CUNEF UNIVERSIDAD

MÁSTER EN CIENCIA DE DATOS

RESUMEN EJECUTIVO TFM

EFECTO DE LAS TÉCNICAS DE SAMPLEO EN PROBLEMAS DE CLASIFICACIÓN DESBALANCEADOS

Over y undersampling aplicado a churn en empresas de telecomunicación

Autor: Blanco García, Ángel

Tutora: Arévalo Barco, Irina

Madrid, junio de 2024

Índice

1. INTRODUCCIÓN

El churn es un problema crítico para las empresas, especialmente aquellas con modelos de negocio basados en suscripciones. Un estudio de CustomerGauge en 2023 revela que las tasas de abandono para empresas B2B y B2C son del 23% y 31%, respectivamente. En el competitivo mercado de operadores móviles en Europa, otro estudio, este de Oliver Wyman, indica que el 44% de los consumidores están dispuestos a cambiar de proveedor. Este trabajo utiliza modelos de machine learning para predecir el churn, enfrentando el desafío del desbalanceo de clases, donde hay más datos sobre clientes que permanecen que sobre los que abandonan, lo que normalmente lleva a los modelos a sesgar sus predicciones hacia la clase sobrerrepresentada y hace que las métricas de rendimiento no sean fiables. 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.

En el segundo 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 una empresa de telecomunicaciones aplicando varias técnicas y evaluando su rendimiento mediante el modelo Random Forest y métricas específicas como el F2-Score. Finalmente, se extraen conclusiones sobre el estudio realizado.

2. ANÁLISIS TEÓRICO DE LAS TÉCNICAS DE BALANCEO

Este capítulo desarrolla teóricamente las principales técnicas de resampling y sus combinaciones.

2.1 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. Un resumen de cada técnica aplicada en el trabajo sería el siguiente:

- **Random Oversampler**: duplica aleatoriamente ejemplos de la clase minoritaria. Es simple, pero puede llevar al sobreajuste al no introducir nueva información.

- **SMOTE** (Synthetic Minority Over-sampling Technique): genera ejemplos sintéticos basados en la interpolación entre ejemplos minoritarios y sus vecinos cercanos. Introduce variabilidad y mejora la generalización del modelo.

- **ADASYN** (Adaptive Synthetic Sampling): similar a SMOTE, pero genera más ejemplos sintéticos para las instancias minoritarias difíciles de aprender.

- **Borderline-SMOTE**: se enfoca en las instancias minoritarias cercanas a la frontera de decisión entre clases, mejorando la capacidad del modelo para manejar estas áreas.

- **SVMSMOTE**: utiliza Support Vector Machines para identificar ejemplos minoritarios cerca de la frontera de decisión y generar ejemplos sintéticos en esas áreas.

- **KMeansSMOTE**: combina K-Means clustering con SMOTE para generar ejemplos sintéticos dentro de cada clúster, capturando mejor la estructura interna de la clase minoritaria.

A continuación, se explicarán las principales técnicas de undersampling.

2.2 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. Un resumen de cada técnica aplicada en el trabajo sería el siguiente:

- **Random Undersampler**: elimina aleatoriamente ejemplos de la clase mayoritaria. Es fácil de implementar, pero puede perder información valiosa.

- **NearMiss**: selecciona ejemplos mayoritarios para eliminar basándose en su proximidad a la clase minoritaria, con variantes como NearMiss-1, NearMiss-2 y NearMiss-3 que utilizan diferentes criterios de selección.

- **TomekLinks**: identifica y elimina pares de instancias de diferentes clases que son vecinos más cercanos entre sí, mejorando la separación entre clases.

- **ClusterCentroids**: utiliza clustering (como K-Means) para agrupar la clase mayoritaria y reemplazar instancias por los centroides de los clústeres.

- **OneSidedSelection**: combina la eliminación de ruido (Tomek Links) con submuestreo (Condensed Nearest Neighbor) para limpiar y equilibrar el conjunto de datos.

- **EditedNearestNeighbours** (ENN): elimina instancias mal clasificadas basándose en sus vecinos más cercanos, reduciendo el ruido.

- **RepeatedEditedNearestNeighbours** (RENN): repite el proceso de ENN hasta que no se detectan instancias ruidosas.

- **AllKNN**: similar a ENN, pero varía el número de vecinos en cada iteración hasta alcanzar un criterio de convergencia.

- **CondensedNearestNeighbour** (CNN): selecciona un subconjunto representativo de las instancias originales preservando los puntos críticos para la correcta clasificación.

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

2.3 COMBINACIÓN DE TÉCNICAS

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. A continuación, el resumen de ambas combinaciones aplicadas en la investigación:

- **SMOTEENN**: combina SMOTE para oversampling y ENN para undersampling, aumentando la representación de la clase minoritaria y limpiando el ruido.
- **SMOTETomek**: integra SMOTE con Tomek Links, generando ejemplos sintéticos y eliminando aquellos que están demasiado cerca de la frontera entre clases.

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

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

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.

3.1 PREPARACIÓN DE LOS DATOS, SELECCIÓN DE MODELO Y MÉTRICAS

El conjunto de datos proviene de Kaggle e incluye información contractual y demográfica de **7,043 clientes** con **22 variables**. Se llevaron a cabo los pertinentes preprocesamientos de variables explicados en detalle en la memoria. Además, se comprobó que efectivamente el conjunto de datos estaba desbalanceado, teniendo un **73.46% de datos sobre clientes que no abandonan** y **solo un 26.54% de datos sobre clientes que sí** lo han hecho.

Se optó por el modelo Random Forest debido a su eficacia en manejar datos no lineales y complejos, así como su robustez frente a outliers y ruido.

Y se eligió el F2-Score como métrica principal por su enfoque en el recall, crucial para minimizar falsos negativos. Desde una perspectiva empresarial, es más perjudicial no identificar a los clientes que se marcharán, comparado con realizar esfuerzos hacia aquellos que no abandonarán. También se consideraron otras métricas como precisión, exactitud y AUC-ROC.

Por último, se incorporó la librería MLFlow para gestionar y comparar resultados de modelos, asegurando reproducibilidad y facilitando la colaboración.

Completado el EDA y seleccionados el modelo y las métricas, se iniciaron los experimentos.

3.2 MODELO BASE

Primero se entrenó un modelo base sin técnicas de resampling ni ajustar parámetros y luego otro ajustando parámetros con Grid Search (búsqueda de los mejores parámetros para el modelo teniendo en cuenta una cuadrícula de parámetros dada). Su objetivo es

servir de comparación a los otros modelos. Los resultados muestran un rendimiento pobre debido al desbalanceo de clases, con un F2-Score de 0.5596 y un recall de 0.5401 en el conjunto de prueba ajustado.

Estos son los resultados de los modelos base:

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**

Veamos los resultados tras aplicar las técnicas al conjunto de datos.

3.3 RESULTADOS CON TÉCNICAS DE OVERSAMPLING

Los modelos con técnicas de oversampling como SMOTE, ADASYN y Borderline-SMOTE mejoran significativamente las métricas en comparación con el modelo base. El mejor rendimiento se obtiene con Borderline-SMOTE ajustado, mostrando un F2-Score de 68.9% y un recall de 74.4%. Esto se logra generando ejemplos sintéticos en las áreas de frontera entre las clases.

He aquí los resultados de los modelos tras aplicar las técnicas de oversampling:

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

Seguidamente, los resultados tras aplicar undersampling al conjunto de datos.

3.4 RESULTADOS CON TÉCNICAS DE UNDERSAMPLING

Las técnicas de undersampling, especialmente Repeated Edited Nearest Neighbour (RENN), muestran un mejor rendimiento y eficiencia.

El modelo RENN sin ajustar obtiene un F2-Score de 72.57% y un recall de 87.7%, superando al modelo base y siendo más eficiente en tiempo de ejecución. Esto se logra eliminando instancias ruidosas y mal clasificadas.

A continuación, los resultados de los modelos con técnicas de undersampling:

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

Por último, las métricas obtenidas al combinar ambas técnicas se muestran a continuación.

3.5 RESULTADOS CON COMBINACIÓN DE TÉCNICAS

La combinación SMOTEENN ofrece buenos resultados, con un F2-Score de 70.75% y un recall de 79.14%, siendo más eficiente que su versión ajustada. SMOTETomek también muestra mejoras, pero no tan significativas como SMOTEENN.

Los resultados obtenidos fueron los siguientes:

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

En el siguiente y último apartado, se desarrollan las conclusiones de la investigación.

4. 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.

Para más detalles: consulta la memoria del TFM y el repositorio de Github del ANEXO I.