UNIVERSIDAD DE GRANADA

Clasificación no balanceada: **SMOTE & ADASYN**

Tratamiento Inteligente de Datos

Máster universitario en Ingeniería Informática

Autor
Pablo Valenzuela Álvarez (pvalenzuela@correo.ugr.es)

Índice

1. Introducción

El problema de las distribuciones de datos que no están balanceados es uno de los típicos que aparecen siempre en las disciplinas de aprendizaje automático y minería de datos. Se da cuando una de las clases está sensiblemente más representada que el resto, y por tanto, puede conducir a aprendizajes sesgados hacia la clase con mayor datos de interés.

A la hora de medir la efectividad del modelo, nos puede engañar la alta cantidad de aciertos que presenta. Solo tiene en cuenta a la clase mayoritaria por lo que genera la falsa sensación de que funciona bien.

1.1. Ejemplo práctico

Para el ejemplo vamos a usar un conjunto de datos usado para identificar transacciones fraudulentas hechas con tarjetas de crédito [2]. El conjunto consta de cerca de 285000 casos de los cuales sólo 500 representan a la clase “fraude”. La distribución de esta clase no llega ni al 0,2% del total, distribución que está reflejada en la imagen siguiente.

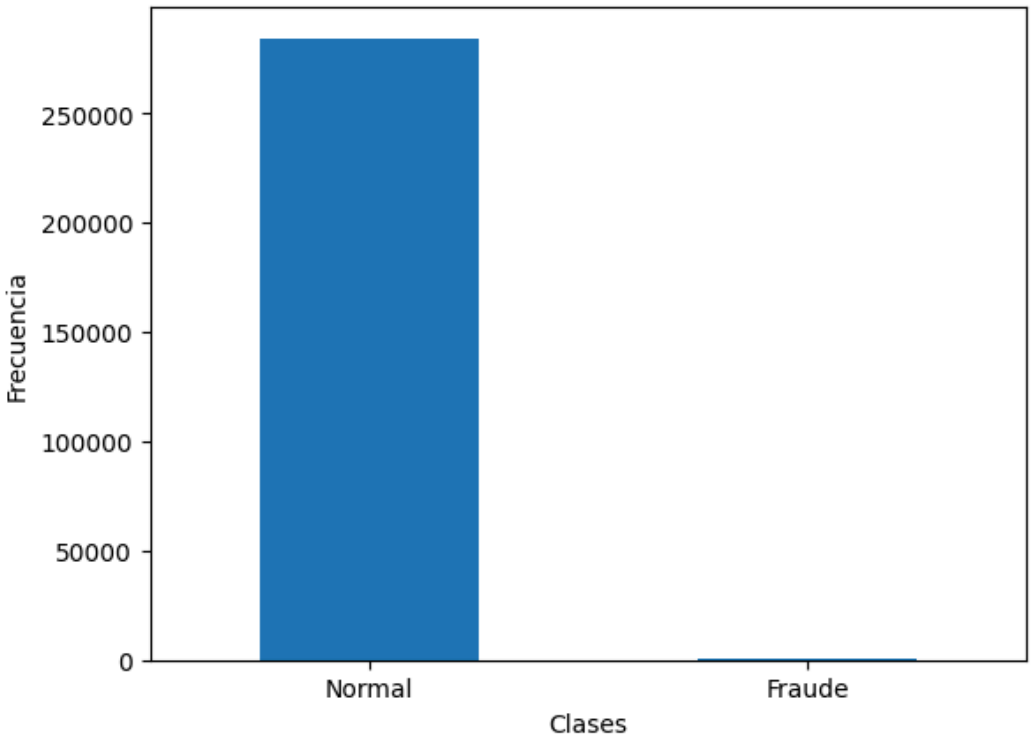

Figura 1: Distribución de las clases.

Si intentamos entrenar un modelo a partir de estos datos, los resultados que obtendremos nos pueden llevar a engaño.

Clase	Precisión	Recall	F1-score	Soporte
Normal	0.9992	0.9998	0.9995	85283
Fraude	0.8716	0.5938	0.7063	160
			Accuracy	0.9991	85443

Tabla 1: Resultados de la predicción

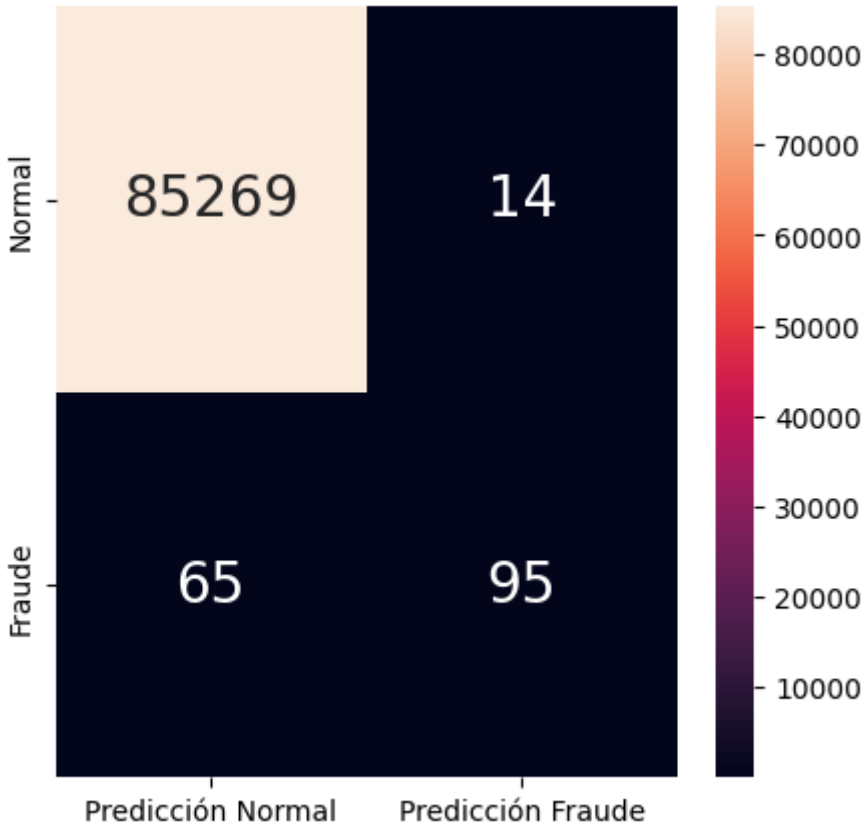

Figura 2: Matriz de confusión

Muestra un **accuracy** extremadamente alto, pero fijándonos bien en los resultados de la tabla 1 y apoyándonos en la figura 2, podemos observar que el **recall** correspondiente con la clase “fraude” es bastante bajo. Esto indica que no identifica bien cuando se da el caso de fraude, por lo que con un conjunto de test “real” los resultados obtenidos pueden ser peores.

1.2. Conjunto de test balanceado

Si probamos con un conjunto de test que tenga los mismos casos para las dos variables objetivos obtendremos resultados parecidos a los de la tabla 2 y figura 3.

Clase	Precisión	Recall	F1-score	Soporte
Normal	0.7143	1.000	0.8333	150
Fraude	1.000	0.6000	0.7500	150
		Accuracy	0.8000	300

Tabla 2: Resultados de la predicción del test balanceado

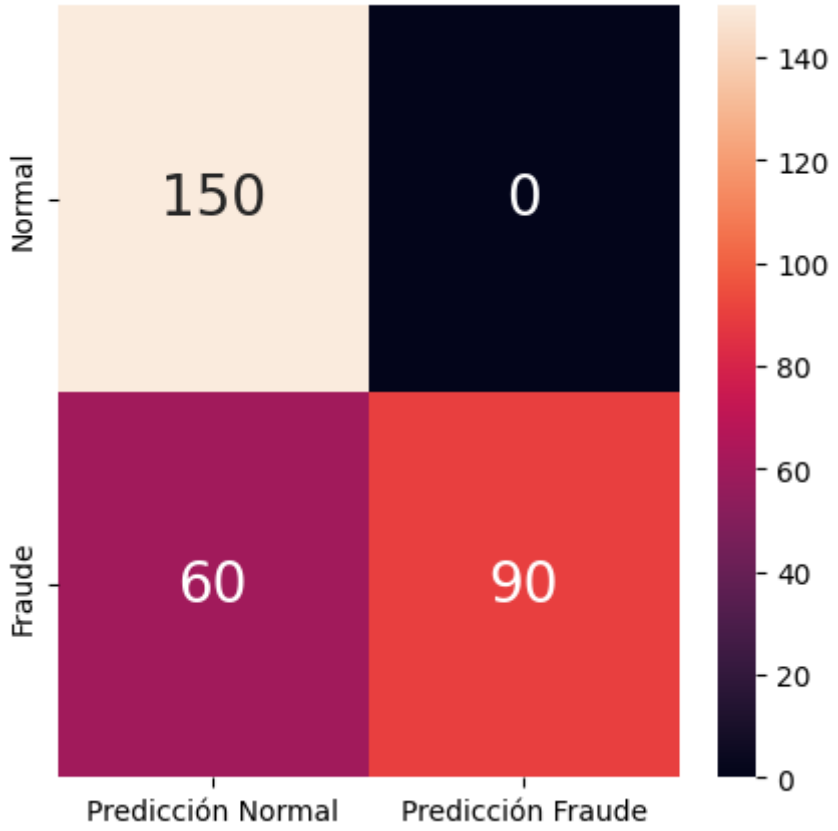

Figura 3: Matriz de confusión con test balanceado

Podemos ver que identifica a la perfección la clase “Normal”, pero sigue teniendo los mismos problemas con la clase “Fraude”, además que obtenemos peor **accuracy**.

1.3. Soluciones al problema

Hay algunas estrategias para enfrentarnos a este tipo de datos.

1. Submuestreo: Reduce el número de casos de la clase mayoritaria (**Tomek**)
2. Sobremuestreo: Aumenta el número de casos de la clase minoritaria (**SMOTE**, **ADASYN**).
3. Uso de algoritmos de algoritmos pensados para conjuntos de datos desbalanceados, como por ejemplo: **BalanceRandomForestClassifier** o **BalanceBaggingClassifier**.
4. Validación cruzada estratificada: donde cada porción tenga una proporción similar de instancias de ambas clases.

En nuestro caso, vamos a estudiar las técnicas de sobremuestreo mencionadas en el punto 2 de la lista anterior: **SMOTE** y **ADASYN**.

2. SMOTE

El significado de **SMOTE** es *Synthetic Minority Over-sampling Technique,* fue presentada en 2002 por **N. V. Chawla**, **K. W. Bowyer**, **L. O. Hall**, **W. P. Kegelmeyer** [9]. Como su nombre indica, es una técnica de sobre-muestreo. Esta técnica interpola nuevas instancias entre las ya existentes de la clase minoritaria.

Estas nuevas instancias se alinean entre los agrupamientos que se va encontrando de la clase minoritaria, de esta forma va añadiendo poco a poco nuevos casos en el espacio de características.

2.1. Funcionamiento

1. Selecciona la clase con menos instancias.
2. Encuentra un número "***k***" de agrupamientos de vecinos de esta clase.
3. Selecciona uno de estos agrupamientos al azar.
4. Genera nuevas instancias entre el agrupamiento escogido y los demás de la clase minoritaria. Se usa esta fórmula: $C = A + lambda * (B - A)$, donde A es el punto de inicio, B el destino y lambda se mueve entre 0 y 1.
5. Repetir desde el paso 1, hasta igualar el número de instancias de ambas clases

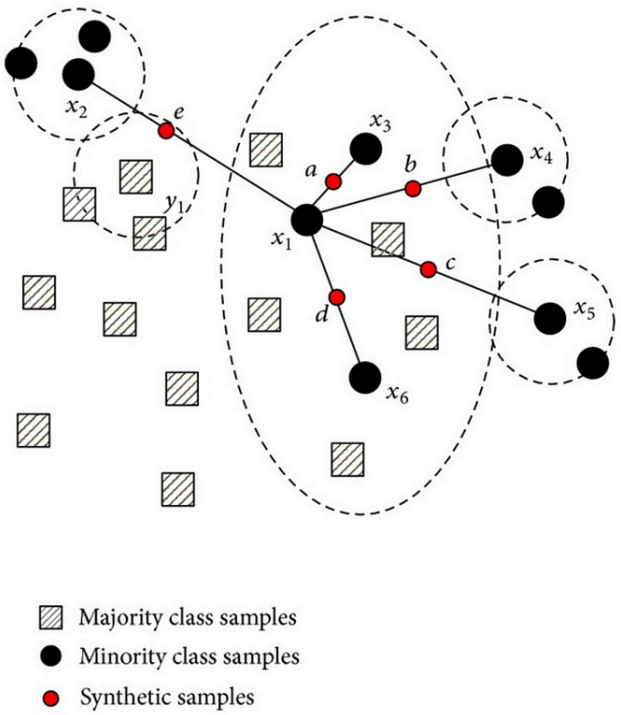

Figura 4: Funcionamiento SMOTE
(https://www.researchgate.net/publication/287601878/figure/fig1/AS:316826589384744@1452548753581/The-schematic-of-NRSBoundary-SMOTE-algorithm.png)

2.2. Beneficios

- Nos ayuda a balancear los datos.
- Reduce el sesgo sobre la clase mayoritaria.
- Fácil de implementar.

2.3. Limitaciones

- No considera la calidad de las instancias. Al generar nuevas instancias linealmente entre las demás muestras de la clase minoritaria, no considera si esos casos son relevantes.
- No funciona bien con clases que están entremezcladas y puede generar ruido.
- Puede tener un alto coste computacional.
- Es sensible al número de **k** (vecinos cercanos).
- Solo funciona con variables continuas.

2.4. Implementación en Python

Vamos a continuar con la implementación del ejercicio que estábamos haciendo en el punto 1. El primer paso es aplicar la técnica SMOTE a los datos,y para ello, necesitamos instalar en nuestro equipo la librería ***imbalanced-learn***.

Una vez instalado, aplicamos SMOTE sobre el conjunto de datos y pasamos de ver los datos como estaban en la figura 5 a la figura 6. Podemos identificar en la figura 6, las líneas que se han creado uniendo los casos de la clase minoritaria.

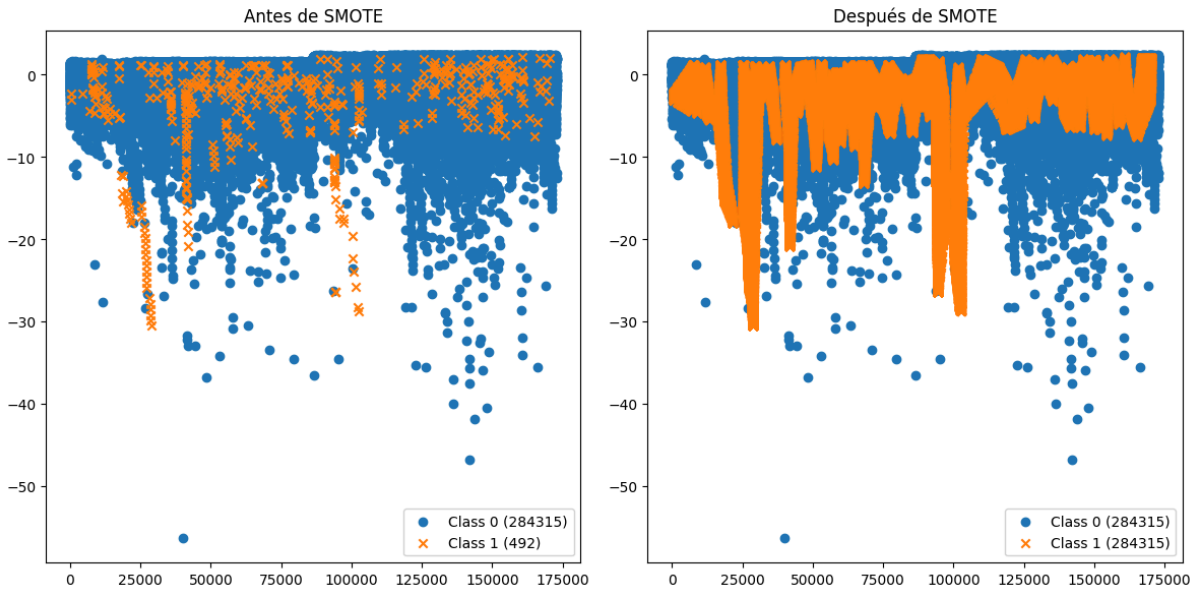

Figura 5: Gráfico de dispersión antes y después de aplicar SMOTE

Entrenando un modelo con estos datos obtendremos los resultados de la tabla 3 y la matriz de confusión de la figura 6.

Clase	Precisión	Recall	F1-score	Soporte
Normal	0.9697	0.9920	0.98084	85138
Fraude	0.9919	0.9692	0.9804	85451
			Accuracy	0.9806	170589

Tabla 3: Resultados tras aplicar SMOTE

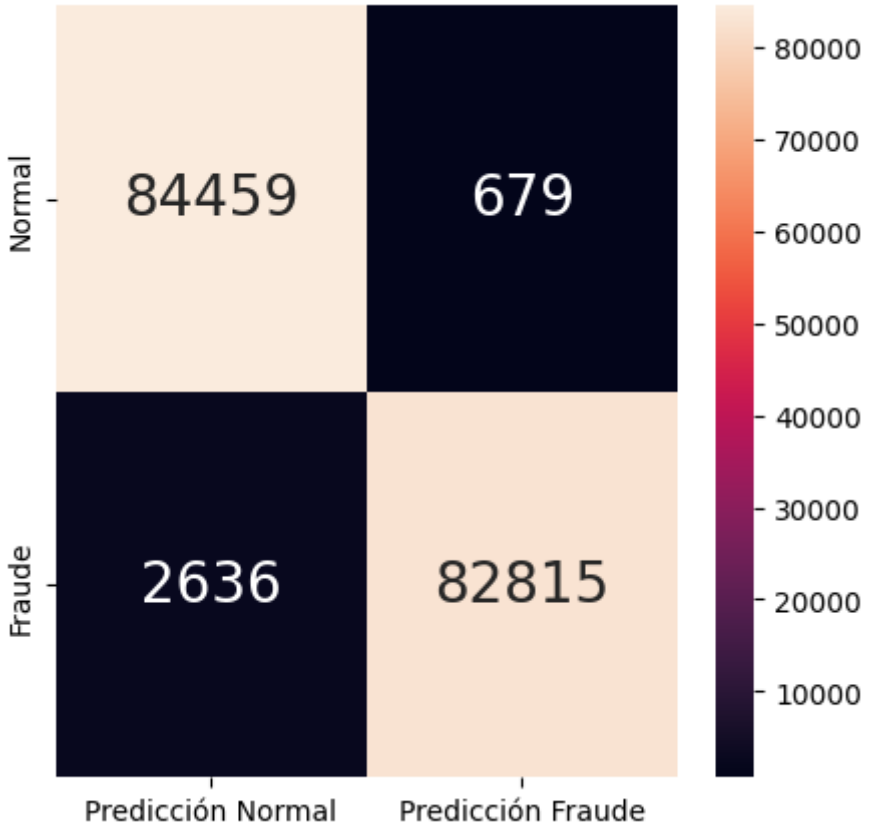

Figura 6: Matriz de confusión SMOTE.

Tenemos una precisión y un recall en el caso de la clase positiva bastante mejores que antes con rangos que superan el 99% y 97% respectivamente, lo que nos indica que esta vez sí identifica bien ambas clases. Y eso se demuestra poniendo atención al alto **accuracy** qué conseguimos.

3. ADASYN

ADASYN significa *Adaptative Synthetic Sampling*, y es una versión mejorada de SMOTE. Fue presentada en 2008 por **Haibo He, Yang Bai, Edwardo A. Garcia y Shutao Li [10].**

Esta técnica se ayuda de la distribución de densidad de la clase minoritaria a la hora de añadir nuevas instancias para aplicar ligeras variaciones en sus características, en vez de interpolar linealmente las instancias nuevas como hace SMOTE. Esto distribuye un poco más los datos y le da un toque más realista.

3.1. Funcionamiento

1. Selecciona la clase minoritaria.
2. Calcula el ratio de dificultad (DR) para un punto A, que representa el nivel de desbalanceamiento del grupo donde está ubicado el punto.

$$DR = \frac{Número\ de\ puntos\ de\ la\ clase\ mayoritaria}{Número\ de\ puntos\ de\ la\ clase\ minoritaria}$$

3. Generación de instancias aplicando DR, genera más donde el valor de este sea más alto ya que indica mayor desbalance en los datos.
4. Repetir desde el punto 1 hasta equilibrar ambas instancias.

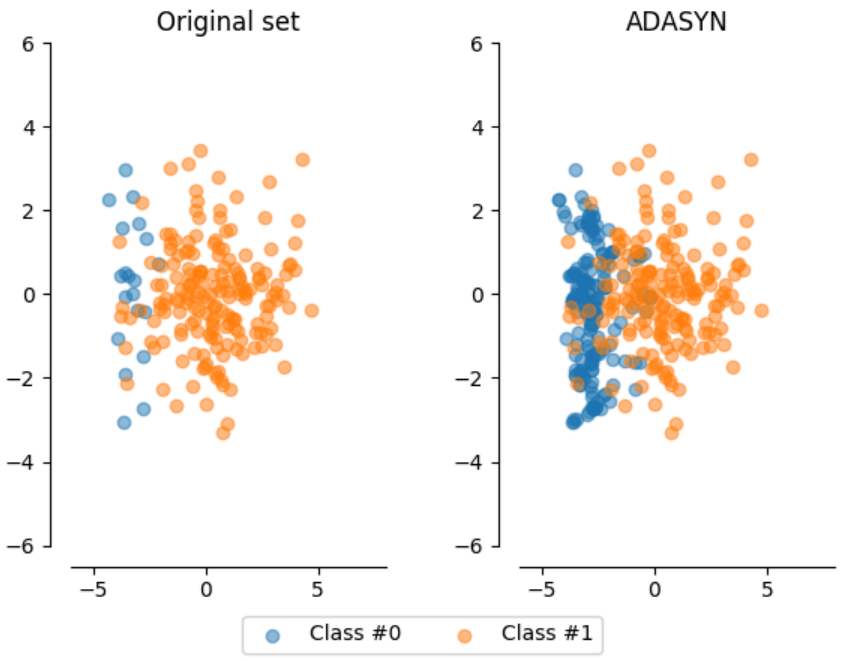

Figura 7: Ejemplo de uso de ADASYN
(https://www.researchgate.net/figure/Schematic-diagram-of-ADASYN-algorithm-application_fig5_344603283)

3.2. Beneficios

- Nos ayuda a balancear los datos.
- Reduce el sesgo sobre la clase mayoritaria.
- Genera instancias de la clase minoritaria más realistas.
- Aplicabilidad a problemas del mundo real (detección de intrusos, investigación médica o detección de fraudes)

3.3. Limitaciones

- Coste computacional alto.
- Peligro de sobreajuste.
- No tiene en cuenta los valores anómalos, pueden afectar a la calidad de los datos.

3.4. Implementación en Python

Siguiendo con el problema planteado, vamos usar la técnica ADASYN proporcionada también en la librería ***imbalanced-learn***.

En las siguiente figuras podemos ver cómo genera las nuevas instancias ADASYN (figura 8), y hacemos una comparación directa de la generación de instancias SMOTE y ADASYN (figura 9).

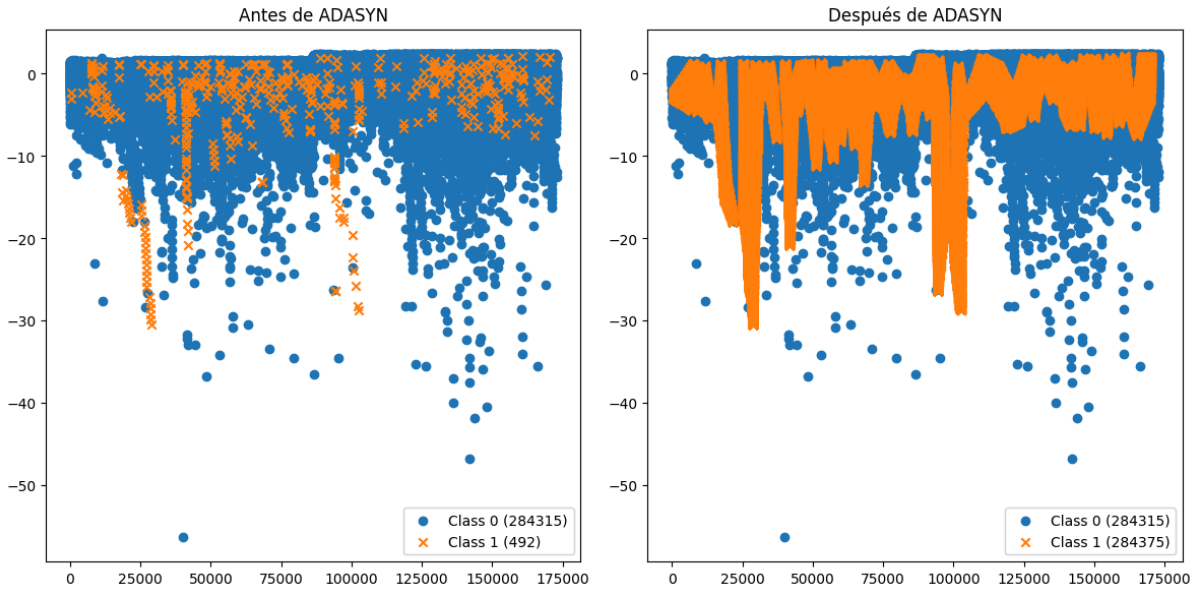

Figura 8: Gráfico de dispersión antes y después de aplicar ADASYN

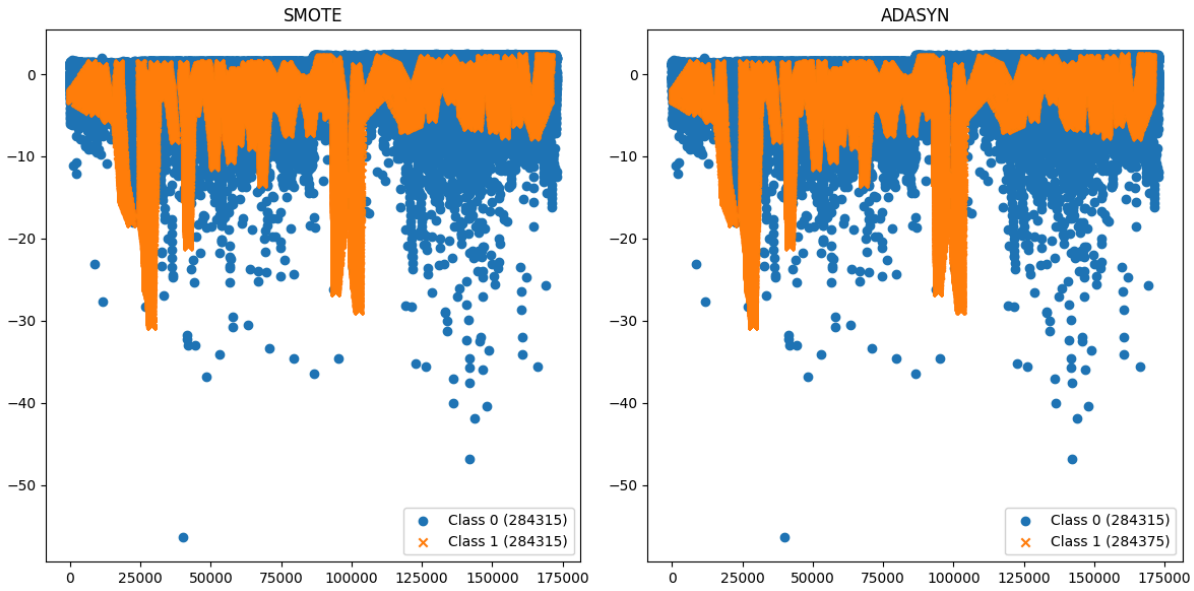

Figura 9. Comparación directa entre SMOTE y ADASYN.

En la figura anterior podemos ver la comparativa entre ambas técnicas. A simple vista no parecen iguales, pero si agudizamos la vista se pueden apreciar pequeños cambios.

Siguiendo con la implementación en python que estamos realizando, al entrenar el modelo obtenemos los resultados de la tabla 4 y figura 10.

Técnica	Clase	Precisión	Recall	F1-score	Soporte
SMOTE	Normal	0.9697	0.9920	0.98084	85138
	Fraude	0.9919	0.9692	0.9804	85451
			Accuracy	**0.9806**	170589
ADASYN	Normal	0.9685	0.9917	0.9799	85191
	Fraude	0.9915	0.9678	0.9795	85416
			Accuracy	**0.9797**	170607

Tabla 4. Comparación de resultados de SMOTE y ADASYN.

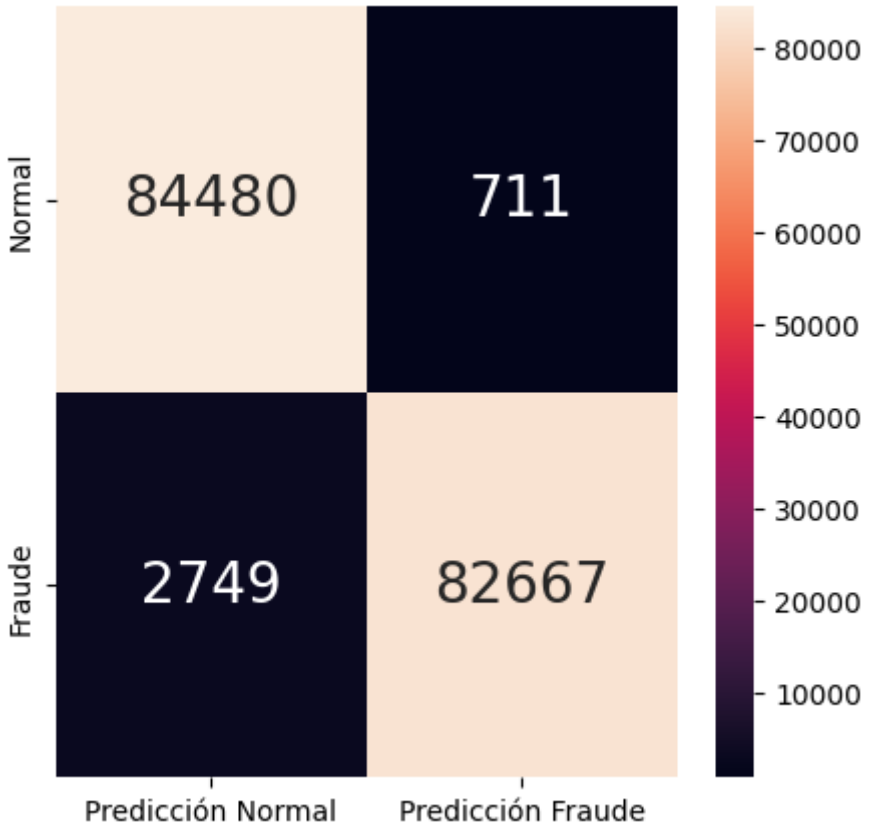

Figura 10. Matriz de confusión ADASYN.

Para nuestro caso práctico, viendo los resultados obtenidos en la tabla 4, la técnica SMOTE es mejor siendo ADASYN peor solo por **9 milésimas**. Esto nos indica que las dos técnicas son válidas para resolver nuestro problema de datos no balanceados. ADASYN al igual que SMOTE obtiene muy buenos datos de precisión y recall, es decir, identifica muy bien ambas clases, y obtiene un **accuracy** muy cercano al **98%**.

4. Conclusión

En este estudio hemos explorado dos técnicas de balanceo de datos, y para decantarnos por una u otra tenemos que tener en cuenta estos dos factores.

1. **La generación de instancias**: SMOTE interpola entre los conjuntos de la clase minoritaria y ADASYN tiene en cuenta el ratio de dificultad para generar instancias más realistas.
2. **El conjunto de datos**: Puede aumentar el coste de computación ante datos muy desbalanceados, o contener outliers con influyan en el resultado final

En resumen, sabiendo estos dos factores, ambas técnicas han demostrado su eficacia al identificar las clases en el ejemplo práctico y queda determinada su elección a las **preferencias del usuario**.

5. Bibliografía

[1]. “Clasificación con datos desbalanceados.” *Aprende Machine Learning*, https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/.

[2]. “Credit Card Fraud Detection.” *Kaggle*, https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud/data.

[3]. “Overcoming Class Imbalance with SMOTE: How to Tackle Imbalanced Datasets in Machine Learning.” *Train in Data Blog*, https://www.blog.trainindata.com/overcoming-class-imbalance-with-smote/.

[4]. “SMOTE and ADASYN (Handling Imbalanced Data Set) | by Indresh Bhattacharyya | Coinmonks.” *Medium*, https://medium.com/coinmonks/smote-and-adasyn-handling-imbalanced-data-set-34f5223e167.

[5]. “Adaptive Synthetic Sampling (ADASYN) | Adaptive Synthetic Sampling (ADASYN) Definition, Adaptive Synthetic Sampling (ADASYN) Use in ML.” *Activeloop*, https://www.activeloop.ai/resources/glossary/adaptive-synthetic-sampling-adasyn/.

[6]. “Data Imbalance: How is ADASYN different from SMOTE?” *Medium*, https://medium.com/@penpencil.blr/data-imbalance-how-is-adasyn-different-from-smote-f4eba54867ab.

[7]. “SMOTE — Version 0.11.0.” *Imbalanced-Learn*, https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html.

[8]. “ADASYN — Version 0.11.0.” *Imbalanced-Learn*, https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html.

[9]. “SMOTE: Synthetic Minority Over-sampling Technique.” *Journal of Artificial Intelligence Research*, https://www.jair.org/index.php/jair/article/view/10302.

[10]. “ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning.” *Soft Computing and Intelligent Information Systems*, https://sci2s.ugr.es/keel/pdf/algorithm/congreso/2008-He-ieee.pdf.