Tecnológico de Monterrey

Desarrollo de aplicaciones avanzadas de ciencias computacionales

“Resultados obtenidos - Bank Analysis"

Presenta

Denisse Domínguez Bolaños | A01702603

Fecha:

29 de abril del 2025

El desbalanceo de datos es un problema común en el desarrollo de modelos de machine learning, donde algunas clases tienen significativamente menos ejemplos que otras. Este desequilibrio puede generar sesgos haciendo que el modelo aprenda de las clases mayoritarias y tenga un rendimiento deficiente al identificar las minoritarias.

El principal problema detectado en el modelo de BankSell fue el desbalanceo de clases, esto fue detectado al analizar la matriz de confusión, donde en ella fue evidente que el modelo tendía a predecir la clase dominante (No), dejando a un lado la clase minoritaria. A pesar que el modelo mostraba una exactitud superior al 80%, esta métrica en muchas ocasiones es engañosa justamente porque el modelo en ocasiones aprende de la clase dominante, por eso, es importante completar el análisis con métricas y herramientas que validen los resultados.

Debido a lo anterior, se aplicaron técnicas de balanceo de datos ya que este era el principal problema en el modelo, para ello, se recurrió a diferentes artículos académicos para buscar la mejor implementación.

Implementación 1

Basada en el paper ***"A hybrid machine learning model for intrusion detection in wireless sensor networks leveraging data balancing and dimensionality reduction" [1]***

En este paper se proponen distintas soluciones para mejorar modelos de clasificación (como el de BankSell) ante conjuntos de datos desbalanceados; se usan técnicas de balanceo de datos como:

- SMOTE TomekLInk (STL)
- Generative Adversarial Network (GAN)
- KMeans-SMOTE (KMS)

Los resultados demuestran que la mejor combinación es KMS + PCA + Random Forest, estos en conjunto demuestran haber mejorado el rendimiento, destacando en métricas como exactitud, precisión, recuperación y f1.

Se decidió implementar KMS, dado que fue la técnica que mostró mejores resultados para el balanceo de datos, sin embargo, no nos dio los resultados esperados, y era esperado ya que se estaba haciendo un balanceo demasiado extremo e incoherente.

Resultados implementación 1

Antes de balancear las clases, teníamos lo siguiente:

- Instancias de No: 17,693
- Instancias de Yes: 3,984

Tras aplicar KMS, se logró igual cantidad de instancias para las clases en el conjunto de entrenamiento. (*train_data)*

- Instancias de No: 17,693
- Instancias de Yes: 17,694

En la *Imagen 1* se presentan los resultados impresos directamente desde la consola del código, los cuales confirman que se logró el balance de clases en el conjunto de entrenamiento

Imagen 1

```
print(f"The number of classes before fit: {Counter(train_labels_target)}")
kms = KMeansSMOTE(random_state=42)
train_data, train_labels_target = kms.fit_resample(train_data, train_labels_target)
print(f"The number of classes after fit: {Counter(train_labels_target)}")
```

```
The number of classes before fit: Counter({np.int64(0): 17693, np.int64(1): 3984})
The number of classes after fit: Counter({np.int64(1): 17694, np.int64(0): 17693})
```

Una vez completado el balanceo, se procedió a reentrenar el modelo y evaluar su desempeño utilizando las métricas sugeridas en el paper

- Accuracy
- Precision
- Recall
- F1

Los resultados obtenidos se muestran en la *Imagen 2*, estos son impresos directamente desde la consola.

Imagen 2

	precision	recall	f1-score	support
no	0.99	0.94	0.97	20624
yes	0.42	0.80	0.55	1053
accuracy			0.94	21677

Recordemos que antes del balanceo, el modelo presentaba un 82% y 88%, con esta técnica, el accuracy subió a 0.94%

De estos resultados interpretamos que:

- Se logró un equilibrio de instancias entre no y yes. (KMS)
- El modelo ahora no ignora la clase débil (yes)
- Efectivamente el modelo mejoró su capacidad para detectar la clase minoritaria (yes), pero contamos con muchos falsos positivos.
- De todo lo que el modelo nos dijo que era “yes” solo el 42% era correcto.
- Con recall notamos que el modelo fue capaz de detectar el 80% de los casos reales para yes, que si lo comparamos a lo que teníamos antes, siendo la clase débil, fue un avance alto.

- El modelo definitivamente necesita afinarse, porque con F1 y recall para la clase de yes, nos damos cuenta que si, el modelo ya captura más casos de "yes", pero necesita reducir sus falsos positivos porque están afectando los resultados, ya que siguen siendo mayoría la clase de no, por mucho.

Sin embargo, el resultado no fue el esperado, porque fue demasiado arriesgado y atrevido generar una cantidad tan grande de datos sintéticos a partir de solo 3,984 ejemplos reales. En el entrenamiento del modelo, esto significó crear una gran cantidad de instancias nuevas a partir de una muestra muy limitada, lo que redujo la calidad y representatividad de los datos generados.

Implementación 2

Basada en el paper *"A deep learning approach for classifying and predicting children's nutritional status in Ethiopia using LSTM-FC neural networks" [2]*

Puesto que la implementación anterior no produjo el balanceo ni los resultados esperados con los datos generados mediante KMS, se tomó la decisión de implementar SMOTE. Aunque KMS es una versión mejorada de SMOTE, en ciertos casos es bueno tomar un paso atrás y recurrir a los métodos originales, esta decisión se basa en que un enfoque más simple puede dar mejores resultados.

A pesar de ser una técnica más básica, tampoco proporcionó los resultados esperados. La situación que se observó fue muy similar o ligeramente diferente a la obtenida con KMS. Fue en este punto cuando se analizó que la generación de datos sintéticos no estaba siendo la estrategia adecuada para nuestro caso. Por ello, se optó por cambiar el enfoque y recurrir a otra técnica e implementación.

Implementación 3

Basada en el paper *"A deep learning approach for classifying and predicting children's nutritional status in Ethiopia using LSTM-FC neural networks" y "A stacked ensemble approach with resampling techniques for highly effective fraud detection in imbalanced datasets" [3]*

Al ver que la generación de datos sintéticos no estaba siendo la mejor opción, optamos por hacer undersampled y oversample, pero antes de ello, también incrementamos la cantidad de datos que teníamos en nuestro set de entrenamiento (*train_data*), creyendo que esto nos ayudaría no solo a tener más datos para entrenar, sino que también mayor cantidad de "yes", pero tristemente, esto no sucedió, si se incrementó la cantidad de datos para el entrenamiento, pero cantidad de datos para "yes" permaneció igual. Ahora contamos con:

- 43,354 datos para entrenar.
- De ellos, más de 30,000 corresponden a "No"

Se decidió usar método SMOTE para el oversample y para el undersampled se usó RandomUnderSampler, la idea de esto fue que de la clase yes, se tuvieran en total 15,000 datos y para la clase "no" tener únicamente 12,000 datos, reconocemos que estamos perdiendo datos, pero en este punto, creemos que es un buen escenario a probar

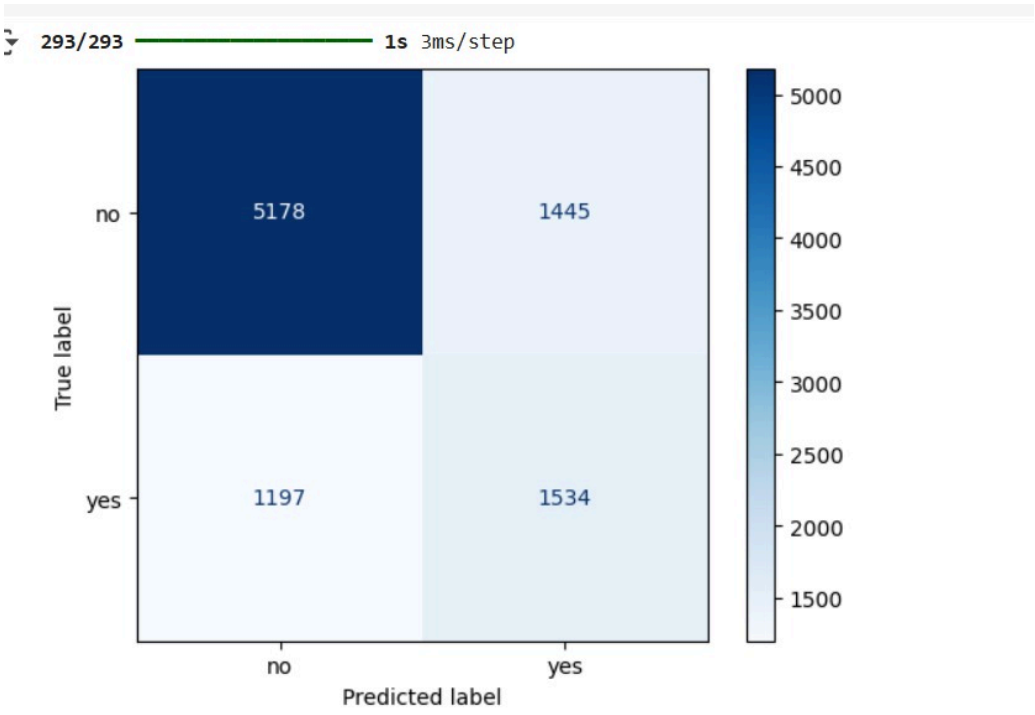
Si bien, reconocemos que estamos perdiendo datos al reducir la cantidad de datos originales, consideramos que en este punto representa un escenario razonable a la situación y a lo previamente aprendido con las implementaciones anteriores, siendo un escenario razonable y prometedor.

Como se muestra en la *Imagen 3,* se logró el objetivo de ajustar el número de instancias para cada clase, pero los resultados seguían sin ser satisfactorios *Imagen 4*. El modelo aún no estaba generando predicciones reales ni alcanzando el desempeño esperado.

Imagen 3

```
The number of classes after fit: Counter({np.int64(1): 15000, np.int64(0): 12000})
```

Imagen 4



Pero aparte de cambiar la implementación para el balanceo de datos, agregamos las mejoras del modelo.

Mejora del modelo

Con ayuda y mejor uso de hiperparametros, quienes son una herramienta para mejorar el desempeño del modelo, hemos hecho uso de algunos de ellos. No solo buscamos hacer un modelo más eficiente, sino que también nos enfocamos en nuestro principal desafío, el desbalanceo de datos. Para ello, seleccionamos hiperparametros que nos ayuden a mitigar ambos casos, de igual manera, referenciando a los artículos académicos consultados.

- Learning Rate: Nos ayuda a controlar la velocidad en la que los pesos cambian.
    - Anteriormente, solo usábamos el que "Adam" usa por default, pero lo hemos alentado más, cambiando a 0.0005, en "*A stacked ensemble approach with resampling techniques for highly effective fraud detection in imbalanced datasets"* [3] usaron un learning rate de 0.01, pero hemos decidido irnos por otro camino.
- Batch Size: Cantidad de muestras que toma por época.
    - Antes no lo teníamos definido y nos hemos ido por un número ni alto, ni bajo, para evitar ruido, lentitud ni prisa por entrenar al modelo, se usó un batch size de 64.
- Class weight: Evita que el modelo ignore la clase minoritaria, decirle al modelo, cuando le va doler equivocarse, si bien, "Yes" tiene más datos, es probable que el modelo, siga prefiriendo a "No" porque es más fácil de aprender de él.
- Threshold: Que tanto le permitimos al modelo tomar una decisión, es decir, controlar que tan convencido debe estar para tomar una decisión.
    - Anteriormente lo teníamos en 0.5, lo hemos bajado a 0.3 para permitir que cuando no esté tan convencido de una respuesta "Yes" aún pueda considerarla como tal
- Dense: Capas y profundidad.
    - En un inicio, solo teníamos un capa básica, después la aumentamos a dos, pero para la mejora final, nos hemos quedado con cuatro capas y cada una cuenta con diferente profundidad.
- En el artículo [3] se usan dropouts para evitar overfitting, obligando al modelo a no depender de ciertos nodos, ayudándonos para nuestra clase "débil" (Yes)
    - Anteriormente no se usaban, ahora los incorporamos con un intervalo de 0.2 - 0.5

Resultados

En la *Imágen 5* se encuentran los resultados de acuerdo a las métricas recomendadas por el artículo [1].

Logramos el objetivo de un balanceo más adecuado y realista de datos, ahora el modelo no tiene tanta preferencia por la clase "No" ya que la precisión de "Yes" subió de 0.42 a 0.50, lo que indica que ya no se cometen tantos falsos positivos.

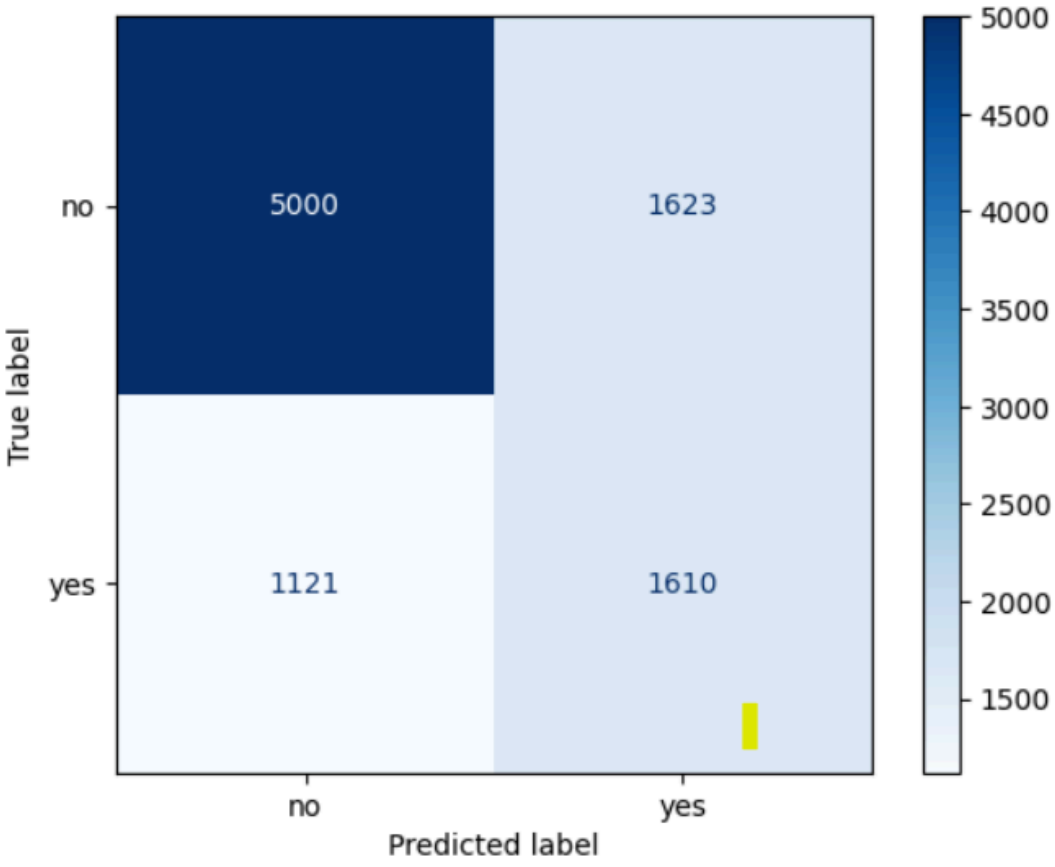
A consecuencia de esto y que ahora el modelo ya no está sesgado con los casos de "No" y toma el riesgo de tomar decisiones estando no tan convencido, tenemos las siguientes consecuencias

- El recall de "Yes" ha bajado drásticamente de 0.80 a 0.59, esto nos dice, que se nos están yendo los casos positivos, el modelo no los está detectando.
- Recall y precision de "No" también bajaron, le damos causa de eso al riesgo que está tomando el modelo en su toma de decisiones.
- El acurracy también bajo impresionantemente de 0.94 a 0.71, pero esto no necesariamente es malo, porque recordemos que antes teníamos un modelo que estaba fuertemente inclinado a los casos de "No", ahora, al tener estrategias que eviten el sesgo, tenemos una predicción más equilibrada, por ello, es esperado que el acurracy sea menor.

A comparación de los resultados del primer modelo, sin las mejoras, podemos decir que las métricas bajaron considerablemente, pero recordemos que estábamos trabajando con un modelo sesgado y fuertemente inclinado por la clase “No”, ahora, gracias a las mejoras implementadas, estamos evitando este caso al igual que se agregaron mejoras que hacen un modelo más eficiente, correcto y realista.

Imágen 5

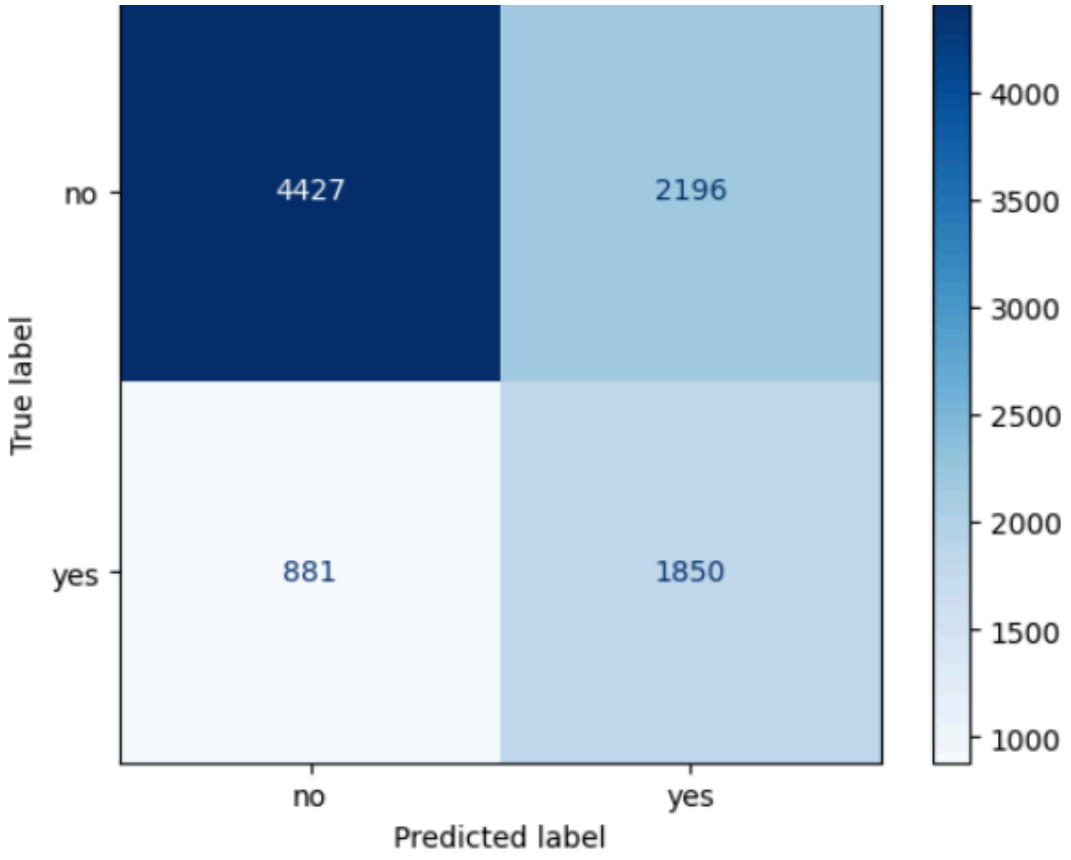
	precision	recall	f1-score	support
no	0.82	0.75	0.78	6623
yes	0.50	0.59	0.54	2731
accuracy			0.71	9354
macro avg	0.66	0.67	0.66	9354
weighted avg	0.72	0.71	0.71	9354



Después, se volvió a correr el código para ver las diferencias, en la *Imágen 6* podemos notarlas, si bien, son diferentes a la anterior, pero es por la creación y reducción de datos.

Imágen 6

	precision	recall	f1-score	support
no	0.83	0.67	0.74	6623
yes	0.46	0.68	0.55	2731
accuracy			0.67	9354
macro avg	0.65	0.67	0.64	9354
weighted avg	0.72	0.67	0.68	9354



Referencias

[1] Talukder, M.A., Khalid, M. & Sultana, N. A hybrid machine learning model for intrusion detection in wireless sensor networks leveraging data balancing and dimensionality reduction. Sci Rep 15, 4617 (2025). https://doi.org/10.1038/s41598-025-87028-1

[2] Begashaw, G.B., Zewotir, T. & Fenta, H.M. A deep learning approach for classifying and predicting children's nutritional status in Ethiopia using LSTM-FC neural networks. BioData Mining 18, 11 (2025). https://doi.org/10.1186/s13040-025-00425-0

[3] A stacked ensemble approach with resampling techniques for highly effective fraud detection in imbalanced datasets. (2025). Journal of the Nigerian Society of Physical Sciences, 7(1), 2066. https://doi.org/10.46481/jnsps.2025.2066