones naturales dentro de la clase minoritaria. Esto puede llevar a un modelo de aprendizaje automático que es más sensible a las diferencias sutiles entre subgrupos, mejorando así la precisión y la capacidad de generalización del modelo.

Al final, KMeansSMOTE ofrece una solución sofisticada para el problema del desequilibrio de clases, proporcionando un enfoque que es tanto metódico como adaptativo a la estructura inherente de los datos.

A continuación, se explicarán las principales técnicas de undersampling.

3.2. TÉCNICAS DE UNDERSAMPLING

El undersampling reduce el número de ejemplos en la clase mayoritaria para igualar el número de ejemplos en la clase minoritaria. Esto se puede hacer eliminando ejemplos de la clase mayoritaria al azar o utilizando métodos más sofisticados para seleccionar los ejemplos más útiles para eliminar.

Aunque estas técnicas pueden ayudar a mejorar la precisión del modelo en la clase minoritaria, también puede llevar a la pérdida de información si no se maneja correctamente. La figura mostrada a continuación muestra un esquema visual del objetivo del undersampling:

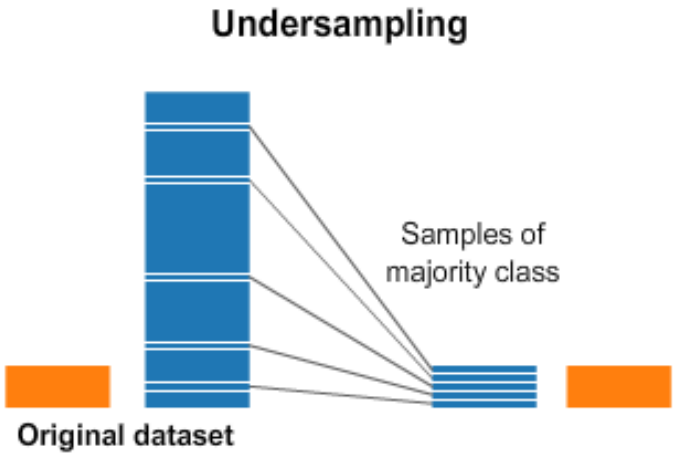


Figura 4: “Undersampling and oversampling: An old and a new approach” (Medium, 2021)

RandomUndersampler

Se trata de un método que sigue la misma filosofía que el Random Oversampling, pero este se centra en reducir la clase mayoritaria en lugar de aumentar la clase minoritaria.

Primero, se identifica la clase mayoritaria. A continuación, se seleccionan aleatoriamente instancias de la clase mayoritaria para ser eliminadas del conjunto de datos. Este proceso se repite hasta que se logra el equilibrio entre ambas clases (Hassannataj Joloudari, J., Marefat, A., Nematollahi, M.A., Oyelere, S.S. & Hussain, S., 2022).

Esta técnica, aunque puede ser efectiva y computacionalmente ligera para equilibrar las clases, puede llevar a la pérdida de información valiosa, ya que se basa en eliminar datos. Además, como el proceso de eliminación de instancias se realiza aleatoriamente, cabe la posibilidad de que la muestra final no sea representativa de la población completa. Esto podría llevar a una pérdida de precisión cuando se hagan predicciones sobre los datos nuevos.

NearMiss

Con este método se seleccionan las instancias a eliminar de una manera más estratégica. NearMiss elimina selectivamente instancias de la clase mayoritaria basándose en su proximidad a las de la clase minoritaria (Hassannataj Joloudari, J., Marefat, A., Nematollahi, M.A., Oyelere, S.S. & Hussain, S., 2022).

Existen varias estrategias conocidas como NearMiss-1, NearMiss-2 y NearMiss-3, cada una con sus propios criterios para determinar qué instancias eliminar:

1- NearMiss-1: esta estrategia selecciona las instancias de la clase mayoritaria que están más cerca de las instancias de la clase minoritaria. Para cada instancia de la clase minoritaria, identifica las instancias de la clase mayoritaria que están más próximas (usualmente se utiliza la distancia euclidiana). Se conserva un número fijo de las instancias más cercanas de la clase mayoritaria para cada instancia de la minoritaria, asegurando que se retengan las instancias más similares y eliminando las demás.

2- NearMiss-2: que funciona de manera opuesta a NearMiss-1. Para cada instancia de la clase mayoritaria, calcula la distancia promedio a las k instancias más cercanas de la clase minoritaria. Las instancias de la clase mayoritaria con las menores distancias promedio a las minoritarias se conservan, mientras que aquellas que están más alejadas se eliminan.

3- NearMiss-3: por último, NearMiss-3 combina elementos de las dos anteriores. Asegura que cada instancia de la clase minoritaria tenga un número fijo de vecinos de la clase mayoritaria que estén más cerca de ella. Se conservan las instancias de la clase mayoritaria que están dentro de los k vecinos más cercanos de al menos una instancia de la clase minoritaria, buscando proporcionar una muestra más equilibrada y representativa de la clase mayoritaria.

Aunque puede ser más efectivo que el Random Undersampling para equilibrar las clases sin perder información valiosa, también puede ser más computacionalmente costoso debido a la necesidad de calcular las distancias entre las instancias.

TomekLinks

En Tomek Links, el objetivo es mejorar la calidad del conjunto de datos al eliminar ciertas instancias que crean solapamiento entre las clases (Hassannataj Joloudari, J., Marefat, A., Nematollahi, M.A., Oyelere, S.S. & Hussain, S., 2022). Un par de instancias (i, j) forman un *Tomek Link* si cumplen dos condiciones:

- Pertenecen a diferentes clases.
- Son los vecinos más cercanos entre sí.

En otras palabras, un par de instancias de diferentes clases se consideran un Tomek Link si cada una es la instancia más cercana de la otra.

Una vez visto esto, el proceso que sigue la técnica es el siguiente:

Primero, se identifican todos los pares de instancias que forman Tomek Links en el conjunto de datos. Esto se hace calculando la distancia (generalmente euclidiana) entre cada instancia y todas las demás, y determinando si forman un Tomek Link.

Una vez identificados los enlaces, se eliminan las instancias del conjunto de datos que forman parte de estos enlaces. Hay diferentes enfoques sobre cuál de las dos instancias se elimina:

- Eliminar solo la instancia de la clase mayoritaria en cada par de Tomek Links. Este enfoque reduce el solapamiento entre las clases y al mismo tiempo ayuda a balancear el conjunto de datos.
- Eliminar ambas instancias de cada par de Tomek Links. Este enfoque es menos común, pero puede ser útil si se quiere eliminar el ruido y las instancias ambiguas en ambos lados del límite de decisión.

En resumen, Tomek Links elimina las instancias que están más cercanas a las instancias de la clase opuesta, reduciendo así el solapamiento entre las clases y mejorando la separación entre ellas. Este método es especialmente útil para limpiar los límites de decisión y mejorar la calidad del conjunto de datos para el entrenamiento de modelos de clasificación.

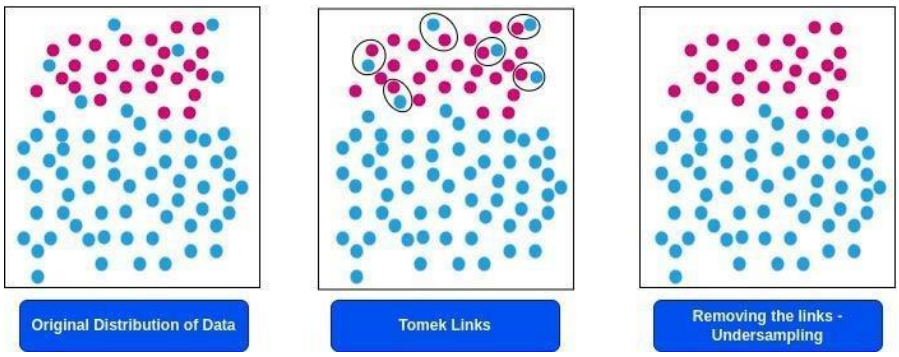


Figura 5: ilustración de Tomek Links (Hassannataj Joloudari, J., Marefat, A., Nematollahi, M.A., Oyelere, S.S. & Hussain, S., 2022)

ClusterCentroids

Este método se basa en la utilización de técnicas de agrupamiento (clustering) para seleccionar qué instancias eliminar (Analytics Vidhya, 2020).

Primero, se lleva a cabo dicha agrupación de instancias de la clase mayoritaria utilizando un algoritmo de clustering, como k-means. Luego, en lugar de trabajar con todas las instancias originales, se utiliza el centroide de cada grupo para representar el conjunto de datos. El número de clústeres, k, se elige de modo que el número de centroides resulte en un balance adecuado con la clase minoritaria. Por ejemplo, si se quiere equilibrar exactamente el número de instancias entre las clases, k se establece igual al número de instancias minoritarias.

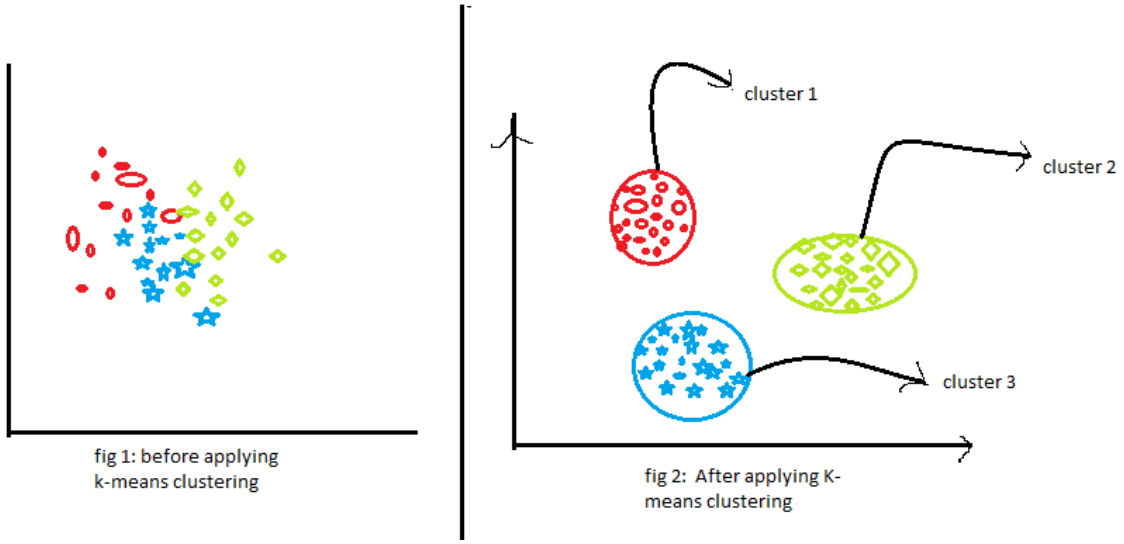


Figura 6: ejemplo de agrupación por clústeres con k-means (Analytics Vidhya, 2020).

Posteriormente, el algoritmo de clustering agrupa las instancias mayoritarias en k clústeres y calcula el centroide de cada clúster. Un centroide es el punto central de un clúster, calculado como el promedio de las instancias en ese clúster.

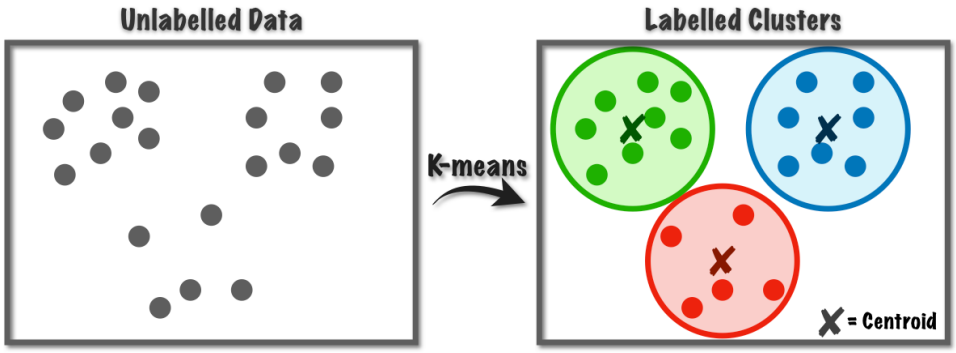


Figura 7: ejemplo de centroides en k-means (Towards Data Science, 2019).

Tras esto, en lugar de mantener todas las instancias originales de la clase mayoritaria, se sustituyen estas instancias por los centroides de los clústeres (Analytics Vidhya, 2020). Esto reduce drásticamente el número de instancias sobrerrepresentadas y mantiene una representación de sus características principales.

Lo positivo de este enfoque es que preserva la estructura y la distribución general de los datos.

One-SidedSelection

En español se conoce como técnica de selección unilateral (One-Sided Selection, OSS). El objetivo que tiene este undersampling es eliminar instancias redundantes y ruidosas, lo que ayuda a crear límites de decisión más claros y precisos en problemas de clasificación binaria. Combina técnicas de eliminación de ruido (Tomek Links) y submuestreo (Condensed Nearest Neighbor, CNN) para lograr un conjunto de datos más equilibrado y limpio (Hassannataj Joloudari, J., Marefat, A., Nematollahi, M.A., Oyelere, S.S. & Hussain, S., 2022).

En primer lugar, se identifican los Tomek Links en el conjunto de datos. Previamente en el trabajo se explican los requisitos para que un par de instancias (i, j) sean Tomek Links (apartado 2.2.3).

Una vez identificadas las uniones, se eliminan las instancias de la clase mayoritaria que forman parte de estos enlaces (M. Kubat, S. Matwin, 1997). Esto ayuda a eliminar las instancias que están en el límite de decisión y que pueden introducir ruido y solapamiento entre las clases y, es por esto por lo que se dice que se utiliza Tomek para reducir el ruido en el conjunto de datos.

Después, se aplica el algoritmo CNN para limpiar aún más la clase mayoritaria. Este elimina una instancia si su clase no coincide con la clase mayoritaria de sus k vecinos más cercanos (usualmente k=3).

Finalmente, OSS aplica submuestreo a la clase mayoritaria para equilibrar el número de instancias entre las clases. El submuestreo puede ser realizado de manera aleatoria o utilizando otras técnicas más sofisticadas para seleccionar las instancias más representativas de la clase mayoritaria.

Al aplicar OSS, se mejora la calidad del conjunto de datos y los límites de decisión del modelo porque se eliminan instancias ruidosas y mal clasificadas, pero además se mejora la precisión y la capacidad generalizadora del modelo.

EditedNearestNeighbours

Su objetivo principal es eliminar las instancias que podrían estar introduciendo ruido o que están mal clasificadas. Asegura de esta forma que el modelo de clasificación que se entrene con estos datos sea más preciso y efectivo (D. Wilson, 1972).

Imaginemos que tienes un conjunto de datos con dos clases: la Clase A, que es mayoritaria, y la Clase B, que es minoritaria. ENN trabaja revisando cada instancia del conjunto de datos para ver si se encuentra en una posición correcta respecto a sus vecinos.

Para cada instancia en el conjunto de datos, busca sus k vecinos más cercanos, que se determinan calculando la distancia entre la instancia analizada y todas las demás instancias del conjunto de datos. Una métrica comúnmente utilizada para esto es la distancia euclidiana, que básicamente mide la recta imaginaria entre dos puntos en el espacio de características.

Una vez que se han encontrado los vecinos más cercanos, se observa a qué clase pertenecen estos vecinos. La idea es ver cuál es la clase más común entre estos (es decir, la clase mayoritaria entre ellos). Si la instancia en cuestión pertenece a la misma clase que la mayoría, se considera que está bien clasificada. Sin embargo, si pertenece a una clase diferente, se asume que podría estar en una región donde está mal clasificada o que está introduciendo ruido en los datos.

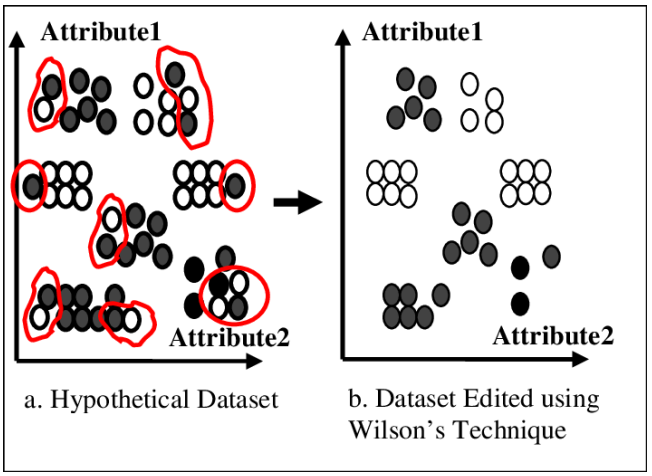


Figura 7: ejemplo visual de ENN (Eick, Christoph & Zeidat, N. & Vilalta, Ricardo. 2004)

Breve ejemplo práctico de ENN

Supongamos que en el conjunto de datos de churn del trabajo tenemos los siguientes datos sobre cinco clientes:

Cliente	Edad	Duración del contrato	Llamadas	Clase
1	30	24	100	0
2	45	12	80	1
3	34	6	200	0
4	22	18	150	1
5	40	24	130	0

Figura 8: tabla ejemplo ENN (Fuente: elaboración propia)

Si a estos mismos datos le aplicamos un ENN con k = 3, los resultados mostrarían:

Las distancias calculadas para el cliente 1 con respecto a los otros clientes son aproximadamente:

- Cliente 2: 27.73
- Cliente 3: 101.69
- Cliente 4: 50.99
- Cliente 5: 31.62

Vecinos más cercanos:

- Cliente 2: clase 1
- Cliente 5: clase 0
- Cliente 4: clase 1

La clase mayoritaria entre los vecinos más cercanos de este cliente es la clase 1.

Como la clase del cliente 1 es la 0, no coincide con la clase mayoritaria de sus vecinos, y la instancia cliente 1 se eliminaría.

El proceso de eliminación de instancias ruidosas o mal clasificadas se repite para todas las instancias del conjunto de datos. Al final, el conjunto de datos resultante tiene menos ruido y es más limpio, lo que ayuda a mejorar la precisión y la eficacia del modelo de clasificación que se entrene con estos datos.

Resultado Final

Después de revisar todas las distancias y las clases, se observa que se eliminarían el cliente 1 y el 2, ya que su clase no coincide con la clase mayoritaria de sus vecinos más cercanos. El data set limpio sería:

Cliente	Edad	Duración del contrato	Llamadas	Clase
3	34	6	200	0
4	22	18	150	1
5	40	24	130	0

Figura 9: tabla resultado del ejemplo ENN (Fuente: elaboración propia)

ENN es especialmente útil porque permite limpiar los datos de manera sistemática, enfocándose en instancias que podrían estar complicando los límites de decisión entre las clases. Esto lleva a modelos que son más robustos y capaces de generalizar mejor a datos no vistos. Sin embargo, como las otras técnicas de undersampling, es importante notar que también puede eliminar algunas instancias valiosas, especialmente en conjuntos de datos pequeños. También, la técnica puede ser computacionalmente costosa en conjuntos de datos grandes.

RepeatedEditedNearestNeighbours

RENN es una extensión de ENN que introduce la repetición del proceso de eliminación de instancias ruidosas (D. Wilson, 1972).

La diferencia clave entre ENN y RENN radica en la repetición del proceso de eliminación de instancias ruidosas.

Mientras que ENN realiza una única pasada para eliminar instancias que no coinciden con la clase mayoritaria de sus vecinos más cercanos, RENN repite este proceso varias veces. En cada iteración, RENN elimina instancias ruidosas y luego vuelve a calcular los vecinos más cercanos de las instancias restantes. Este proceso se repite hasta que ya no se detectan instancias ruidosas o se alcanza un criterio predefinido de convergencia.

La repetición del proceso en RENN permite una mayor exhaustividad en la identificación y eliminación de instancias ruidosas. Esto puede ser especialmente útil en conjuntos de datos con un alto grado de ruido o desbalanceados, donde una sola pasada puede no ser suficiente para eliminar todas las instancias problemáticas.

Por estas características, RENN puede llegar a ser más efectivo que ENN, pero también resulta más costoso computacionalmente.

AllKNN

Este undersampling sigue el mismo procedimiento que el ENN, pero lo aplica repetidas veces variando para cada vez el número de vecinos más cercanos (D. Wilson, 1972). Cada repetición se aumenta el número de vecinos en una unidad.

El proceso termina cuando se examina el número máximo de vecinos o cuando la clase mayoritaria se convierte en la clase minoritaria, lo que ocurra primero.

Por tratarse de otra variación del ENN, tiene los mismos pros y contras que este, los cuales fueron explicados en el apartado 2.2.6 (explicación del ENN).

CondensedNearestNeighbour

Este procedimiento selecciona un subconjunto representativo de las instancias originales, preservando aquellos puntos más críticos para la correcta clasificación (G. Batista, R. C. Prati, M. C. Monard, 2004).

El método comienza eligiendo aleatoriamente un ejemplo de cada clase del conjunto de datos, formando así el conjunto de partida. Luego, se itera a través del resto de las muestras y se clasifica cada una utilizando los vecinos más próximos de este conjunto inicial. Si una instancia es incorrectamente clasificada, se añade al conjunto de referencia. Este procedimiento se repite hasta que no se agreguen nuevas instancias en una iteración completa, asegurando que solo las instancias esenciales, aquellas que definen los límites de las clases, sean retenidas.

Una de las principales ventajas del Condensed Nearest Neighbour es su capacidad para reducir significativamente el tamaño del conjunto de datos sin perder información crucial para la discriminación de clases. Esto puede llevar a un aumento en la eficiencia computacional durante la fase de entrenamiento de los modelos, ya que se trabaja con menos datos, y a una mejora en la capacidad de generalización del modelo al eliminar ruido.

No obstante, esta técnica también presenta ciertas limitaciones. En conjuntos de datos con alta superposición entre clases, CNN puede no ser tan efectivo, ya que el proceso puede conservar un número excesivo de instancias, minimizando así el beneficio de la reducción. Además, en casos de alta dimensionalidad, la técnica puede no ser suficiente para manejar la complejidad del espacio de características.

Condensed Nearest Neighbour es particularmente útil en problemas de clasificación binaria y multiclase donde la distribución de las clases es altamente desigual. Es frecuente su aplicación en áreas como la detección de fraudes, el diagnóstico médico y otras situaciones donde los datos minoritarios son críticos.

Al implementar esta técnica, es esencial realizar un análisis previo del conjunto de datos para evaluar la adecuación de CNN. En muchos casos, combinar CNN con otras técnicas de preprocesamiento y selección de características puede potenciar sus beneficios. Además, la validación cruzada y otros métodos de evaluación son cruciales para asegurar que el modelo resultante sea robusto y generalizable.

Para finalizar la explicación teórica, el siguiente apartado desarrolla las diferentes combinaciones de técnicas que se han aplicado durante la investigación.

3.3. COMBINACIÓN DE AMBOS TIPOS DE TÉCNICAS

Una vez analizadas las distintas técnicas de undersampling y oversampling de forma individual, es fundamental explorar la combinación de ambas metodologías para abordar de manera más efectiva los problemas de desbalanceo de clases en conjuntos de datos. La combinación de estas estrategias permite aprovechar las fortalezas de cada enfoque, mitigando las limitaciones inherentes cuando se aplican de manera aislada.

Como ya se ha explicado, el undersampling se caracteriza por reducir la cantidad de ejemplos de la clase mayoritaria, mientras que el oversampling incrementa el número de instancias de la clase minoritaria. Integrar estas dos aproximaciones puede resultar en un conjunto de datos más equilibrado y representativo, mejorando así el rendimiento y la precisión de los modelos predictivos.

En esta sección, se profundizará en las metodologías que combinan ambas técnicas, detallando cómo su sinergia puede optimizar la calidad de los datos y, en consecuencia, la eficacia de los algoritmos de aprendizaje automático. En concreto, en el trabajo se han aplicado dos combinaciones diferentes: SMOTEENN y SMOTETomek. A continuación, su explicación teórica.

SMOTEENN

Esta combinación integra el método de oversampling SMOTE con el método de undersampling Edited Nearest Neighbours (G. Batista, R. C. Prati, M. C. Monard, 2004). Por ser combinación de estas dos técnicas, su enfoque es el mismo que tienen estas. Busca mejorar la calidad del conjunto de datos al aumentar la representación de la clase infrarrepresentada y limpiar el ruido y las instancias ambiguas en la clase sobrerrepresentada.

Una vez explicadas las técnicas de SMOTE (Synthetic Minority Over-sampling Technique) y ENN (Edited Nearest Neighbours) en apartados anteriores, es fundamental entender cómo la combinación de ambas puede mejorar el balanceo de clases en un conjunto de datos.

Por un lado, resumidamente, SMOTE es una técnica de oversampling que crea nuevas instancias sintéticas de la clase minoritaria mediante la interpolación de las características entre instancias existentes y sus vecinos más cercanos. Esto resulta en un aumento de la diversidad y la representatividad de la clase minoritaria, ayudando a mitigar problemas de desbalanceo.

ENN, por otro lado, es una técnica de undersampling que elimina instancias ruidosas o ambiguas. Identifica y elimina ejemplos que no coinciden con la mayoría de sus vecinos más cercanos, reduciendo así el ruido.

La técnica SMOTEENN combina las fortalezas de SMOTE y ENN en un proceso de dos fases.

Primero, se utiliza SMOTE para generar nuevas instancias sintéticas de la clase minoritaria. Este paso aumenta el tamaño y la diversidad de la clase minoritaria, logrando aumentar el equilibrio.

A continuación, se aplica ENN para limpiar el conjunto de datos resultante. ENN elimina aquellas instancias que son clasificadas incorrectamente por sus vecinos más cercanos, reduciendo el ruido y eliminando ejemplos ambiguos tanto en la clase mayoritaria como en la minoritaria.

La combinación de ambas técnicas asegura un conjunto de datos más equilibrado sin sobrerrepresentar la clase minoritaria ni mantener instancias ruidosas. La fase de ENN elimina el ruido, lo que contribuye a mejorar la precisión y la capacidad de generalización del modelo. Y la fase de SMOTE incrementa la diversidad en la clase minoritaria, lo que ayuda al modelo a aprender características más representativas.

SMOTEENN puede ser computacionalmente intenso, especialmente en conjuntos de datos grandes. También hay que tener en cuenta que la efectividad de este combinado depende de la correcta elección de parámetros, como el número de vecinos utilizados tanto en SMOTE como en ENN.

SMOTETomek

Por último, esta otra combinación, como su nombre indica, integra SMOTE y Tomek-Links.

De nuevo, la explicación individual de las técnicas se ha hecho en apartados posteriores, luego en este simplemente se expone un resumen de ellas y una explicación en detalle de la combinación.

A modo de recordatorio, Tomek Links identifica y elimina pares de instancias de diferentes clases que son vecinos más cercanos entre sí.

Introducido por G. Batista en 2003, este método combina la capacidad de SMOTE para generar datos sintéticos de la clase minoritaria con la capacidad de Tomek Links para eliminar datos de la clase mayoritaria que se identifican como enlaces de Tomek (es decir, muestras de datos de la clase sobrerrepresentada que están muy cerca de los datos de la clase infrarrepresentada). El proceso de SMOTE-Tomek Links se desarrolla de la siguiente manera (G. Batista, R. C. Prati, M. C. Monard, 2003):

Primero, se aplica SMOTE siguiendo los mismos pasos explicados en el apartado anterior.

Después, con Tomek-Links se seleccionan datos aleatorios de la clase mayoritaria y, si el vecino más cercano de estos datos es una muestra de la clase minoritaria (formando un enlace de Tomek), entonces elimina el enlace.

Este enfoque permite equilibrar el conjunto de datos al aumentar la cantidad de datos de la clase minoritaria y eliminar aquellos datos de la clase mayoritaria que están demasiado cerca de los datos minoritarios, mejorando así la separación entre las clases.

Una vez visto el enfoque teórico, el apartado siguiente contiene el desarrollo de la parte práctica del estudio.

4. ANÁLISIS PRÁCTICO DE LA APLICACIÓN DE LAS TÉCNICAS DE BALANCEO

El objetivo principal de este trabajo es explorar el efecto que tienen las técnicas de balanceo de clases en los resultados de modelos de clasificación utilizados para problemas desbalanceados. En los apartados anteriores se han explicado las diferentes técnicas de forma teórica, a continuación, se explicarán sus efectos mediante su aplicación práctica a un conjunto de datos sobre el abandono de clientes en una empresa de telecomunicaciones.

4.1. EDA

Antes de la modelización y el uso de este tipo de técnicas, se ha llevado a cabo un proceso comúnmente conocido como EDA o análisis exploratorio de datos. Este procedimiento se lleva a cabo en todos los proyectos de Ciencia de Datos para lograr entender el conjunto de datos del que se dispone, instancias faltantes, datos inservibles, información potencialmente sensible y bajo políticas de protección de datos.

Para comenzar este exploratorio, se han comprobado las características esenciales del conjunto. Lo primero que se debe mencionar de los datos, es su origen. Estos datos se obtuvieron de la página web de Kaggle en el enlace: https://www.kaggle.com/datasets/blastchar/telco-customer-churn?resource=download.

Kaggle es una plataforma en línea que alberga la comunidad de Ciencia de Datos más grande del mundo. Permite el acceso gratuito a una gran cantidad de datos y código compartidos por la comunidad (DataScientest, 2024).

Se trata de información perteneciente al banco de conjuntos de datos de ejemplo de la consultora IBM. Esta se compone de 7.043 filas y 21 columnas con información contractual y demográfica de los clientes y, además, una columna con información sobre si han dejado de ser clientes o no (en total 22 columnas).

Un desglose de las variables del conjunto sería el siguiente:

1. Age: la edad actual del cliente, en años, al finalizar el trimestre fiscal.

2. Churn: que si es “Yes” significa que dejó la empresa este trimestre. Mientras que, si es “No”, el cliente se quedó con la empresa.

3. Contract: muestra el tipo de contrato actual: Month-to-Month (mensual), One Year (anual), Two Year (bianual).

4. customerID: un ID único que identifica a cada cliente.

5. Dependents: si vive con dependientes. Esta variable también se representa como Yes o No. Los dependientes pueden ser hijos, padres, abuelos, etc.

6. DeviceProtection: si tiene suscrito un servicio adicional de seguridad en línea proporcionado por la empresa (Yes, No).

7. gender: el género del cliente. Teniendo como opciones Male (Masculino) o Female (Femenino).

8. InternetService: para ver si tiene suscrito el servicio de Internet con la empresa. Puede verse como No, DSL, Fiber Optic (Fibra Óptica) o Cable.

9. Partner: indica si está casado (Yes o No).

10. MonthlyCharges: cargo mensual total actual por todos sus servicios de la empresa.

11. MultipleLines: certifica que tiene suscrito múltiples líneas telefónicas con la empresa (Yes o No).

12. OnlineBackup: para comprobar si tiene suscrito un servicio adicional de copia de seguridad en línea proporcionado por la empresa. También con valores Yes o No.

13. OnlineSecurity: si ha contratado un plan adicional de protección de dispositivos para su equipo de Internet proporcionado por la empresa: Yes si lo tiene contratado y No si no lo ha contratado.

14. PaperlessBilling: si hace la facturación de manera electrónica: Yes, No.

15. PaymentMethod: muestra el método para pagar su factura: Bank Withdrawal (Retiro bancario), Credit Card (Tarjeta de crédito), Mailed Check (Cheque enviado por correo).

16. PhoneService: en caso de que el cliente tenga suscrito el servicio telefónico residencial con la empresa aparece como Yes, en caso contrario, como No.

17. SeniorCitizen: si tiene 65 años o más (Yes, No).

18. StreamingMovies: para ver si utiliza su servicio de Internet para transmitir películas de un proveedor externo: Yes, No. La empresa no cobra una tarifa adicional por este servicio.

19. StreamingTV: variable que informa sobre si usa su servicio de internet para transmitir programación de televisión de un proveedor externo: Yes, No. La empresa no cobra una tarifa adicional por este servicio.

20. TechSupport: es el indicador de si tiene suscrito un plan adicional de soporte técnico de la empresa con tiempos de espera reducidos: Yes, No

21. tenure: es el total de meses que el cliente ha estado con la empresa.

22. TotalCharges: son los cargos totales, calculados hasta el final del trimestre especificado anteriormente.

En total, disponemos de trece variables binarias, cuatro variables categóricas y cuatro numéricas.

El objetivo principal de este conjunto es que sea utilizado para hacer modelos sobre el abandono de clientes y que, el algoritmo, al haber aprendido sobre los datos que ha utilizado para entrenarse, sea capaz de hacer predicciones buenas con datos de clientes nuevos o de los que aún no se sabe si abandonarán la compañía. De esta forma, la empresa podría hacer campañas de marketing o focalizar sus esfuerzos para intentar retener a los clientes que tengan una mayor probabilidad de hacer churn.

A continuación, una gráfica de la distribución del abandono en el conjunto de datos del trabajo:

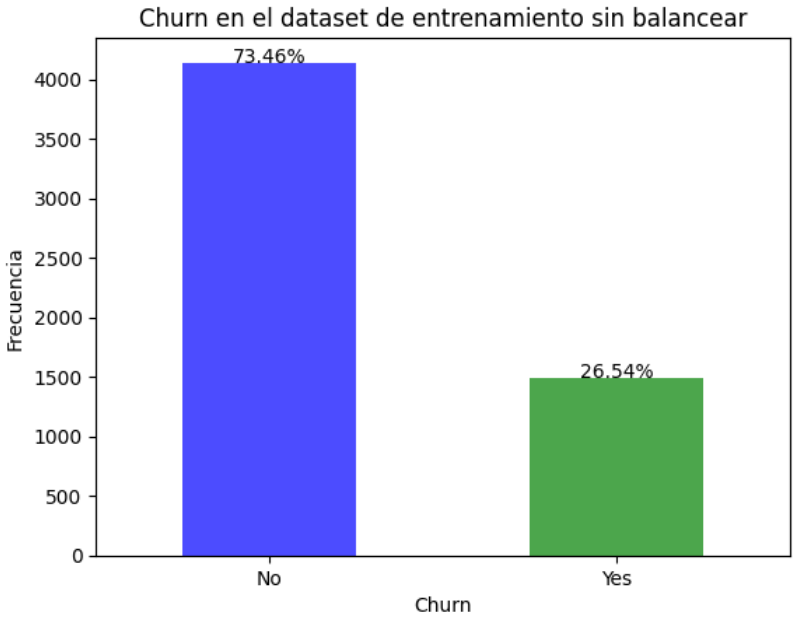


Figura 10: distribución de churn en el conjunto de datos utilizado

(Fuente: elaboración propia con matplotlib)

Tras el análisis, se han eliminado las variables customerID y Gender porque, no solo no aportaría nada al modelo en términos de clasificación, sino que, además, las normativas de protección de datos requieren que ningún dato de este estilo sea utilizado para hacer modelos. Con lo que, de usarlo para modelar, estaríamos incurriendo en una ilegalidad. Además, la columna TotalCharges tenía 11 valores nulos, pero, tras revisar el data set

original, se observó que esos valores deberían ser un 0, así que se imputaron de esa forma.

Finalmente, tras el tratamiento de nulos y valores faltantes, se construyó un pipeline de pasos para automatizar el preprocesamiento de los datos. Dicho pipeline consiste en un primer paso de codificación de variables categóricas mediante One Hot Encoding y un escalado de las variables numéricas mediante Standard Scaler para que todas tengan un impacto equiparable en el modelo.

Resumidamente, el One-Hot Encoding es una técnica de preprocesamiento utilizada para convertir variables categóricas en un formato que pueda ser provisto a algoritmos de machine learning. Transforma cada categoría en una nueva columna binaria (0 o 1), indicando la presencia (1) o ausencia (0) de esa categoría en cada instancia del conjunto de datos. Por otra parte, el Standard Scaler es una técnica de normalización que transforma las características de los datos para tener una media de 0 y una desviación estándar de 1. Esto se logra restando la media y dividiendo por la desviación estándar de cada característica, lo que ayuda a estandarizar los datos para algoritmos de machine learning que son sensibles a la escala de los datos.

Seguidamente, se llevó a cabo un proceso de selección del modelo a utilizar.

4.2. SELECCIÓN DEL MODELO

Como el objetivo principal del estudio no es comparar diferentes tipos de modelos, sino ver los efectos de las técnicas de balanceo, se ha utilizado el mismo modelo para todas las técnicas para facilitar la comparación de resultados.

Random Forest

Este modelo pertenece a la familia de modelos de ensambles de árboles de decisión. Una de sus principales ventajas es su versatilidad, ya que puede utilizarse tanto para problemas de clasificación como de regresión (IBM, 2024).

Consiste en construir múltiples árboles de decisión durante el entrenamiento y combinar sus predicciones para mejorar la precisión y robustez del modelo. La idea central es que un conjunto de modelos "débiles" (los árboles de decisión) pueden combinarse para formar un modelo "fuerte" que obtenga mejores resultados.

El proceso de construcción de un Random Forest involucra varios pasos clave:

Se comienza generando múltiples subconjuntos del dataset original seleccionando aleatoriamente ejemplos de datos con reemplazo. Esto significa que un mismo ejemplo puede aparecer en más de un subconjunto. Este proceso se conoce como Boostrapping.

A continuación, cada uno de estos subconjuntos se utiliza para entrenar un árbol de decisión independiente. Estos árboles son entrenados con una variación: en cada nodo de decisión, solo se considera un subconjunto aleatorio de características en lugar de todas las características disponibles. Esto introduce diversidad adicional en los árboles.

Una vez se han entrenado los árboles, cada uno proporciona una predicción basada en los datos de entrada y, después, las predicciones de todos los árboles se combinan. En tareas de regresión, se promedia el resultado de todos los árboles mientras que, en problemas de clasificación (como el de este estudio), la predicción final se obtiene mediante votación mayoritaria. Es decir, la clase más frecuente es la predicción del bosque.

Al promediar los resultados de múltiples árboles, Random Forest tiende a reducir el riesgo de sobreajuste en comparación con un solo árbol de decisión.

Por otro lado, los modelos de Random Forest son robustos frente a outliers[3] y ruido en los datos, ya que los árboles individuales no se ven tan influenciados por ejemplos atípicos.

También, este modelo proporciona una estimación de la importancia de cada característica, lo cual es útil para entender qué variables son más influyentes en las predicciones del modelo.

Por último, es efectivo para datos con relaciones no lineales y complejas.

4.3. SELECCIÓN DE MÉTRICAS

Antes de explicar las diferentes métricas de los modelos, conviene hablar de la matriz de confusión y sus elementos. Un esquema de la matriz sería el que aparece a continuación en la figura 10.

	Predicho como positivo	Predicho como negativo
Realmente positivo	Verdaderos Positivos (VP)	Falsos Negativos (FN)
Realmente negativo	Falsos Positivos (FP)	Verdaderos Negativos (VN)

[3] En palabras más sencillas, un outlier es una observación dentro de una muestra o una serie temporal de datos que no es consistente con el resto, es decir, que se separa mucho de la media de distribución de datos (Sanjuán y López, 2021).

Figura 11: matriz de confusión (Fuente: elaboración propia)

Los verdaderos positivos son los clientes que el modelo ha clasificado como personas con una alta probabilidad de abandonar y en el conjunto de datos realmente han abandonado. Mientras que, los verdaderos negativos son los que ha predicho que se mantengan como clientes y realmente siguen siéndolo.

Por otra parte, los falsos positivos son aquellos clientes que el modelo ha predicho que abandonarán, pero la realidad dice lo contrario. De forma similar, los falsos negativos son esos casos en los que el modelo se equivoca y predice que no se irán, cuando en realidad si lo han hecho.

Una vez visto esquemáticamente la matriz de confusión, pasemos a ver las principales métricas existentes (DataSource.ai, 2024):

- Exactitud (Accuracy): la exactitud es la proporción de predicciones correctas (tanto verdaderos positivos como verdaderos negativos) entre el total de predicciones.

  Se calcula como:

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

  Es útil cuando las clases están balanceadas, pero puede ser engañosa en problemas de clasificación desbalanceados como este porque un modelo que siempre predice la clase mayoritaria puede tener una alta exactitud, pero será ineficaz para identificar correctamente la clase minoritaria.

- Precisión (Precision): mide la proporción de verdaderos positivos entre todos los positivos predichos.

  Se calcula así:

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

Es especialmente útil cuando el coste de los falsos positivos es alto. Por ejemplo, en un modelo de detección de spam, la precisión sería importante para asegurar que los correos legítimos no sean etiquetados incorrectamente como spam.

- Exhaustividad (Recall) o Sensibilidad (Sensitivity): mide la proporción de verdaderos positivos entre todos los positivos reales.

  Se calcula de la siguiente forma:

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

  Debe fijarse como objetivo cuando el costo de los falsos negativos es alto.

- F1-Score: es la media armónica de la precisión y el recall. Ofrece un balance entre ambos. Su fórmula matemática es la siguiente:

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

  Es útil cuando se necesita un equilibrio entre precisión y recall.

- F2-Score: es una variante del F1-score que pone más énfasis en el recall que en la precisión. Esto puede ser útil en situaciones donde es más importante capturar la mayor cantidad posible de verdaderos positivos, incluso si eso significa aceptar un mayor número de falsos positivos.

  Se define de la siguiente forma:

$$F2Score = 5 * \frac{Precision * Recall}{4 * Precision + Recall}$$

En comparación con el F1-Score, el F2 penaliza más fuertemente los falsos negativos que los falsos positivos y es útil en situaciones donde es crucial capturar la mayoría de los casos positivos como, por ejemplo, en modelos para la detección de enfermedades mortales o el fraude fiscal.

- AUC-ROC (Área Bajo la Curva de la Característica Operativa del Receptor): Evalúa el rendimiento del modelo en diferentes umbrales de clasificación. Un valor más alto indica mejor rendimiento.

  Matemáticamente, se expresa así:

$$AUC - ROC = \int_0^1 Tasa\ de\ VP\ \ d(Tasa\ de\ FP)$$

  Una representación gráfica del AUC-ROC sería la siguiente:

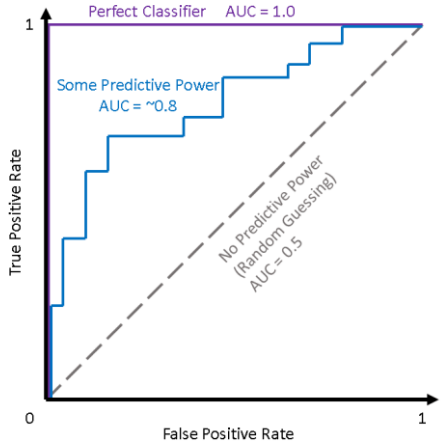


Figura 12: gráfico explicativo de la curva ROC y el AUC (Sanahuja, 2021)

El clasificador perfecto tendría un valor de AUC igual a uno, mientras que el que tenga un valor de 0.5 es un clasificador sin poder predictivo, que realiza predicciones aleatorias (Sanahuja, 2021).

De todas estas, la métrica seleccionada para este estudio ha sido el F2-Score. Para seleccionar la métrica se ha tenido en cuenta la premisa de que a la empresa le cuesta muchísimo más captar nuevos clientes que retenerlos. Por tanto, siguiendo esta lógica, tiene sentido utilizar el F2-Score como guía para seleccionar el mejor modelo, ya que penaliza más fuertemente los falsos negativos que los falsos positivos. Es decir, a la

organización le saldría mucho más caro que el modelo se equivoque y diga que no hay peligro de abandono con un cliente que finalmente prescinda de sus servicios, a que el modelo diga que sí va a hacer churn, pero luego no lo haga, ya que, en el primer caso, el coste de reemplazar al cliente es enorme y, en el segundo, se incurrirían en costes de descuentos y campañas de marketing que hasta puede que hagan que el cliente recomiende la empresa a otros. Dicho de otra forma, los gastos en el segundo caso podrían llegar a considerarse una buena inversión. Si bien es cierto que tampoco es bueno que la organización dirija gran parte de sus recursos económicos hacia sus clientes más leales porque no se estaría focalizando bien el esfuerzo, esto es un mal menor comparado con la pérdida de clientes que ocasionaría un modelo que tenga una alta tasa de falsos negativos.

Una vez realizado el análisis exploratorio, seleccionado el tipo de modelo a utilizar y las métricas más adecuadas, se puso a prueba el rendimiento del modelo sin aplicar técnicas con el fin de tener un punto de comparación para los resultados posteriores. El proceso que se ha seguido con todas estas pruebas ha consistido en probar modelos estándar, es decir, sin ajustar sus parámetros y, después, probar los mismos modelos realizando un ajuste de parámetros. De esta forma se han obtenido dos tipos de resultados para cada modelo. Además, para cada uno, se han calculado las predicciones sobre el set de entrenamiento y sobre el de prueba. De esta forma, es posible ver si los modelos están sufriendo de sobreajuste.

Veamos ahora los resultados resumidos de cada modelo (para más detalles: ANEXO II, ANEXO III y ANEXO IV).

4.4. MODELO BASE

Es importante destacar que la idea de este modelo base no es conseguir el mejor modelo posible, sino, como se ha mencionado previamente, lograr crear un punto de referencia para hacer una comparación de resultados. Es decir, obtener los primeros resultados sin ningún esfuerzo para luego probar formas de mejorarlos.

Para este primer caso, no se aplican técnicas de resampling, por lo que se espera que el modelo esté muy sesgado hacia la clase mayoritaria y el rendimiento sea malo tanto en el modelo sin ajustar, como en el modelo ajustado. Dicho ajuste se ha realizado haciendo lo que se conoce como un *grid search*, que en español sería una búsqueda de los mejores parámetros para el modelo teniendo en cuenta una cuadrícula de parámetros dada. Es decir, se prueban los modelos con diferentes combinaciones de los parámetros de la cuadrícula dada hasta encontrar la combinación que brinda los mejores resultados.

Resultados

El modelo base recibió el conjunto de datos con la distribución de churn mostrada en la figura 10 (73.46% de continuación y solo 26.54% de abandono). Con estas condiciones, ha logrado las métricas expuestas en la siguiente tabla:

Conjunto	Modelo	F2-Score	Recall	Tiempo (min)
Entrenamiento	Base	0.9933	0.9926	0.01
	Base Ajustado	0.7063	0.6870	3.38
Test	Base	0.5089	0.4893	0.01
	Base Ajustado	**0.5596**	**0.5401**	**3.38**

Figura 13: tabla de métricas del modelo base (Fuente: elaboración propia)

En este caso, se aprecia como el modelo tiene mejores métricas cuando se calculan sobre el set de entrenamiento, lo cual es perfectamente normal. Además, como era de esperar, el resultado del modelo en el test ha sido muy malo, puesto que no dispone de un equilibrio suficiente entre las clases de la variable Churn y seguramente no haya aprendido lo suficiente sobre los clientes que abandonaron la empresa como para poder hacer predicciones sólidas con nuevos datos.

Por último, cabe destacar que el ajuste de parámetros ha demostrado mejorar el rendimiento del modelo sobre el set importante, el de prueba. Se observa también sobreajuste porque la diferencia entre las métricas logradas sobre el set de entrenamiento y sobre el de prueba, son bastante pronunciadas.

A continuación, prosigue el análisis con los resultados de los modelos tras la aplicación de las distintas técnicas de oversampling.

4.5. RESULTADOS OBTENIDOS CON LAS TÉCNICAS DE OVERSAMPLING

Para la obtención de estos resultados, simplemente se ha aplicado cada técnica al conjunto de datos y se han entrenado los modelos (para más detalle sobre la distribución de churn: ANEXO II). Primeramente, sin realizar ajustes de parámetros y después haciendo el mismo proceso de grid search que se explicó anteriormente en el trabajo. Una vez que se realizó este proceso, se compararon los resultados de los modelos sin ajustar con los de su versión ajustada.

La tabla mostrada a continuación es un resumen de las métricas obtenidas por los modelos tras aplicar las técnicas de oversampling y predecir sobre el conjunto de prueba (para más detalle: ANEXO IV):

Conjunto	Modelo	F2-Score	Recall	Tiempo (min)
Test	Random Oversampler	0.5726	0.5802	0.01
	Random Oversampler Ajustado	0.5839	0.5936	2.83
	SMOTE	0.5775	0.5775	0.02
	SMOTE Ajustado	0.6639	0.6639	3.37
	ADASYN	0.5559	0.5535	0.02
	ADASYN Ajustado	0.5949	0.6070	3.39
	Borderline-SMOTE	0.5682	0.5722	0.02
	Borderline-SMOTE Ajustado	**0.6881**	**0.7433**	**3.36**
	SVMSMOTE	0.5659	0.5695	0.03
	SVMSMOTE Ajustado	0.6234	0.6417	3.45
	KMeansSMOTE	0.5546	0.5455	0.02
	KMeansSMOTE Ajustado	0.6267	0.6310	3.06

Figura 14: tabla de métricas obtenidas con los modelos tras aplicar métodos de oversampling (Fuente: elaboración propia)

Como puede observarse, el modelo con mejores resultados en este caso ha sido el **Borderline-SMOTE con los parámetros ajustados**. Este modelo tuvo un F2-Score de 68.9%, un recall del 74.4% y tardó en ajustar sus parámetros y entrenarse un total de 3 minutos y 36 segundos.

Pese a no ser el modelo más rápido de estos, la diferencia en métricas compensa ese factor.

Si lo comparamos con el modelo base, el Borderline-SMOTE también tiene un mejor rendimiento y, además, tarda ligeramente menos en obtener los resultados.

A continuación, se exponen los resultados de los modelos con las técnicas de undersampling.

4.6. RESULTADOS OBTENIDOS CON LAS TÉCNICAS DE UNDERSAMPLING

Las métricas conseguidas fueron las siguientes:

Conjunto	Modelo	F2-Score	Recall	Tiempo (min)
Test	Random Undersampler	0.6796	0.7487	0.01
	Random Undersampler Ajustado	0.7246	0.8048	1.33
	NearMiss	0.5722	0.7353	0.01
	NearMiss Ajustado	0.6442	0.8529	1.36
	TomekLinks	0.5769	0.5775	0.01
	TomekLinks Ajustado	0.6130	0.6150	1.99
	ClusterCentroids	0.7116	0.8155	0.04
	ClusterCentroids Ajustado	0.7073	0.8102	1.46
	OneSidedSelection	0.5838	0.5829	0.01
	OneSidedSelection Ajustado	0.6066	0.6070	1.95
	EditedNearestNeighbours	0.6994	0.7888	0.01
	EditedNearestNeighbours Ajustado	0.6977	0.7888	1.49
	RepeatedEditedNearestNeighbours	**0.7257**	**0.8770**	**0.01**
	RepeatedEditedNearestNeighbours Ajustado	0.7294	0.8824	1.24
	AllKNN	0.7202	0.8396	0.01
	AllKNN Ajustado	0.7198	0.8422	1.36
	CondensedNearestNeighbour	0.6212	0.6524	0.69
	CondensedNearestNeighbour Ajustado	0.6785	0.7246	1.34

Figura 15: tabla de métricas obtenidas con los modelos tras aplicar métodos de undersampling (Fuente: elaboración propia)

Dentro de los resultados de los modelos con undersampling, puede verse como el mejor modelo es el que tiene aplicada la técnica del Repeated Edited Nearest Neighbour y tiene los parámetros ajustados, sin embargo, si tenemos en cuenta el tiempo de ejecución, la versión sin ajustar obtiene unos resultados similares en mucho menor tiempo, luego ese sería el mejor modelo por ser más eficiente.

La diferencia entre un tiempo de ejecución en este caso es notable. Aun así, cualquiera que vea los tiempos que se han registrado podría pensar que no hablamos de una gran cantidad de tiempo y que quizá valga la pena esperar, pero fuera del mundo académico, en la realidad hay conjuntos de datos muchísimo más grandes que el que se ha utilizado para este estudio, así que cualquier diferencia de tiempo con una obtención de resultados similares, podría suponer a la empresa un gran ahorro de tiempo, energía, recursos y costes generales.

En comparación con el modelo base, el RENN es mejor en todos los aspectos. Mientras el f2-score del modelo base ajustado es 0.5596 y el recall es 0.5401, el resultado en estas métricas del RENN sin siquiera necesitar ajustar los parámetros es **0.7257 de F2** y **0.8770 de recall**.

Veamos ahora los resultados de la combinación de tipos de técnicas.

4.7. RESULTADOS OBTENIDOS CON LA MEZCLA DE AMBOS TIPOS DE TÉCNICAS

Al combinar under y oversampling, se lograron los siguientes resultados:

Conjunto	Modelo	F2-Score	Recall	Tiempo (min)
Test	**SMOTEENN**	**0.7075**	**0.7914**	**0.01**
	SMOTEENN Ajustado	0.7057	0.7888	1.82
	SMOTETomek	0.5864	0.5882	0.02
	SMOTETomek Ajustado	0.5870	0.5936	3.22

Figura 16: tabla de métricas obtenidas con los modelos tras aplicar una combinación de métodos de balanceo (Fuente: elaboración propia)

En este caso, el modelo con mejores resultados fue el Random Forest con la combinación de SMOTEENN siendo, no solo el que mejores métricas ha obtenido, sino también el más eficiente a la hora de hacerlo. Ocurre algo similar al caso anterior, los resultados contra su modelo ajustado son prácticamente iguales, pero el modelo sin ajuste es bastante más rápido, por eso se preferiría en un caso real. De nuevo, el Random Forest con esta técnica logra unos resultados muy superiores al modelo base.

En el siguiente apartado, antes de pasar a las conclusiones, he creído relevante mencionar la incorporación al TFM de mlflow y las ventajas que esto tendría en un caso real.

5. MLFLOW

Para mejorar el seguimiento y comparación de los resultados de los modelos en este proyecto, se ha integrado mlflow. Esta librería es una herramienta de código abierto comúnmente utilizada en la industria que ofrece una interfaz fácil de acceder y utilizar para comparar resultados de diferentes modelos y, no solo eso, registrar todos los hiperparámetros y características de cada uno.

Una de las ventajas más destacadas de esta plataforma es su capacidad para garantizar la reproducibilidad de los experimentos. Cada ejecución de modelo se registra de forma exhaustiva, lo que permite a cualquier miembro del equipo replicar fácilmente los resultados y colaborar de manera efectiva en el desarrollo de modelos.

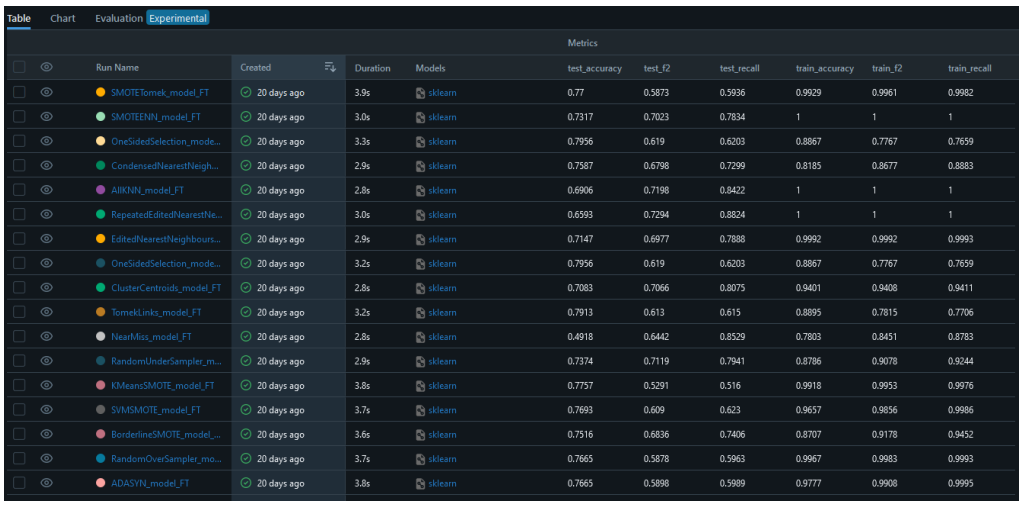


Figura 17: ejemplo de la interfaz de mlflow (Fuente: elaboración propia)

Ha sido utilizada en este trabajo con el objetivo de implementar buenas prácticas y hacer del estudio un ejemplo más realista. De esta forma, cualquiera que desee acceder al trabajo y probar cosas diferentes o ampliar lo que se ha hecho, podría hacerlo teniendo una interfaz fácil de acceder y sin necesidad de crear sus propias tablas de métricas ni gráficos.

Para hacerlo, simplemente tendría que clonar mi repositorio de GitHub adjuntado en el ANEXO I para poder acceder a todo el código y material utilizado para hacer este trabajo. Una vez tenga el repositorio en sus archivos locales, debe crearse un entorno de programación para instalar todas las librerías que aparecen en el archivo de texto adjunto “requirements.txt”. Habiendo hecho esto, simplemente tendría que abrir la consola de su ordenador o editor de código y seguir los pasos que aparecen en la siguiente imagen:

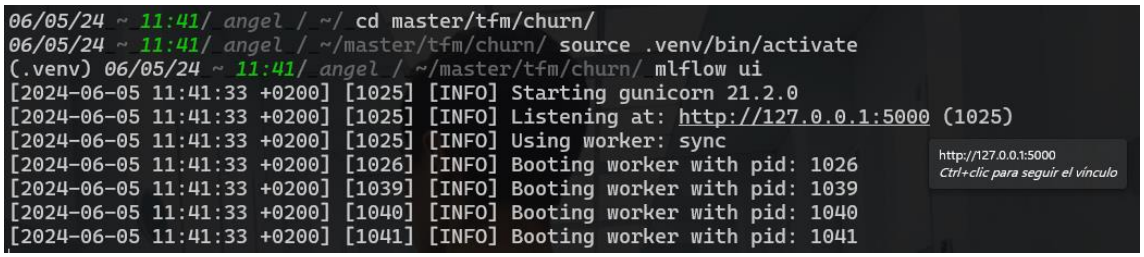


Figura 18: como acceder a la interfaz de mlflow (Fuente: elaboración propia)

Primero, accede a la carpeta donde ha clonado el repositorio y tiene el entorno de librerías creado y, una vez ahí dentro, activa dicho entorno según su tipo y su nombre (el mío es tipo *pip* lo he llamado *.venv*).

Por último, simplemente escribe “mlflow ui” y hace Ctrl + clic en el enlace que aparece unos momentos más tarde. De esta forma, ya tendría acceso a la interfaz de mlflow para gestionar los modelos.

Una vez visto la utilidad de mlflow y el porqué de su aplicación a este estudio, pasemos a hablar de las conclusiones alcanzadas.

6. CONCLUSIONES

Del estudio se pueden sacar varias conclusiones. A continuación, algunas de ellas:

- El mejor modelo **sin ajustar parámetros** ha sido el Random Forest tras haber aplicado la técnica de undersampling de **Repeated Edited Nearest Neighbours**. Además, si se tiene en cuenta una ratio entre el resultado y el tiempo de entrenamiento, ha sido también el modelo más eficiente (0.01 minutos sin ajustar y 1.24 minutos ajustado).

- Por otra parte, es el **Random Forest con la misma técnica de undersampling** que obtiene los mejores resultados dentro de los modelos con parámetros tuneados previamente. Pese a esto, no es tan eficiente en cuanto al tiempo que tarda en entrenar como su versión sin ajustar, por lo que no compensa la poca mejora de métricas que consigue (0,0037 más de F2 y 0,0054 más de recall).

- Por tanto, el mejor modelo teniendo en cuenta todos los tipos y técnicas probadas, es el **Random Forest con RENN sin ajustar** puesto que consiguió los segundos mejores resultados de una manera mucho más eficiente que el modelo de las mejores métricas y, además, no presentó síntomas de sobreajuste ya que las métricas en test no se redujeron drásticamente con respecto a las logradas en el set de entrenamiento.

- Pese a que la técnica que de los mejores resultados depende en gran parte del conjunto de datos del que se disponga, las técnicas de resampleo, así como la combinación de ambos tipos, han demostrado ser una opción viable para mejorar la calidad de los modelos de predicción en problemas desbalanceados. Y no solo eso, en concreto, las técnicas de undersampling, son una gran herramienta para reducir el tamaño de enormes conjuntos de datos y hacer que los modelos sean más eficientes, lo que podría suponer un ahorro energético, de tiempo y de desgaste de recursos enorme para una organización.

- Otra de las conclusiones del trabajo, quizá en un plano más personal, es que resulta de vital importancia aplicar buenas prácticas en programación y, para estos casos de pruebas de modelos, utilizar librerías como mlflow para mantener el control de las diferentes versiones de los modelos y probar otros nuevos sin necesidad de crear más código o almacenarlos tratando de no sobrescribirlos.

- Por último, este trabajo nunca tuvo como objetivo alcanzar el mejor modelo posible de todos, para asegurarse de eso habría que explorar otras combinaciones de parámetros del modelo, técnicas y también introducir regularizaciones para combatir el sobreajuste. Pese a ello, serviría como plantilla para que cualquier persona use el mismo código y haga pruebas con diferentes modelos y combinaciones de parámetros.

7. ANEXOS

ANEXO I: ENLACE A REPOSITORIO DE GITHUB

Para la elaboración del trabajo, se ha utilizado Python como principal lenguaje de programación.

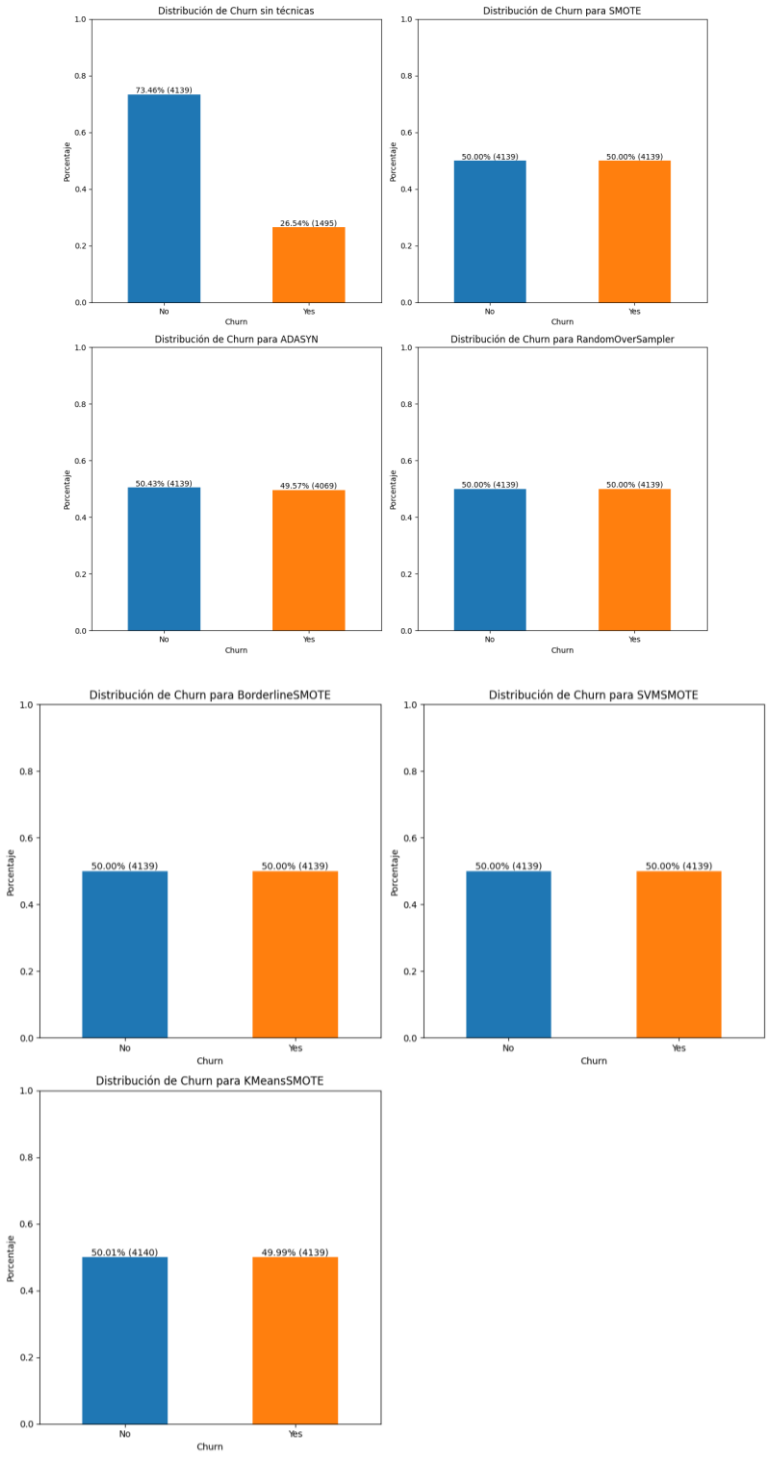
https://github.com/angelblancoq/churn

ANEXO II: DISTRIBUCIÓN DE LA VARIABLE OBJETIVO TRAS APLICAR RESAMPLING

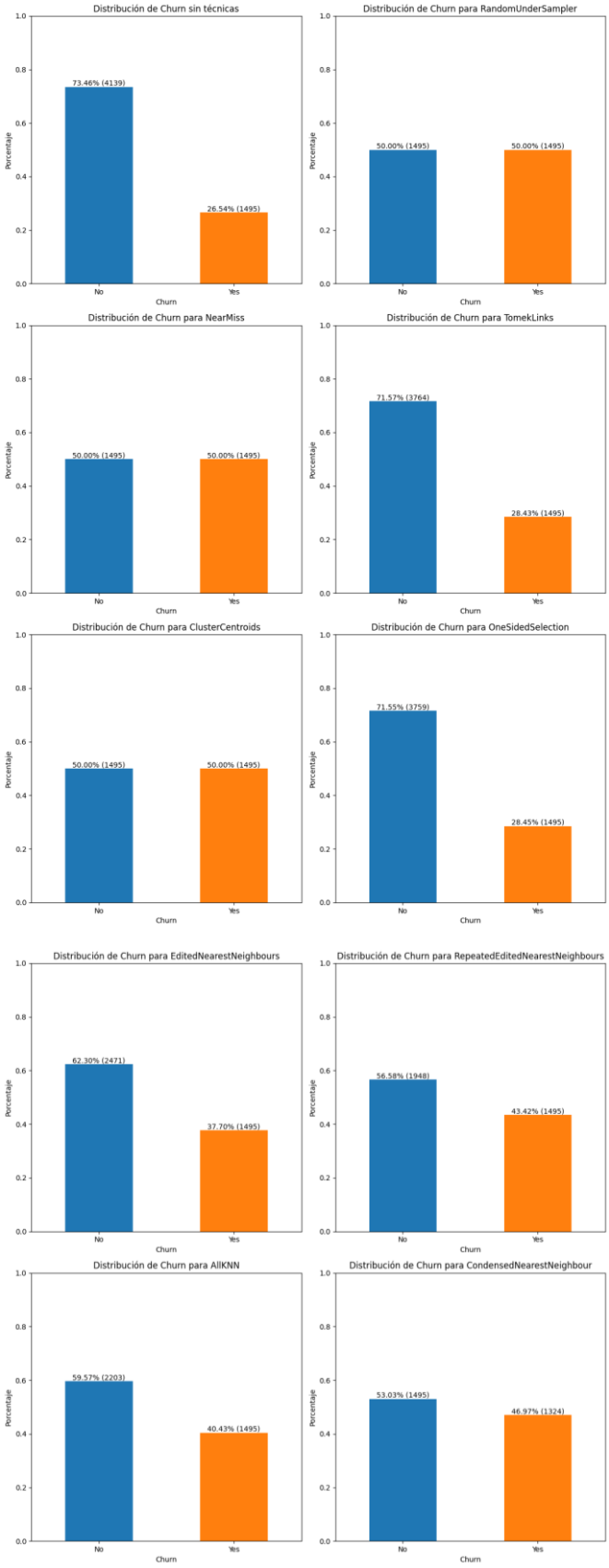
Este segundo anexo sirve para mostrar gráficamente las distintas distribuciones de la variable objetivo Churn tras la aplicación de cada técnica de resampling y compararlas fácilmente con la distribución del conjunto de datos original. Cabe destacar que cada barra representa el porcentaje de cada valor de la variable y que también aparece representado un valor numérico para saber cuánto es ese porcentaje en cada caso.

Todos los gráficos que aparecen a continuación han sido elaborados por mí con la librería de Python Matplotib:

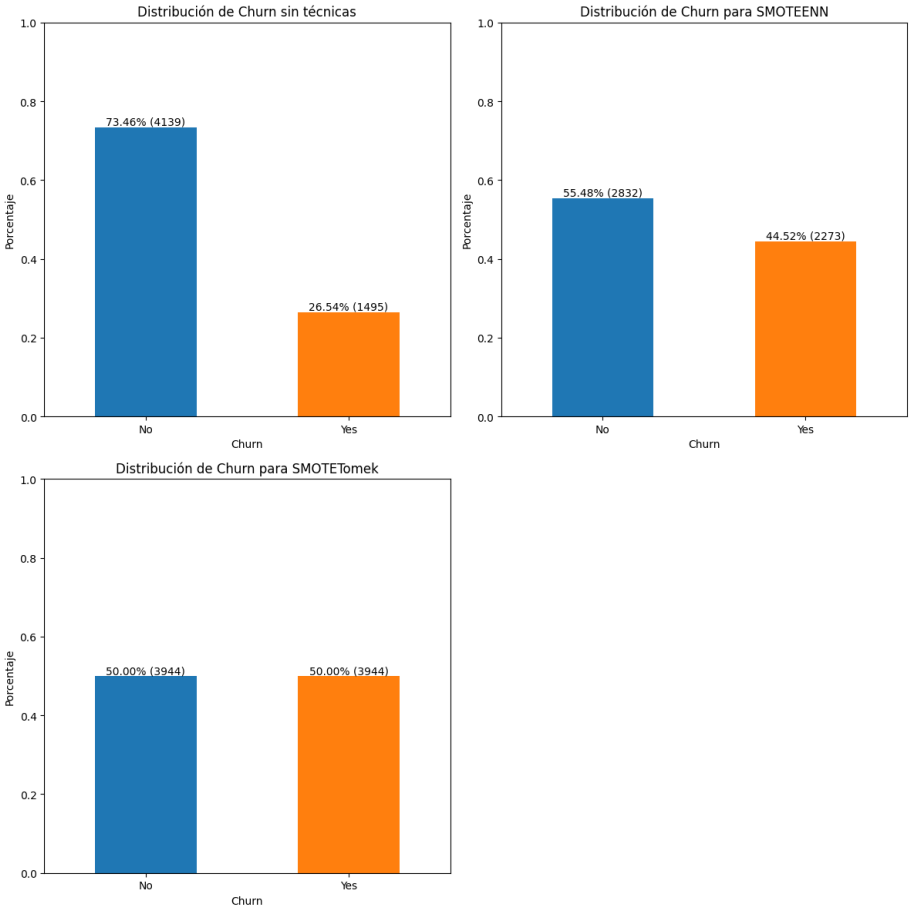
Distribución de Churn tras la aplicación de oversampling



Distribución de Churn tras la aplicación de undersampling

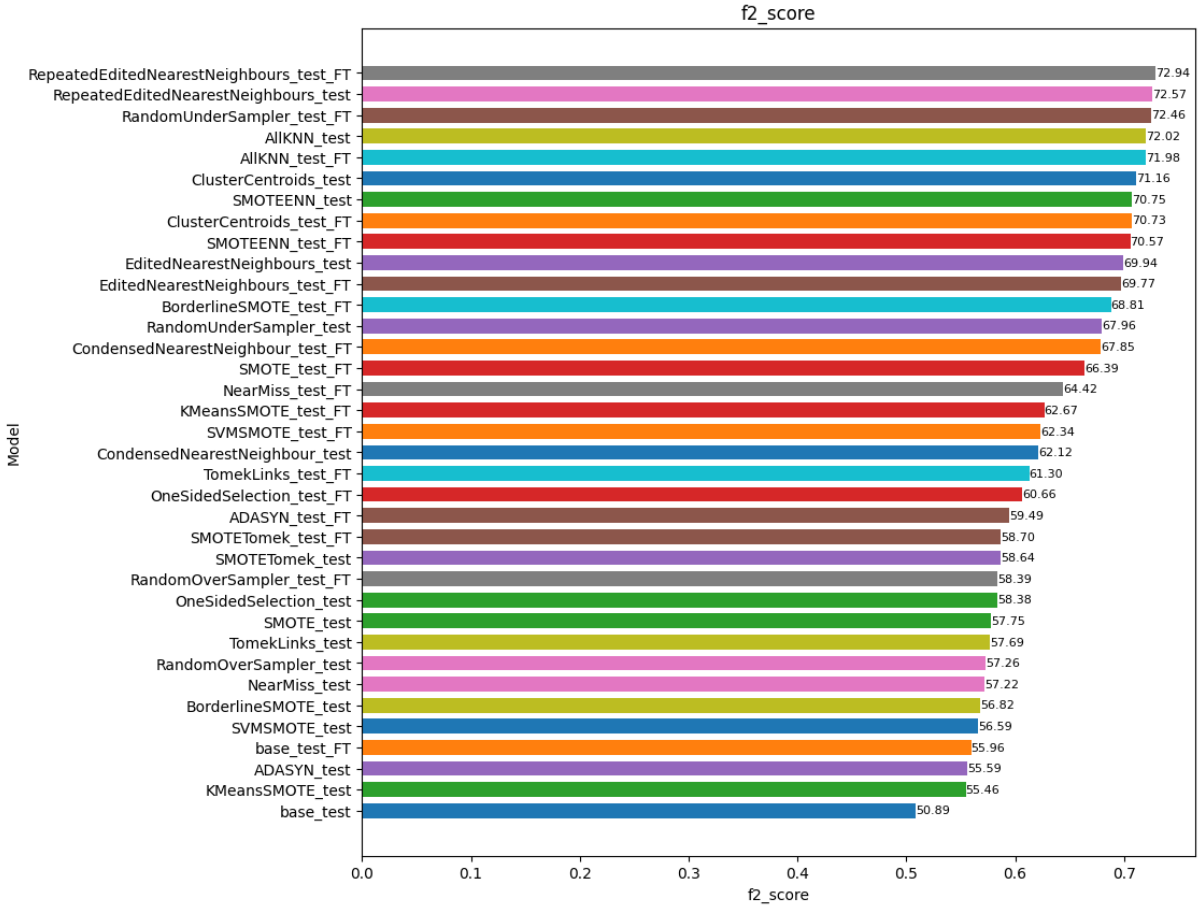

Distribución de Churn sin técnicas
73.46% (4139)
26.54% (1495)
Distribución de Churn para RandomUnderSampler
50.00% (1495)
50.00% (1495)
Distribución de Churn para NearMiss
50.00% (1495)
50.00% (1495)
Distribución de Churn para TomekLinks
71.57% (3764)
28.43% (1495)
Distribución de Churn para ClusterCentroids
50.00% (1495)
50.00% (1495)
Distribución de Churn para OneSidedSelection
71.55% (3759)
28.45% (1495)
Distribución de Churn para EditedNearestNeighbours
62.30% (2471)
37.70% (1495)
Distribución de Churn para RepeatedEditedNearestNeighbours
56.58% (1948)
43.42% (1495)
Distribución de Churn para AllKNN
59.57% (2203)
40.43% (1495)
Distribución de Churn para CondensedNearestNeighbour
53.03% (1495)
46.97% (1324)
Porcentaje
Churn
No
Yes

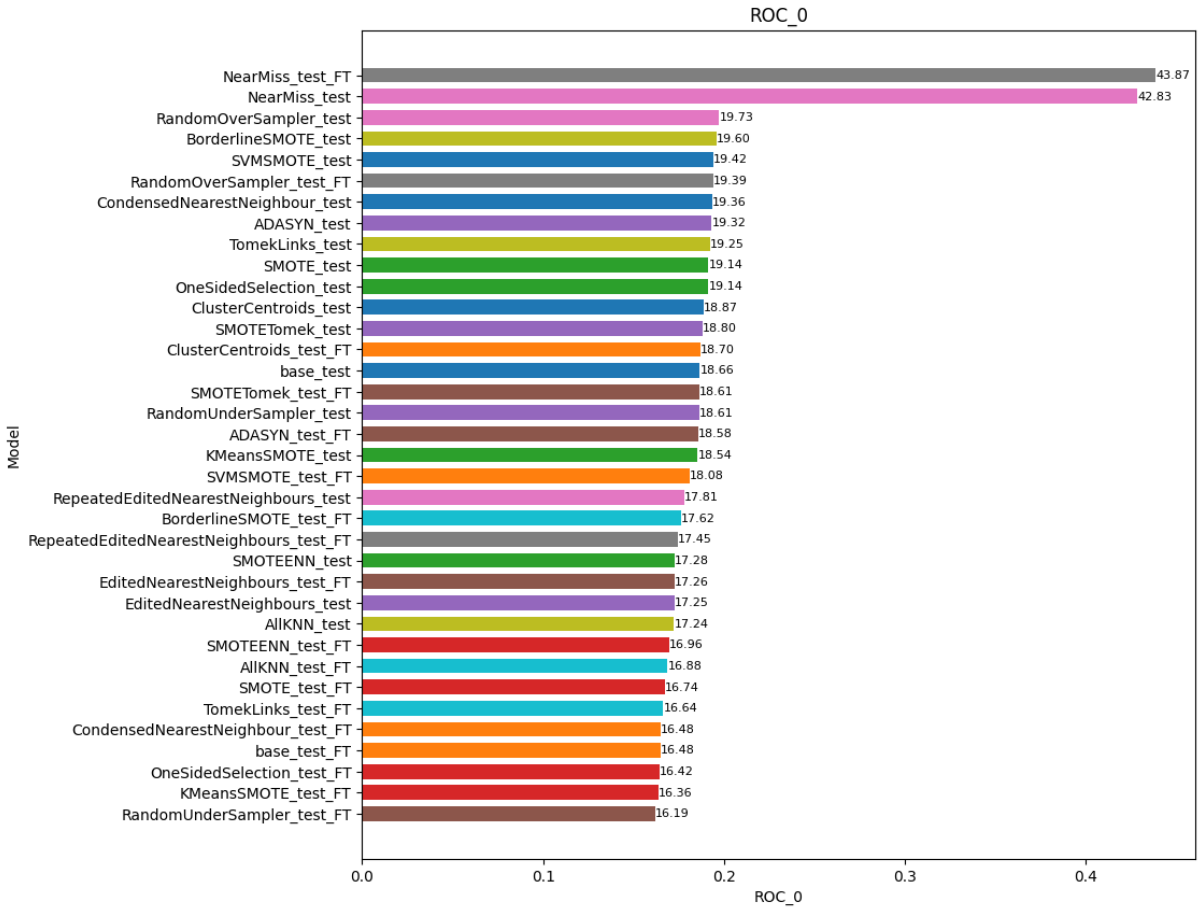
Distribución de Churn tras combinación de técnicas

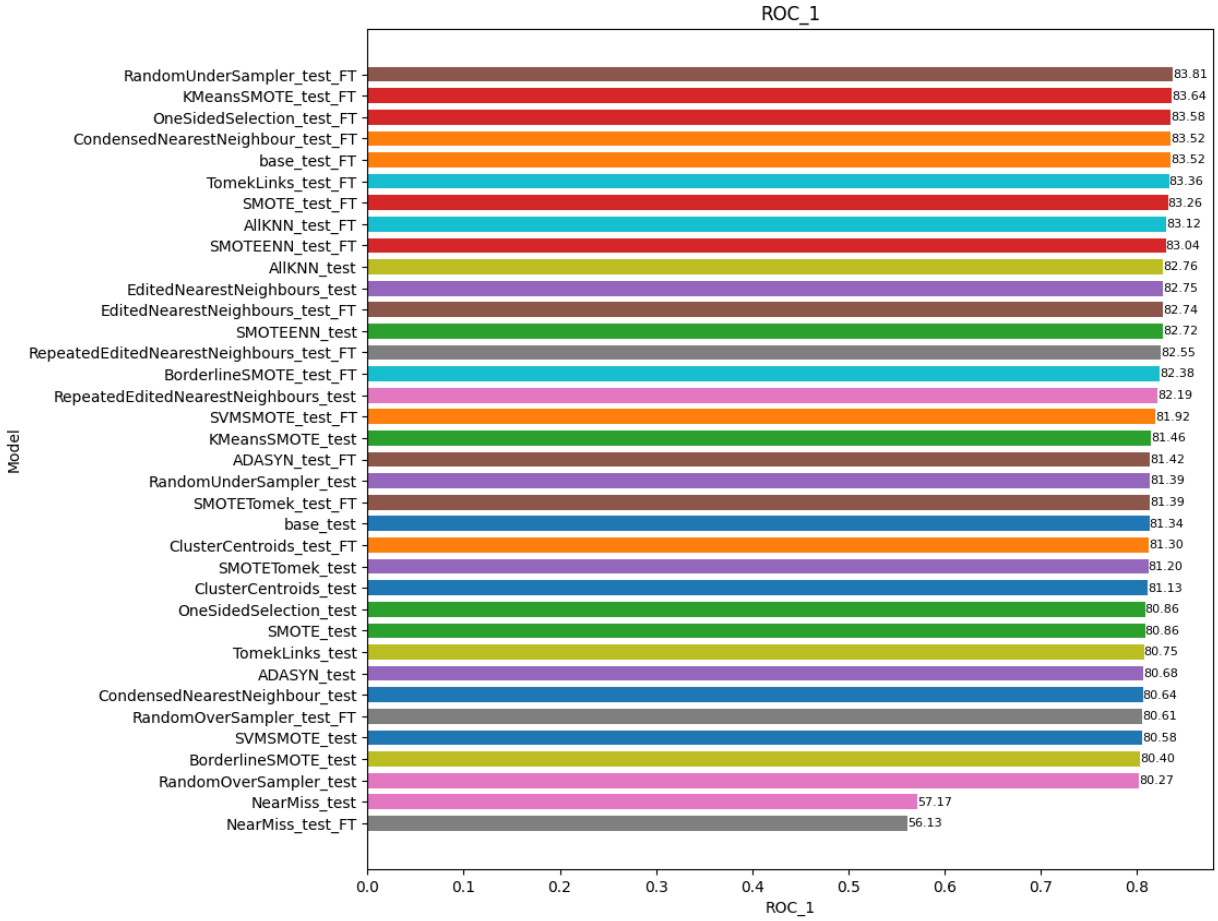


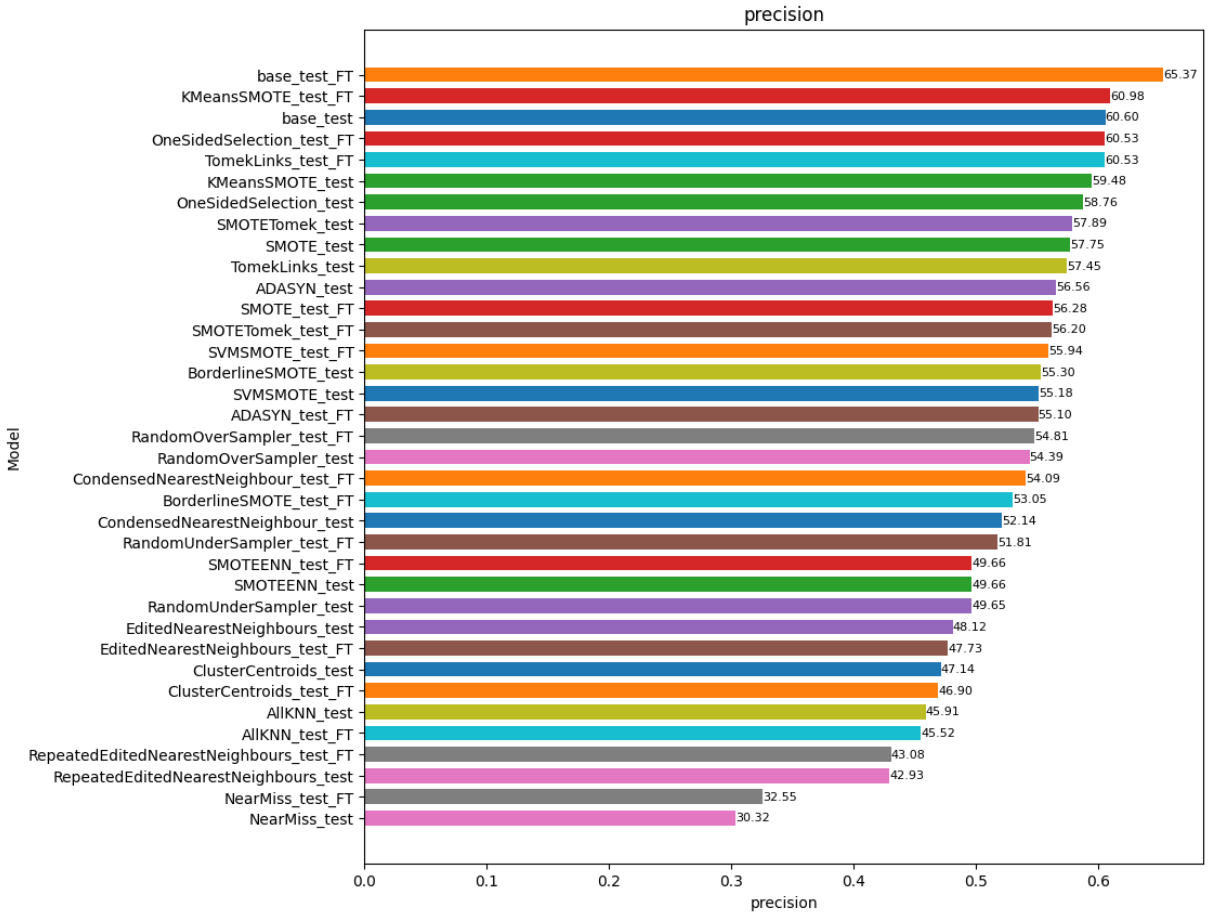
ANEXO III: GRÁFICOS COMPARATIVOS DE MÉTRICAS

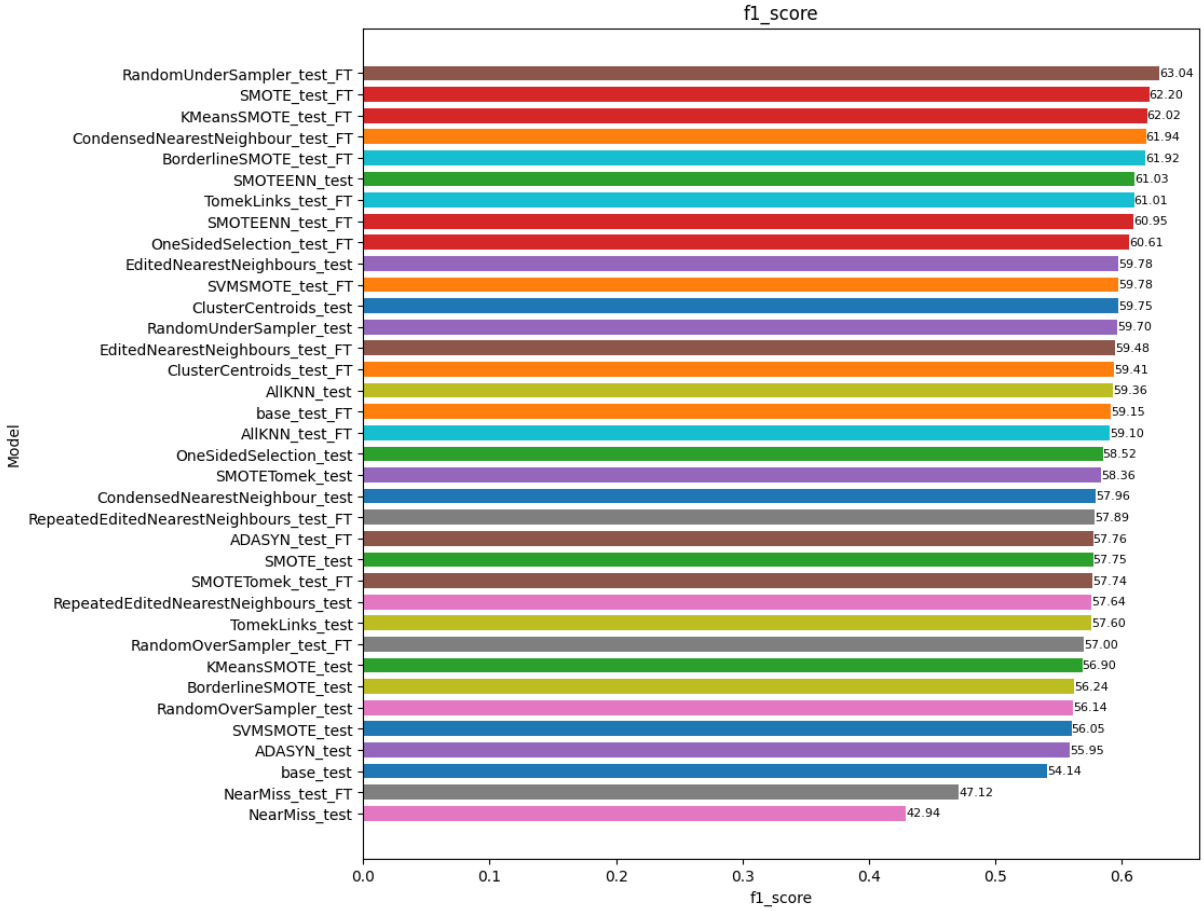
En este tercer anexo se exponen los gráficos de cada métrica calculada para todos los modelos siguiendo un orden de mayor a menor para ver fácilmente cuál de todos es el mejor en cada métrica. Cabe destacar que los modelos con parámetros ajustados aparecen con el mismo nombre, pero la nomenclatura FT al final (del término fine tuned). **Todos los gráficos que aparecen a continuación han sido elaborados por mí con la librería de Python Matplotib**:

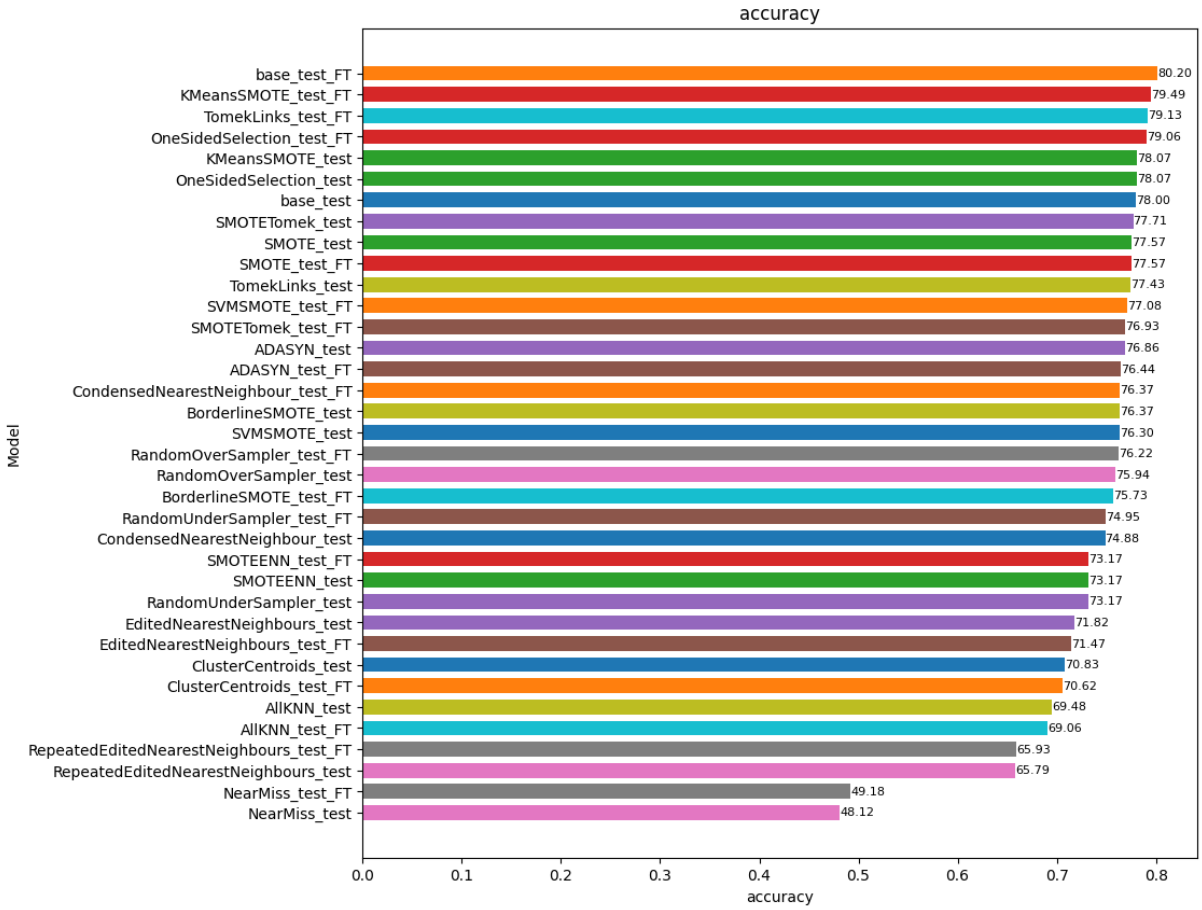

f2_score
RepeatedEditedNearestNeighbours_test_FT 72.94
RepeatedEditedNearestNeighbours_test 72.57
RandomUnderSampler_test_FT 72.46
AllKNN_test 72.02
AllKNN_test_FT 71.98
ClusterCentroids_test 71.16
SMOTEENN_test 70.75
ClusterCentroids_test_FT 70.73
SMOTEENN_test_FT 70.57
EditedNearestNeighbours_test 69.94
EditedNearestNeighbours_test_FT 69.77
BorderlineSMOTE_test_FT 68.81
RandomUnderSampler_test 67.96
CondensedNearestNeighbour_test_FT 67.85
SMOTE_test_FT 66.39
NearMiss_test_FT 64.42
KMeansSMOTE_test_FT 62.67
SVMSMOTE_test_FT 62.34
CondensedNearestNeighbour_test 62.12
TomekLinks_test_FT 61.30
OneSidedSelection_test_FT 60.66
ADASYN_test_FT 59.49
SMOTETomek_test_FT 58.70
SMOTETomek_test 58.64
RandomOverSampler_test_FT 58.39
OneSidedSelection_test 58.38
SMOTE_test 57.75
TomekLinks_test 57.69
RandomOverSampler_test 57.26
NearMiss_test 57.22
BorderlineSMOTE_test 56.82
SVMSMOTE_test 56.59
base_test_FT 55.96
ADASYN_test 55.59
KMeansSMOTE_test 55.46
base_test 50.89
Model
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7
f2_score

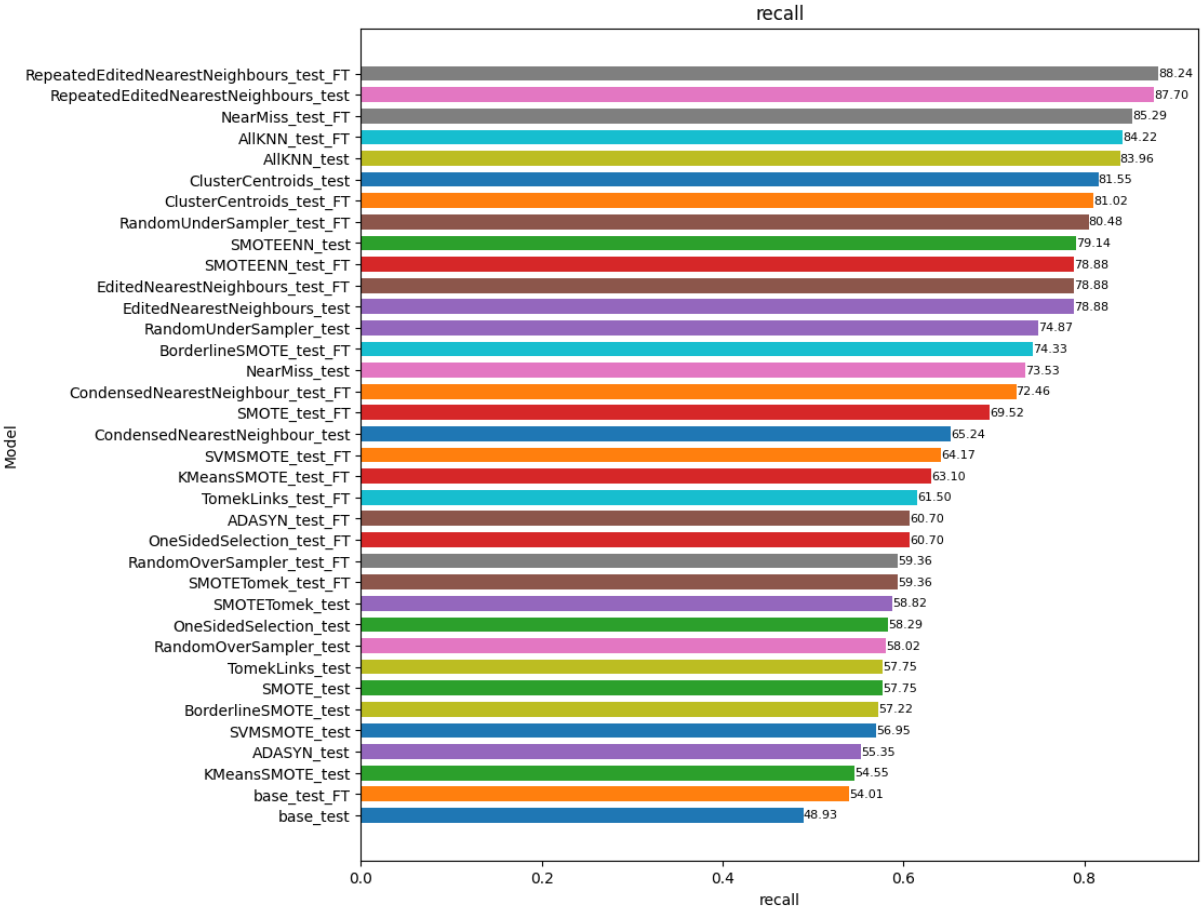

ROC_0
NearMiss_test_FT 43.87
NearMiss_test 42.83
RandomOverSampler_test 19.73
BorderlineSMOTE_test 19.60
SVMSMOTE_test 19.42
RandomOverSampler_test_FT 19.39
CondensedNearestNeighbour_test 19.36
ADASYN_test 19.32
TomekLinks_test 19.25
SMOTE_test 19.14
OneSidedSelection_test 19.14
ClusterCentroids_test 18.87
SMOTETomek_test 18.80
ClusterCentroids_test_FT 18.70
base_test 18.66
SMOTETomek_test_FT 18.61
RandomUnderSampler_test 18.61
ADASYN_test_FT 18.58
KMeansSMOTE_test 18.54
SVMSMOTE_test_FT 18.08
RepeatedEditedNearestNeighbours_test 17.81
BorderlineSMOTE_test_FT 17.62
RepeatedEditedNearestNeighbours_test_FT 17.45
SMOTEENN_test 17.28
EditedNearestNeighbours_test_FT 17.26
EditedNearestNeighbours_test 17.25
AllKNN_test 17.24
SMOTEENN_test_FT 16.96
AllKNN_test_FT 16.88
SMOTE_test_FT 16.74
TomekLinks_test_FT 16.64
CondensedNearestNeighbour_test_FT 16.48
base_test_FT 16.48
OneSidedSelection_test_FT 16.42
KMeansSMOTE_test_FT 16.36
RandomUnderSampler_test_FT 16.19
Model
0.0 0.1 0.2 0.3 0.4
ROC_0


ROC_1
RandomUnderSampler_test_FT 83.81
KMeansSMOTE_test_FT 83.64
OneSidedSelection_test_FT 83.58
CondensedNearestNeighbour_test_FT 83.52
base_test_FT 83.52
TomekLinks_test_FT 83.36
SMOTE_test_FT 83.26
AllKNN_test_FT 83.12
SMOTEENN_test_FT 83.04
AllKNN_test 82.76
EditedNearestNeighbours_test 82.75
EditedNearestNeighbours_test_FT 82.74
SMOTEENN_test 82.72
RepeatedEditedNearestNeighbours_test_FT 82.55
BorderlineSMOTE_test_FT 82.38
RepeatedEditedNearestNeighbours_test 82.19
SVMSMOTE_test_FT 81.92
KMeansSMOTE_test 81.46
ADASYN_test_FT 81.42
RandomUnderSampler_test 81.39
SMOTETomek_test_FT 81.39
base_test 81.34
ClusterCentroids_test_FT 81.30
SMOTETomek_test 81.20
ClusterCentroids_test 81.13
OneSidedSelection_test 80.86
SMOTE_test 80.86
TomekLinks_test 80.75
ADASYN_test 80.68
CondensedNearestNeighbour_test 80.64
RandomOverSampler_test_FT 80.61
SVMSMOTE_test 80.58
BorderlineSMOTE_test 80.40
RandomOverSampler_test 80.27
NearMiss_test 57.17
NearMiss_test_FT 56.13
Model
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
ROC_1


precision
base_test_FT 65.37
KMeansSMOTE_test_FT 60.98
base_test 60.60
OneSidedSelection_test_FT 60.53
TomekLinks_test_FT 60.53
KMeansSMOTE_test 59.48
OneSidedSelection_test 58.76
SMOTETomek_test 57.89
SMOTE_test 57.75
TomekLinks_test 57.45
ADASYN_test 56.56
SMOTE_test_FT 56.28
SMOTETomek_test_FT 56.20
SVMSMOTE_test_FT 55.94
BorderlineSMOTE_test 55.30
SVMSMOTE_test 55.18
ADASYN_test_FT 55.10
RandomOverSampler_test_FT 54.81
RandomOverSampler_test 54.39
CondensedNearestNeighbour_test_FT 54.09
BorderlineSMOTE_test_FT 53.05
CondensedNearestNeighbour_test 52.14
RandomUnderSampler_test_FT 51.81
SMOTEENN_test_FT 49.66
SMOTEENN_test 49.66
RandomUnderSampler_test 49.65
EditedNearestNeighbours_test 48.12
EditedNearestNeighbours_test_FT 47.73
ClusterCentroids_test 47.14
ClusterCentroids_test_FT 46.90
AllKNN_test 45.91
AllKNN_test_FT 45.52
RepeatedEditedNearestNeighbours_test_FT 43.08
RepeatedEditedNearestNeighbours_test 42.93
NearMiss_test_FT 32.55
NearMiss_test 30.32
Model
0.0 0.1 0.2 0.3 0.4 0.5 0.6
precision


f1_score
RandomUnderSampler_test_FT 63.04
SMOTE_test_FT 62.20
KMeansSMOTE_test_FT 62.02
CondensedNearestNeighbour_test_FT 61.94
BorderlineSMOTE_test_FT 61.92
SMOTEENN_test 61.03
TomekLinks_test_FT 61.01
SMOTEENN_test_FT 60.95
OneSidedSelection_test_FT 60.61
EditedNearestNeighbours_test 59.78
SVMSMOTE_test_FT 59.78
ClusterCentroids_test 59.75
RandomUnderSampler_test 59.70
EditedNearestNeighbours_test_FT 59.48
ClusterCentroids_test_FT 59.41
AllKNN_test 59.36
base_test_FT 59.15
AllKNN_test_FT 59.10
OneSidedSelection_test 58.52
SMOTETomek_test 58.36
CondensedNearestNeighbour_test 57.96
RepeatedEditedNearestNeighbours_test_FT 57.89
ADASYN_test_FT 57.76
SMOTE_test 57.75
SMOTETomek_test_FT 57.74
RepeatedEditedNearestNeighbours_test 57.64
TomekLinks_test 57.60
RandomOverSampler_test_FT 57.00
KMeansSMOTE_test 56.90
BorderlineSMOTE_test 56.24
RandomOverSampler_test 56.14
SVMSMOTE_test 56.05
ADASYN_test 55.95
base_test 54.14
NearMiss_test_FT 47.12
NearMiss_test 42.94
Model
0.0 0.1 0.2 0.3 0.4 0.5 0.6
f1_score


accuracy
base_test_FT 80.20
KMeansSMOTE_test_FT 79.49
TomekLinks_test_FT 79.13
OneSidedSelection_test_FT 79.06
KMeansSMOTE_test 78.07
OneSidedSelection_test 78.07
base_test 78.00
SMOTETomek_test 77.71
SMOTE_test 77.57
SMOTE_test_FT 77.57
TomekLinks_test 77.43
SVMSMOTE_test_FT 77.08
SMOTETomek_test_FT 76.93
ADASYN_test 76.86
ADASYN_test_FT 76.44
CondensedNearestNeighbour_test_FT 76.37
BorderlineSMOTE_test 76.37
SVMSMOTE_test 76.30
RandomOverSampler_test_FT 76.22
RandomOverSampler_test 75.94
BorderlineSMOTE_test_FT 75.73
RandomUnderSampler_test_FT 74.95
CondensedNearestNeighbour_test 74.88
SMOTEENN_test_FT 73.17
SMOTEENN_test 73.17
RandomUnderSampler_test 73.17
EditedNearestNeighbours_test 71.82
EditedNearestNeighbours_test_FT 71.47
ClusterCentroids_test 70.83
ClusterCentroids_test_FT 70.62
AllKNN_test 69.48
AllKNN_test_FT 69.06
RepeatedEditedNearestNeighbours_test_FT 65.93
RepeatedEditedNearestNeighbours_test 65.79
NearMiss_test_FT 49.18
NearMiss_test 48.12
Model
0.0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8
accuracy


recall
RepeatedEditedNearestNeighbours_test_FT 88.24
RepeatedEditedNearestNeighbours_test 87.70
NearMiss_test_FT 85.29
AllKNN_test_FT 84.22
AllKNN_test 83.96
ClusterCentroids_test 81.55
ClusterCentroids_test_FT 81.02
RandomUnderSampler_test_FT 80.48
SMOTEENN_test 79.14
SMOTEENN_test_FT 78.88
EditedNearestNeighbours_test_FT 78.88
EditedNearestNeighbours_test 78.88
RandomUnderSampler_test 74.87
BorderlineSMOTE_test_FT 74.33
NearMiss_test 73.53
CondensedNearestNeighbour_test_FT 72.46
SMOTE_test_FT 69.52
CondensedNearestNeighbour_test 65.24
SVMSMOTE_test_FT 64.17
KMeansSMOTE_test_FT 63.10
TomekLinks_test_FT 61.50
ADASYN_test_FT 60.70
OneSidedSelection_test_FT 60.70
RandomOverSampler_test_FT 59.36
SMOTETomek_test_FT 59.36
SMOTETomek_test 58.82
OneSidedSelection_test 58.29
RandomOverSampler_test 58.02
TomekLinks_test 57.75
SMOTE_test 57.75
BorderlineSMOTE_test 57.22
SVMSMOTE_test 56.95
ADASYN_test 55.35
KMeansSMOTE_test 54.55
base_test_FT 54.01
base_test 48.93
Model
0.0
0.2
0.4
0.6
0.8
recall

ANEXO IV: TABLAS DE MÉTRICAS EN DETALLE

Por último, en este cuarto anexo se adjuntan todas las tablas de métricas de elaboración propia que exponen los resultados de los modelos sobre ambos conjuntos:

Tabla de métricas del modelo base

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	Base	0.0002	0.9933	0.9998	0.9960	0.9943	0.9970	0.9926
	Base Ajustado	0.0489	0.7063	0.9511	0.7961	0.7375	0.8703	0.6870
Test	Base	0.1866	0.5089	0.8134	0.6060	0.5414	0.7800	0.4893
	Base Ajustado	0.1648	0.5596	0.8352	0.6537	0.5915	0.8020	0.5401

RESULTADOS OBTENIDOS CON LAS TÉCNICAS DE OVERSAMPLING

Tabla de métricas con Random Oversampler

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	Random Oversampler	0.0002	0.9953	0.9998	0.9900	0.9933	0.9965	0.9967
	Random Oversampler Ajustado	0.0001	0.9977	0.9999	0.9942	0.9964	0.9964	0.9986
Test	Random Oversampler	0.1973	0.5726	0.8027	0.5439	0.5614	0.7594	0.5802
	Random Oversampler Ajustado	0.1939	0.5839	0.8061	0.5481	0.5700	0.7622	0.5936

Tabla de métricas con SMOTE

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	SMOTE	0.0003	0.9948	0.9997	0.9927	0.9940	0.9968	0.9953
	SMOTE Ajustado	0.0481	0.9151	0.9519	0.8579	0.8928	0.8883	0.9307
Test	SMOTE	0.1914	0.5775	0.8086	0.5775	0.5775	0.7757	0.5775
	SMOTE Ajustado	0.1674	0.6639	0.8326	0.5628	0.6220	0.7757	0.6952

Tabla de métricas con ADASYN

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	ADASYN	0.0002	0.9952	0.9998	0.9920	0.9940	0.9968	0.9960
	ADASYN Ajustado	0.0006	0.9904	0.9994	0.9565	0.9774	0.9771	0.9993
Test	ADASYN	0.1932	0.5559	0.8068	0.5656	0.5595	0.7686	0.5535
	ADASYN Ajustado	0.1858	0.5949	0.8142	0.5510	0.5776	0.7644	0.6070

Tabla de métricas con Borderline-SMOTE

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	Borderline-SMOTE	0.0003	0.9952	0.9997	0.9920	0.9940	0.9968	0.9960
	Borderline-SMOTE Ajustado	0.0488	0.9270	0.9512	0.8296	0.8879	0.8794	0.9551
Test	Borderline-SMOTE	0.1960	0.5682	0.8040	0.5530	0.5624	0.7637	0.5722
	Borderline-SMOTE Ajustado	0.1762	0.6881	0.8238	0.5305	0.6192	0.7573	0.7433

Tabla de métricas con SVMSMOTE

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	SVMSMOTE	0.0003	0.9952	0.9997	0.9920	0.9940	0.9968	0.9960
	SVMSMOTE Ajustado	0.0022	0.9859	0.9978	0.9360	0.9666	0.9655	0.9993
Test	SVMSMOTE	0.1942	0.5659	0.8058	0.5518	0.5605	0.7630	0.5695
	SVMSMOTE Ajustado	0.1808	0.6234	0.8192	0.5594	0.5978	0.7708	0.6417

Tabla de métricas con KMeansSMOTE

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	KMeansSMOTE	0.0002	0.9936	0.9998	0.9946	0.9940	0.9968	0.9933
	KMeansSMOTE Ajustado	0.0349	0.9056	0.9651	0.8858	0.8980	0.8966	0.9106
Test	KMeansSMOTE	0.1854	0.5546	0.8146	0.5948	0.5690	0.7807	0.5455
	KMeansSMOTE Ajustado	0.1636	0.6267	0.8364	0.6098	0.6202	0.7949	0.6310

RESULTADOS OBTENIDOS CON LAS TÉCNICAS DE UNDERSAMPLING

Tabla de métricas con RandomUndersampler

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	RandomUndersampler	0.0287	0.9141	0.9713	0.6841	0.8118	0.8772	0.9980
	RandomUndersampler Ajustado	0.0427	0.9303	0.9573	0.8551	0.9006	0.8950	0.9512
Test	RandomUndersampler	0.1861	0.6796	0.8139	0.4965	0.5970	0.7317	0.7487
	RandomUndersampler Ajustado	0.1619	0.7246	0.8381	0.5181	0.6304	0.7495	0.8048

Tabla de métricas con NearMiss

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	NearMiss	0.1458	0.7992	0.8542	0.4475	0.6173	0.6727	0.9946
	NearMiss Ajustado	0.1054	0.8451	0.8946	0.7343	0.7999	0.7803	0.8783
Test	NearMiss	0.4283	0.5722	0.5717	0.3032	0.4294	0.4812	0.7353
	NearMiss Ajustado	0.4387	0.6442	0.5613	0.3255	0.4712	0.4918	0.8529

Tabla de métricas con TomekLinks

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	TomekLinks	0.0144	0.9643	0.9856	0.8634	0.9238	0.9565	0.9933
	TomekLinks Ajustado	0.0386	0.7815	0.9614	0.8288	0.7986	0.8895	0.7706
Test	TomekLinks	0.1925	0.5769	0.8075	0.5745	0.5760	0.7743	0.5775
	TomekLinks Ajustado	0.1664	0.6130	0.8336	0.6053	0.6101	0.7913	0.6150

Tabla de métricas con ClusterCentroids

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	ClusterCentroids	0.0713	0.8668	0.9287	0.5654	0.7224	0.7961	1.0000
	ClusterCentroids Ajustado	0.0209	0.9192	0.9791	0.9197	0.9194	0.9194	0.9191
Test	ClusterCentroids	0.1887	0.7116	0.8113	0.4714	0.5975	0.7083	0.8155
	ClusterCentroids Ajustado	0.1870	0.7073	0.8130	0.4690	0.5941	0.7062	0.8102

Tabla de métricas con OneSidedSelection

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	OneSidedSelection	0.0146	0.9646	0.9854	0.8688	0.9263	0.9581	0.9920
	OneSidedSelection Ajustado	0.0376	0.7820	0.9624	0.8347	0.8010	0.8911	0.7699
Test	OneSidedSelection	0.1914	0.5838	0.8086	0.5876	0.5852	0.7807	0.5829
	OneSidedSelection Ajustado	0.1642	0.6066	0.8358	0.6053	0.6061	0.7906	0.6070

Tabla de métricas con EditedNearestNeighbours

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	EditedNearestNeighbours	0.0744	0.8764	0.9256	0.5883	0.7404	0.8142	0.9987
	EditedNearestNeighbours Ajustado	0.0000	0.9992	1.0000	0.9987	0.9990	0.9992	0.9993
Test	EditedNearestNeighbours	0.1725	0.6994	0.8275	0.4812	0.5978	0.7182	0.7888
	EditedNearestNeighbours Ajustado	0.1726	0.6977	0.8274	0.4773	0.5948	0.7147	0.7888

Tabla de métricas con RepeatedEditedNearestNeighbours

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	RepeatedEditedNearestNeighbours	0.1115	0.8176	0.8885	0.4727	0.6419	0.7039	1.0000
	RepeatedEditedNearestNeighbours Ajustado	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Test	RepeatedEditedNearestNeighbours	0.1781	0.7257	0.8219	0.4293	0.5764	0.6579	0.8770
	RepeatedEditedNearestNeighbours Ajustado	0.1745	0.7294	0.8255	0.4308	0.5789	0.6593	0.8824

Tabla de métricas con AllKNN

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	AllKNN	0.0947	0.8500	0.9053	0.5313	0.6939	0.7659	1.0000
	AllKNN Ajustado	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Test	AllKNN	0.1724	0.7202	0.8276	0.4591	0.5936	0.6948	0.8396
	AllKNN Ajustado	0.1688	0.7198	0.8312	0.4552	0.5910	0.6906	0.8422

Tabla de métricas con CondensedNearestNeighbour

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	CondensedNearestNeighbour	0.0075	0.9573	0.9925	0.8269	0.9039	0.9437	0.9967
	CondensedNearestNeighbour Ajustado	0.1215	0.8333	0.8785	0.7766	0.8111	0.7904	0.8488
Test	CondensedNearestNeighbour	0.1936	0.6212	0.8064	0.5214	0.5796	0.7488	0.6524
	CondensedNearestNeighbour Ajustado	0.1648	0.6785	0.8352	0.5409	0.6194	0.7637	0.7246

RESULTADOS OBTENIDOS CON LA MEZCLA DE AMBOS TIPOS DE TÉCNICAS

Tabla de métricas con SMOTEENN

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	SMOTEENN	0.1070	0.8069	0.8930	0.5745	0.7006	0.7964	0.8977
	SMOTEENN Ajustado	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Test	SMOTEENN	0.1728	0.7075	0.8272	0.4966	0.6103	0.7317	0.7914
	SMOTEENN Ajustado	0.1696	0.7057	0.8304	0.4966	0.6095	0.7317	0.7888

Tabla de métricas con SMOTETomek

Conjunto	Modelo	ROC 0	F2-Score	ROC 1	Precisión	F1 - Score	Accuracy	Recall
Entrenamiento	SMOTETomek	0.0159	0.9431	0.9841	0.9173	0.9333	0.9640	0.9498
	SMOTETomek Ajustado	0.0002	0.9960	0.9998	0.9850	0.9918	0.9918	0.9987
Test	SMOTETomek	0.1880	0.5864	0.8120	0.5789	0.5836	0.7771	0.5882
	SMOTETomek Ajustado	0.1861	0.5870	0.8139	0.5620	0.5774	0.7693	0.5936

8. BIBLIOGRAFÍA

Amazon Web Services (2024) *"¿Qué es el sobreajuste? - Explicación del sobreajuste en machine learning – AWS*". Disponible en: https://aws.amazon.com/es/what-is/overfitting/ [Consultado: 2 de junio, 2024]. (Volver al apartado)

Analytics Vidhya (2020). "*A Simple Explanation of K-means Clustering*". Disponible en: https://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/ [Consultado: 27 de mayo de 2024]. (Volver al apartado)

Analytics Vidhya (2020). "*Overcoming Class Imbalance using SMOTE Techniques".* Disponible en: https://editor.analyticsvidhya.com/uploads/77417image1.png [Consultado: 23 de mayo de 2024]. (Volver al apartado)

CustomerGauge. (18 de abril, 2024). What's the Average Churn Rate by Industry? Disponible en: https://customergauge.com/blog/average-churn-rate-by-industry [Consultado: 23 de mayo, 2024]. (Volver al apartado)

DataScientest (2024) "*Kaggle: todo lo que hay que saber sobre esta plataforma*". Disponible en: https://datascientest.com/es/kaggle-todo-lo-que-hay-que-saber-sobre-esta-plataforma [Consultado: 2 junio 2024]. (Volver al apartado)

DataSource.ai (2024) "*Métricas De Evaluación De Modelos En El Aprendizaje Automático*". Disponible en: https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico [Consultado: 2 junio 2024]. (Volver al apartado)

D. Wilson (1972) "*Asymptotic Properties of Nearest Neighbor Rules Using Edited Data*" In IEEE Transactions on Systems, Man, and Cybernetrics, vol. 2 (3), pp. 408-421. [Consultado: 23 de mayo, 2024]. (Volver al apartado)

Eick, Christoph & Zeidat, N. & Vilalta, Ricardo. (2004). "*Using Representative-Based Clustering for Nearest Neighbor Dataset Editing*" - Scientific Figure on ResearchGate. Disponible en: https://www.researchgate.net/figure/Wilson-Editing-for-a-1-NN-Classifier_fig1_4133603. pp. 375 - 378. 10.1109/ICDM.2004.10044. [Consultado: 27 de mayo, 2024]. (Volver al apartado)

Georgios Douzas, Fernando Bacao, Felix Last (2018), "*Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE*", Information Sciences, Volume 465, pp. 1-20. [Consultado: 23 de mayo, 2024]. (Volver al apartado)

G. Batista, B. Bazzan, M. Monard (2003), "Balancing Training Data for Automated Annotation of Keywords: A Case Study" In WOB, pp. 10-18. [Consultado: 23 de mayo, 2024]. (Volver al apartado)

G. Batista, R. C. Prati, M. C. Monard (2004). "*A study of the behavior of several methods for balancing machine learning training data*" ACM Sigkdd Explorations Newsletter 6 (1), pp. 20-29. [Consultado: 23 de mayo, 2024]. (Volver al apartado)

G. Menardi, N. Torelli (2014), "*Training and assessing classification rules with imbalanced data*" Data Mining and Knowledge Discovery, 28(1), pp.92-122. [Consultado: 23 de mayo, 2024]. (Volver al apartado)

Hassannataj Joloudari, J., Marefat, A., Nematollahi, M.A., Oyelere, S.S. & Hussain, S. (2022). "*Effective Class-Imbalance learning based on SMOTE and Convolutional Neural Networks*". pp. 7. Disponible en: https://www.researchgate.net/publication/363269818_Effective_Class-Imbalance_learning_based_on_SMOTE_and_Convolutional_Neural_Networks [Consultado: 27 May 2024]. (Volver al apartado)

He, Haibo, Yang Bai, Edwardo A. Garcia, and Shutao Li (2008). "*ADASYN: Adaptive synthetic sampling approach for imbalanced learning*" In IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), pp. 1322-1328. [Consultado: 23 de mayo, 2024]. (Volver al apartado)

Huilgol, P. (2024) "*Precision and Recall | Essential Metrics for Machine Learning*". Disponible en: https://www.analyticsvidhya.com/blog/2020/09/precision-recall-machine-learning/ [Consultado: 30 de mayo, 2024].

H. Han, W. Wen-Yuan, M. Bing-Huan (2005), "*Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning*" Advances in intelligent computing, 878-887. [Consultado: 25 de mayo, 2024]. (Volver al apartado)

H. M. Nguyen, E. W. Cooper, K. Kamei (2009) "*Borderline over-sampling for imbalanced data classification*" International Journal of Knowledge Engineering and Soft Data Paradigms, 3(1), pp.4-21. [Consultado: 26 de mayo, 2024]. (Volver al apartado)

IBM. (2024) "*What is a Random Forest*". Disponible en: https://www.ibm.com/topics/random-forest [Consultado: 2 junio 2024]. (Volver al apartado)

Marco Sanjuán, F. J., López, J. F. (2021) "*Outlier - Qué es, definición y concepto*". Disponible en: https://economipedia.com/definiciones/outlier.html [Consultado: 2 junio 2024]. (Volver al apartado)

Medium (2021). "*Undersampling and oversampling: An old and a new approach.* Disponible en: https://miro.medium.com/v2/resize:fit:725/1*7xf9e1EaoK5n05izIFBouA.png [Consultado: 20 de mayo, 2024]. (Volver al apartado)

M. Kubat, S. Matwin (1997). "*Addressing the curse of imbalanced training sets: one-sided selection*" In ICML, vol. 97, pp. 179-186. [Consultado: 28 de mayo, 2024]. (Volver al apartado)

Na8 (2019) "*Clasificación con datos desbalanceados*". Disponible en: https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/ [Consultado: 2 de junio, 2024]. (Volver al apartado)

N. V. Chawla, K. W. Bowyer, L. O. Hall, W. P. Kegelmeyer (2002), "*SMOTE: synthetic minority over-sampling technique*" Journal of artificial intelligence research, 321-357. [Consultado: 28 de mayo, 2024]. (Volver al apartado)

Oliver Wyman (2023). "*Analyzing Churn Challenges In The European Telecom Landscape"*. Disponible en: https://www.oliverwyman.com/our-expertise/insights/2023/oct/european-telecoms-churn-trends.html [Consultado: 23 de mayo de 2024]. (Volver al apartado)

Pei, Xin & Mei, Fei & Gu, Jiaqi. (2022). "*The real-time state identification of the electricity-heat system based on Borderline-SMOTE and XGBoost*" - Scientific Figure on ResearchGate. Disponible en: https://www.researchgate.net/figure/Schematic-diagram-of-Borderline-SMOTE-algorithm-principle_fig4_362631677 [Consultado: 6 Jun 2024] (Volver al apartado)

Sanahuja, P. M. (2021) "*Entendiendo la curva ROC y el AUC: Dos medidas del rendimiento de un clasificador binario que van de la mano*". Disponible en: https://polmartisanahuja.com/entendiendo-la-curva-roc-y-el-auc-dos-medidas-del-rendimiento-de-un-clasificador-binario-que-van-de-la-mano/ [Consultado: 2 junio 2024]. (Volver al apartado)

Towards Data Science (2019). "*K-means: A Complete Introduction*". Disponible en: https://towardsdatascience.com/k-means-a-complete-introduction-1702af9cd8c [Consultado: 27 mayo 2024]. (Volver al apartado)