DETECCIÓN DE FRAUDE EN OPERACIONES CON TARJETA DE CRÉDITO EN EUROPA

CoderHouse – Data Science

Belen Loiácono

Abril 2024

Contents

Descripción del caso de negocio

En un contexto donde el uso de tarjetas de crédito es una práctica común en Europa, la detección de fraudes se convierte en un desafío crucial para las instituciones financieras y los proveedores de servicios de pago.

La creciente sofisticación de las técnicas de fraude, combinada con el vasto volumen de transacciones diarias, hace que la identificación temprana y precisa de actividades fraudulentas sea imperativa para salvaguardar la integridad del sistema financiero y proteger a los usuarios.

En este trabajo final del curso de Data Science para Coder House, exploraremos en detalle las diversas metodologías, técnicas y algoritmos empleados para detectar y prevenir fraudes en tarjetas de crédito en Europa. Analizaremos una base de datos, aplicaremos modelos predictivos y de aprendizaje automático, y evaluaremos la eficacia de nuestras soluciones propuestas en la detección y mitigación del fraude.

Nuestro objetivo es contribuir al desarrollo de estrategias y herramientas más efectivas que permitan a las entidades financieras y a los reguladores enfrentar los desafíos cada vez más sofisticados y cambiantes en el panorama del fraude en tarjetas de crédito.

Tabla de versionado

Mantener un registro de las versiones del modelo predictivo a lo largo del tiempo es fundamental para garantizar la transparencia, reproducibilidad y trazabilidad en el desarrollo y despliegue de modelos. Esta permite un seguimiento detallado de los cambios en el modelo y facilita la comparación entre diferentes versiones del modelo y la identificación de mejoras o posibles problemas.

Fecha	**Versión**	**Autor/a**	**Comentarios**
Abril 2024	V1	Belén Loiácono	-

Objetivos del modelo

El objetivo principal de nuestro modelo es desarrollar un sistema robusto y preciso que sea capaz de identificar y prevenir transacciones fraudulentas de manera efectiva. Para lograr esto, emplearemos técnicas avanzadas de análisis de datos y machine learning para detectar patrones anómalos y comportamientos sospechosos en los datos de transacciones. Nuestro modelo aspira a proporcionar a las instituciones financieras una herramienta confiable que les permita mitigar riesgos, proteger los activos de los usuarios y mantener la integridad del sistema financiero en un entorno cada vez más desafiante y dinámico.

Principales preguntas a responder

1. ¿Cómo detectamos las operaciones fraudulentas?
2. ¿Con qué variables de nuestro Data Set se encuentran relacionadas?

Descripción de los datos

El conjunto de datos contiene transacciones realizadas con tarjetas de crédito en septiembre de 2013 por titulares europeos. Este conjunto de datos presenta transacciones que ocurrieron en dos días, donde tenemos 492 fraudes de un total de 219,129 transacciones. El conjunto de datos está altamente desbalanceado, la clase positiva (fraudes) representa el 0.172% de todas las transacciones.

Contiene sólo variables de entrada numéricas que son el resultado de una transformación PCA. La transformación PCA se utilizó para reducir el conjunto de datos original y eliminar atributos redundantes. Las características V1, V2, ..., V28 son los componentes principales obtenidos con la transformación PCA, las únicas características que no se han transformado con PCA son *'Time'* y *'Amount'*.

'Time' contiene los segundos transcurridos entre cada transacción y la primera transacción en el conjunto de datos.

'Amount' es el monto de la transacción. Esta característica puede ser utilizada para aprendizaje sensible al costo dependiente del ejemplo.

'Class' es la variable de respuesta y toma el valor 1 en caso de fraude y 0 en caso contrario.

Tabla resumen de las variables y sus principales características

Column	Type	Non-Null	Nulls
Id	int64	219,129	-
Time	float64	219,129	-
V1	float64	219,129	-
V2	float64	219,129	-
V3	float64	219,129	-
V4	float64	219,129	-
V5	float64	219,129	-
V6	float64	219,129	-
V7	float64	219,129	-
V8	float64	219,129	-
V9	float64	219,129	-
V10	float64	219,129	-
V11	float64	219,129	-
V12	float64	219,129	-
V13	float64	219,129	-
V14	float64	219,129	-
V15	float64	219,129	-
V16	float64	219,129	-
V17	float64	219,129	-
V18	float64	219,129	-
V19	float64	219,129	-
V20	float64	219,129	-
V21	float64	219,129	-
V22	float64	219,129	-
V23	float64	219,129	-
V24	float64	219,129	-
V25	float64	219,129	-
V26	float64	219,129	-
V27	float64	219,129	-
V28	float64	219,129	-
Amount	float64	219,129	-
Class	int64	219,129	-

Podemos ver que todas las variables son del tipo float64, indicando que son valores numéricos y que no encontraremos ningún valor que no sea numérico. Las únicas dos variables con un tipo diferente son "Class" y "id", lo cual tiene sentido dado que estos son números enteros, no requieren 64 bits de memoria.

Con este análisis también vemos que no contamos con ningún valor nulo. Vale aclarar, que un valor que sea 0 no es un valor nulo. Un valor nulo hace referencia a una celda que se encuentre vacía.

Considerando que la columna "Id" sólo nos brinda información respecto al número de orden de los registros, podemos desprendernos dado que contamos con un índice que provee la misma información.

Del total de las 31 variables restantes no retiraremos ninguna de antemano especialmente considerando que desconocemos lo que cada una de las V representa.

EDA: Exploratory Data Analysis

El Exploratory Data Analysis es una fase fundamental en el proceso de análisis de datos, que tiene como objetivo comprender la naturaleza y características del conjunto de datos. Durante esta etapa, se realizan una serie de técnicas y visualizaciones para explorar las distribuciones, relaciones y patrones presentes en los datos. El EDA permite identificar posibles problemas de calidad de datos, detectar valores atípicos, comprender la estructura subyacente de los datos y generar hipótesis iniciales sobre las relaciones entre las variables.

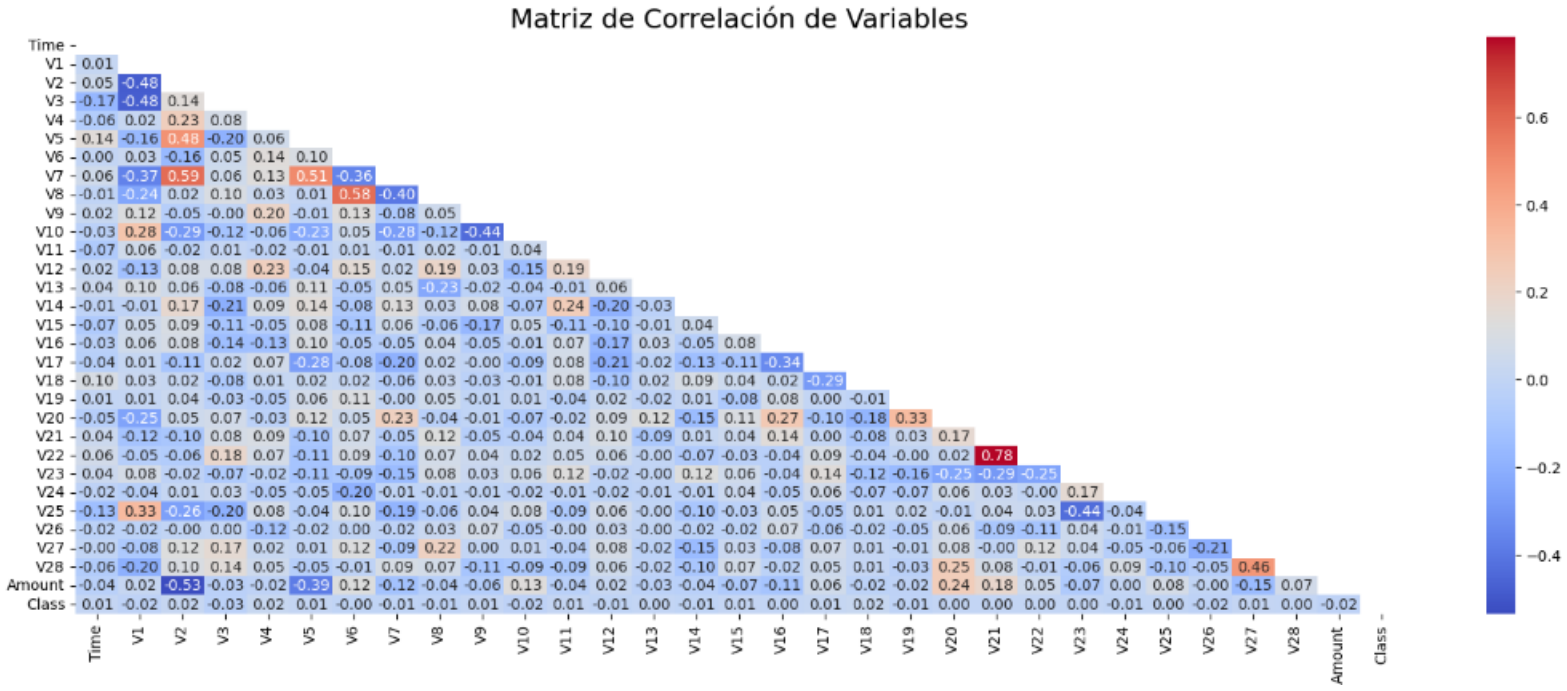
A continuación, realizaremos un análisis detallado de las variables presentes en nuestro conjunto de datos. Examinaremos las características numéricas y categóricas para comprender su distribución, tendencias y posibles correlaciones. Además, exploraremos la relación entre las variables y la variable de respuesta 'Class', que indica si una transacción es fraudulenta o no. Utilizaremos técnicas de visualización como histogramas, gráficos de dispersión y matrices de correlación para obtener una comprensión completa de los datos y prepararnos para el siguiente paso en nuestro análisis predictivo.

Correlación entre variables independientes

La correlación entre variables es una medida que indica cómo cambian juntas dos variables. El análisis de correlación es útil para comprender las relaciones entre las variables en un conjunto de datos y puede ayudar en la selección de características y en la identificación de posibles patrones. Sin embargo, la correlación no implica causalidad, es decir, no indica necesariamente que un cambio en una variable cause un cambio en la otra.

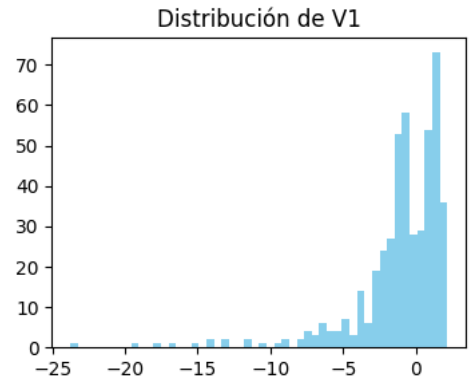
Para analizar la correlación entre nuestras variables independientes, se utilizó una *'Matriz de Correlación'* que permite ver las relaciones, 1 a 1, entre todas las variables del DataSet. Asumimos que si el valor absoluto de la correlación entre dos variables es superior a 0.6 existe una correlación y, si es superior a 0.8, la correlación es fuerte lo que implicaría tener que eliminar alguna de las dos variables.

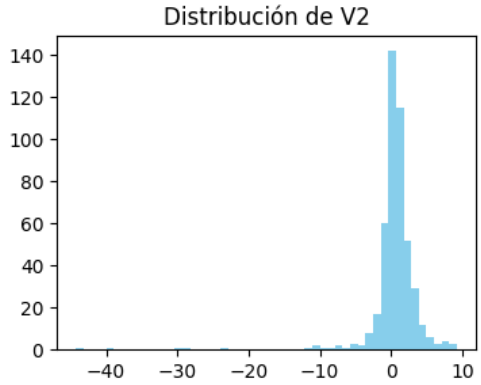
Para evaluar la correlación se utilizó el método Spearman dado que este método es útil cuando los datos no siguen una distribución normal o cuando la relación entre las variables no es lineal. Es especialmente útil en situaciones donde la relación entre las variables puede ser más compleja o no lineal.

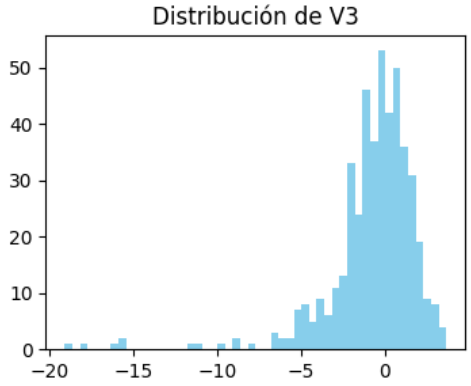


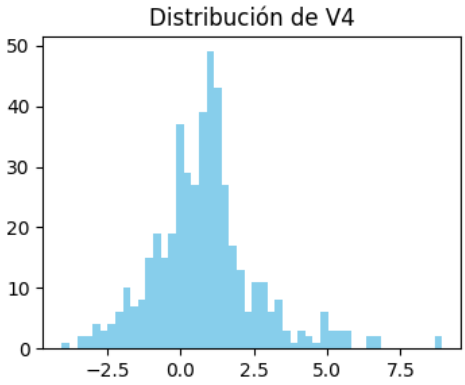
La correlación entre variables es baja, a excepción de las variables V21 y V22 que presentan un grado de 0.78. En caso de, al continuar el análisis, tener que reducir la cantidad de variables a utilizar, se podrá descartar una de estas dos por grado alto de multicolinealidad con la otra.

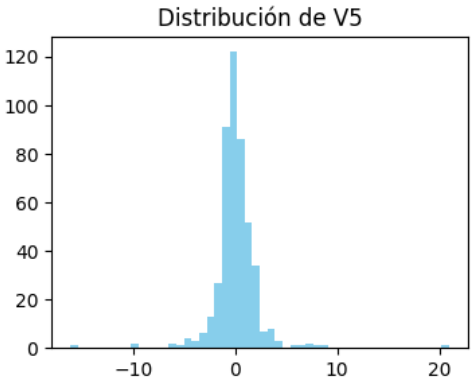
A continuación, podemos ver las distribuciones de las variables, limitándonos a aquellos registros para los cuales *Class = 1* (es decir, aquellos registros que son operaciones fraudulentas). Como vemos, ninguna de las variables responde a una distribución normal (algunas se acercan más que otras) y todas cuentan con valores extraordinarios (outliers), por lo que es recomendable utilizar el método de Spearman al de Pearson, como fue previamente aclarado.

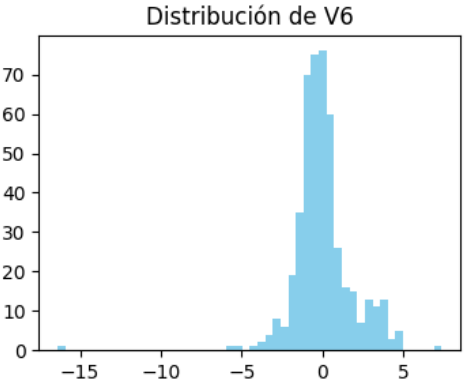

Distribución de V1
70
60
50
40
30
20
10
0
−25
−20
−15
−10
−5
0

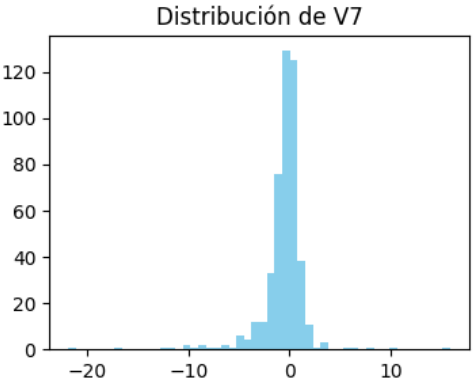

Distribución de V2
140
120
100
80
60
40
20
0
−40
−30
−20
−10
0
10

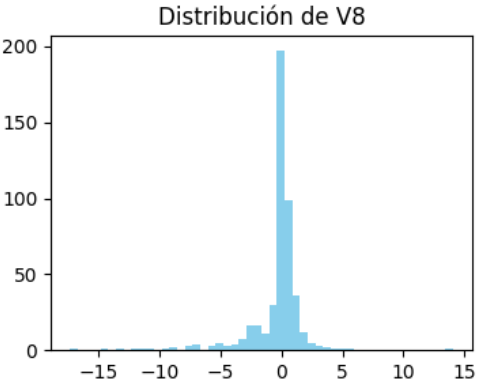

Distribución de V3
50
40
30
20
10
0
−20
−15
−10
−5
0

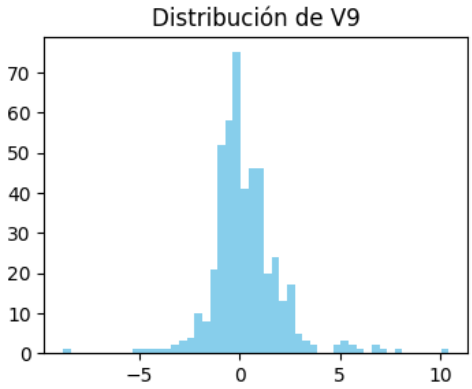

Distribución de V4
50
40
30
20
10
0
−2.5
0.0
2.5
5.0
7.5

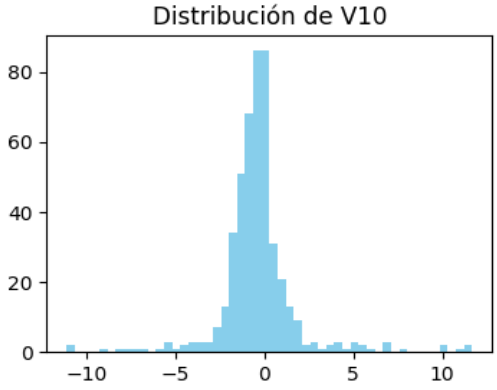

Distribución de V5
120
100
80
60
40
20
0
−10
0
10
20

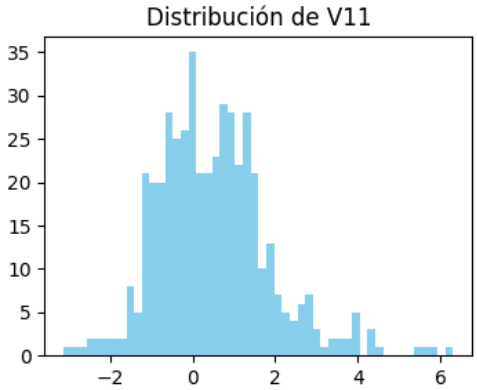

Distribución de V6
70
60
50
40
30
20
10
0
−15
−10
−5
0
5

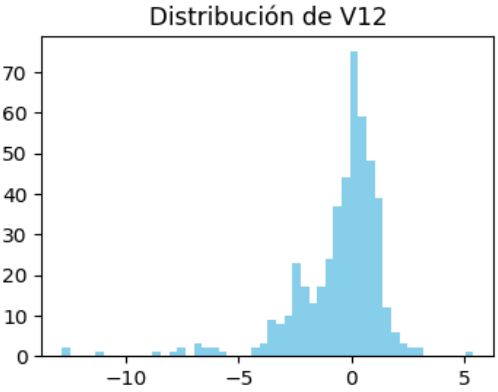

Distribución de V7
120
100
80
60
40
20
0
−20
−10
0
10

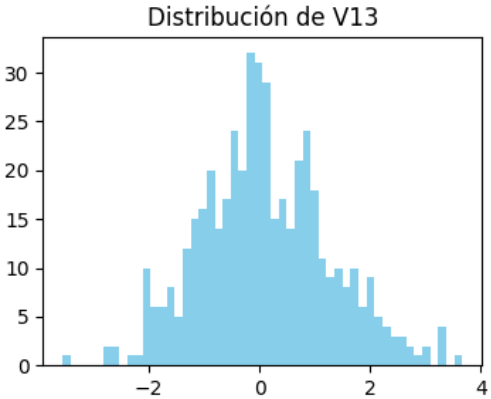

Distribución de V8
200
150
100
50
0
−15
−10
−5
0
5
10
15

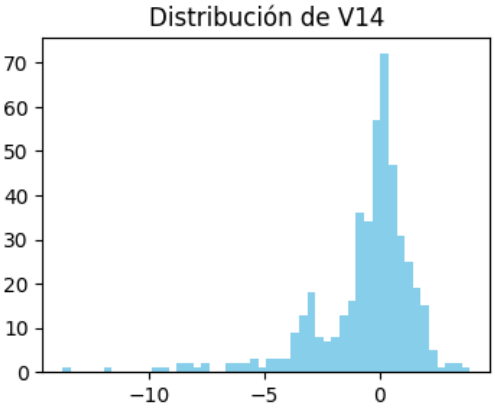

Distribución de V9

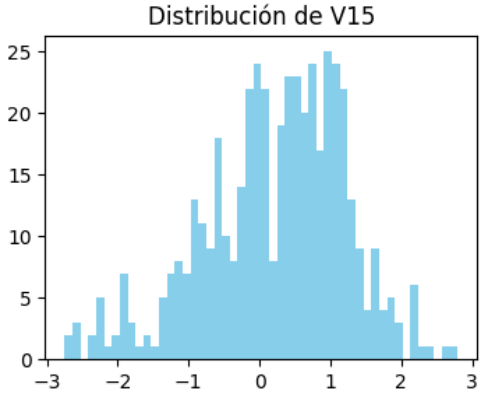

Distribución de V10

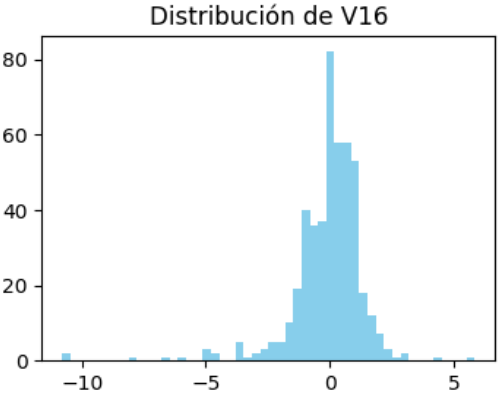

Distribución de V11

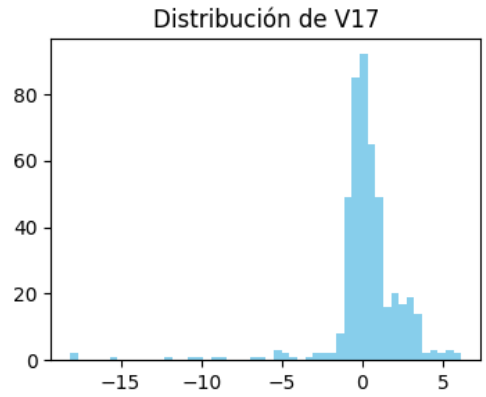

Distribución de V12

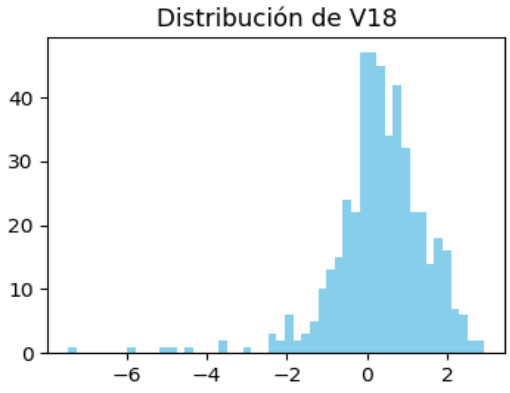

Distribución de V13

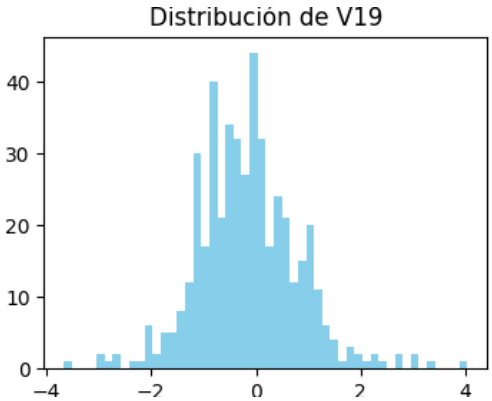

Distribución de V14

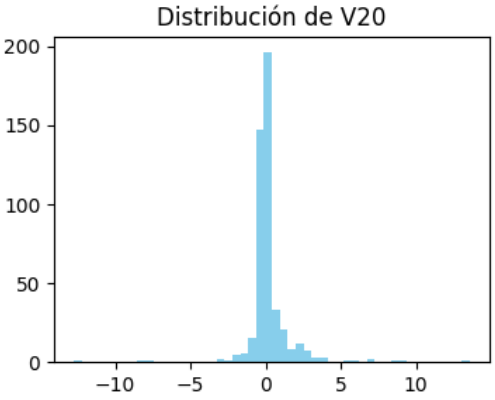

Distribución de V15

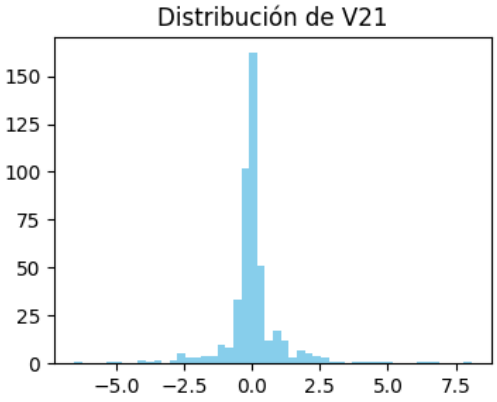

Distribución de V16

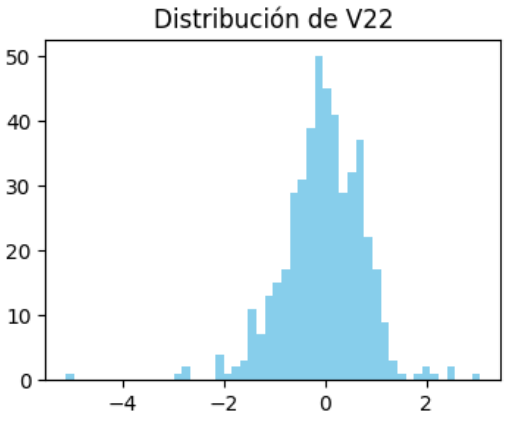

Distribución de V17
80
60
40
20
0
−15
−10
−5
0
5

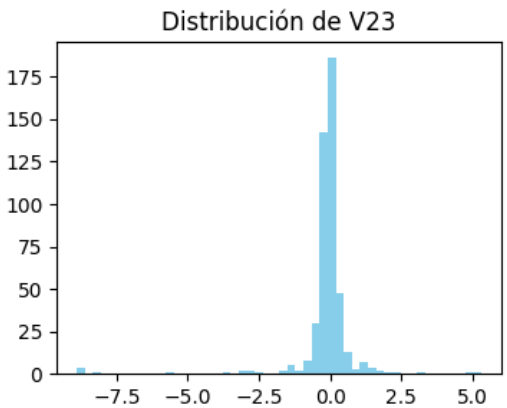

Distribución de V18
40
30
20
10
0
−6
−4
−2
0
2

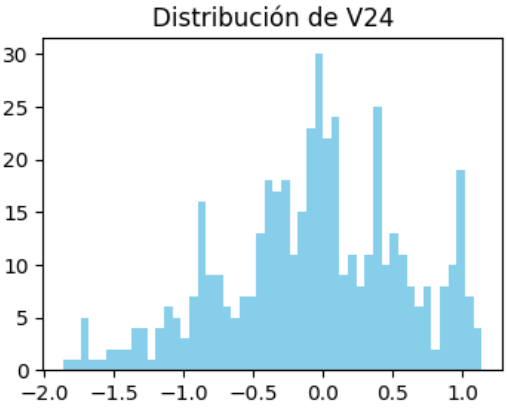

Distribución de V19
40
30
20
10
0
−4
−2
0
2
4

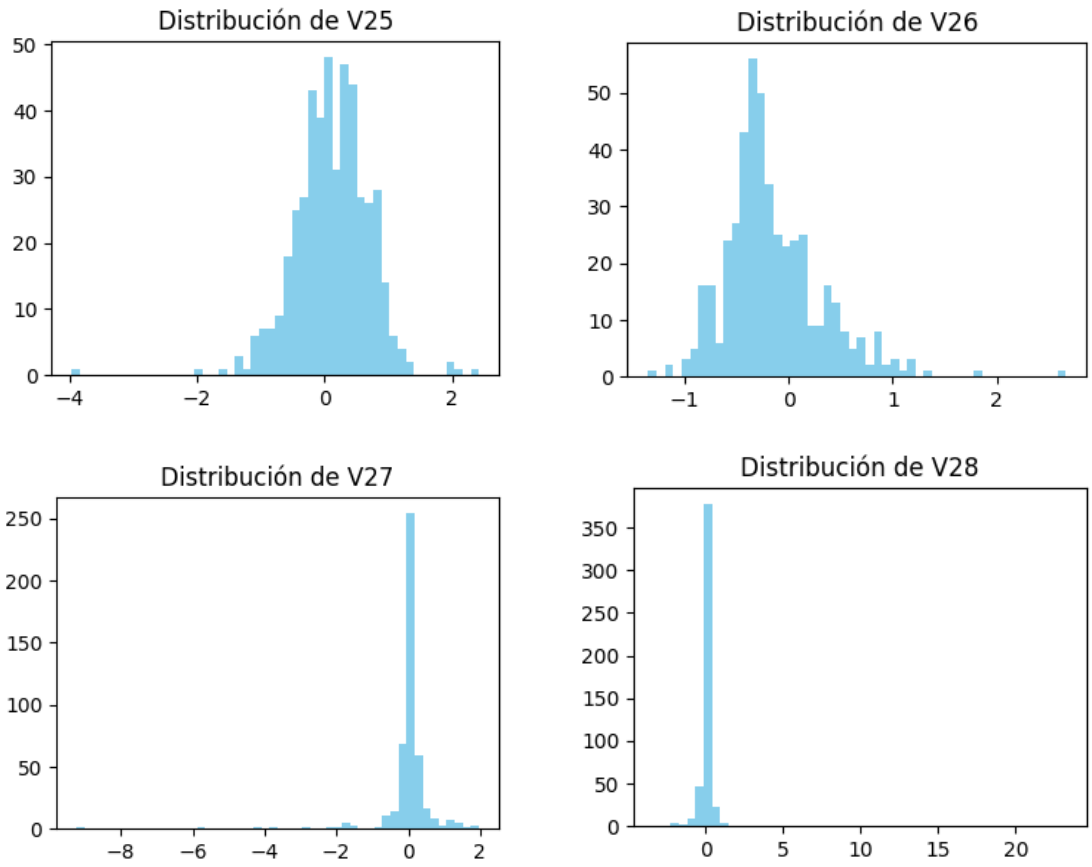

Distribución de V20
200
150
100
50
0
−10
−5
0
5
10


Distribución de V21
150
125
100
75
50
25
0
−5.0
−2.5
0.0
2.5
5.0
7.5


Distribución de V22
50
40
30
20
10
0
−4
−2
0
2


Distribución de V23
175
150
125
100
75
50
25
0
−7.5
−5.0
−2.5
0.0
2.5
5.0


Distribución de V24
30
25
20
15
10
5
0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0



Algoritmo elegido

Se utilizaron tres algoritmos diferentes en búsqueda del óptimo.

1. **Árbol de decisión**

Los árboles de decisión son modelos de aprendizaje automático que utilizan una estructura de árbol para tomar decisiones basadas en las características de entrada. En cada nodo del árbol, se realiza una pregunta sobre una característica específica, dividiendo así el conjunto de datos en subconjuntos más pequeños. Este proceso se repite recursivamente en cada subconjunto hasta alcanzar un criterio de parada, como un límite en la profundidad del árbol o la pureza de los nodos. Los árboles de decisión son fáciles de interpretar y pueden manejar datos numéricos y categóricos.

2. **Random Forest**

Random Forest es un algoritmo de aprendizaje automático basado en la técnica de "ensamble". Consiste en la construcción de múltiples árboles de decisión durante el entrenamiento y la combinación de sus predicciones para obtener una predicción más robusta y precisa. Cada árbol de decisión en el bosque se entrena con una muestra aleatoria del conjunto de datos y una selección aleatoria de características. Al combinar múltiples árboles entrenados de esta manera, Random Forest reduce el sobreajuste y mejora la generalización del modelo.

3. **XGBoost**

XGBoost (Extreme Gradient Boosting) es una implementación optimizada y altamente eficiente de la técnica de "boosting" de árboles de decisión. Al igual que Random Forest, XGBoost crea un conjunto de árboles de decisión durante el entrenamiento. Sin embargo, *en lugar de entrenar cada árbol de forma independiente, XGBoost entrena árboles secuencialmente, donde cada árbol se enfoca en corregir los errores de predicción del árbol anterior*. XGBoost utiliza regularización y técnicas de poda para evitar el sobreajuste y mejorar la precisión del modelo. Es conocido por su velocidad y rendimiento sobresaliente en una amplia gama de problemas de aprendizaje automático.

La complejidad del DataSet, no solamente el tamaño y el desbalance entre valores fraude y no fraude sino también el desconocimiento de la teoría detrás de las variables ‘Vi’ nos llevaron a tener que descartar los dos primeros modelos. No se eligió el algoritmo XGBoost porque generase un modelo un poco mejor a los algoritmos anteriores, sino que los primeros dos algoritmos quedaron completamente descartados porque, como se puede evidenciar a continuación, el área bajo la curva ROC mostraba que el modelo generado no tenía capacidad predictiva.

Métricas de desempeño del modelo

Para poder evaluar el modelo predictivo creado se utilizó la técnica *Stratified K Fold*.

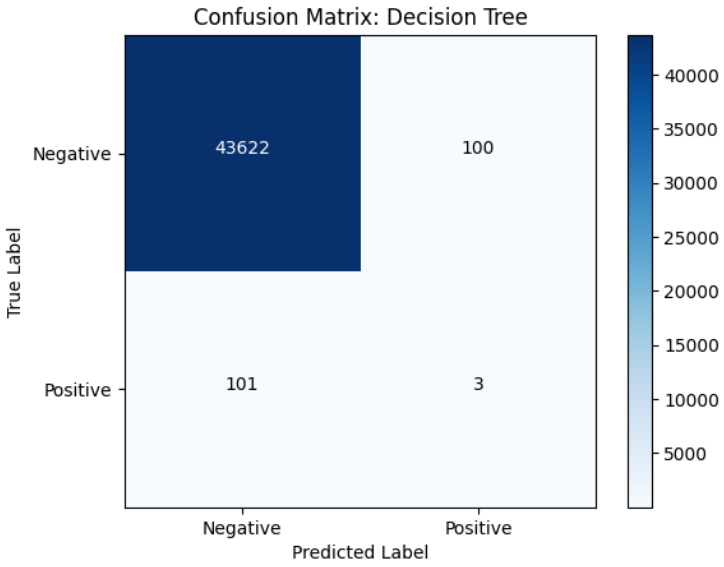
En la validación cruzada K-Fold estándar, el conjunto de datos se divide en K partes iguales, llamadas "pliegues" (folds), donde el modelo se entrena K veces, utilizando un pliegue diferente como conjunto de prueba en cada iteración y los restantes como conjunto de entrenamiento. Sin embargo, en el caso de conjuntos de datos desbalanceados, donde una clase puede estar subrepresentada, la validación cruzada estándar puede no ser adecuada.

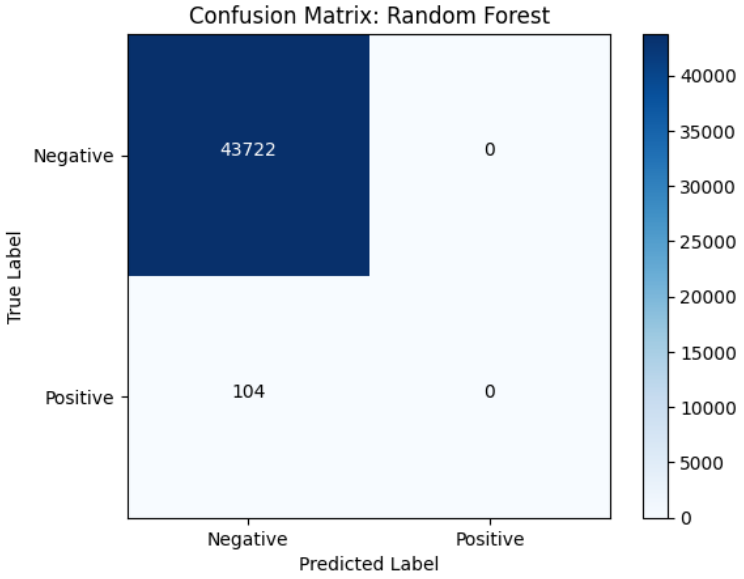
Stratified K-Fold aborda este problema al mantener la proporción de clases en cada pliegue lo más cercana posible a la proporción original en el conjunto de datos completo. Esto significa que cada pliegue tendrá una distribución similar de las clases de la variable objetivo. Considerando el desbalance ya mencionado de nuestro DataSet, Stratified K Fold resulta ser la opción más correcta para validar el modelo.

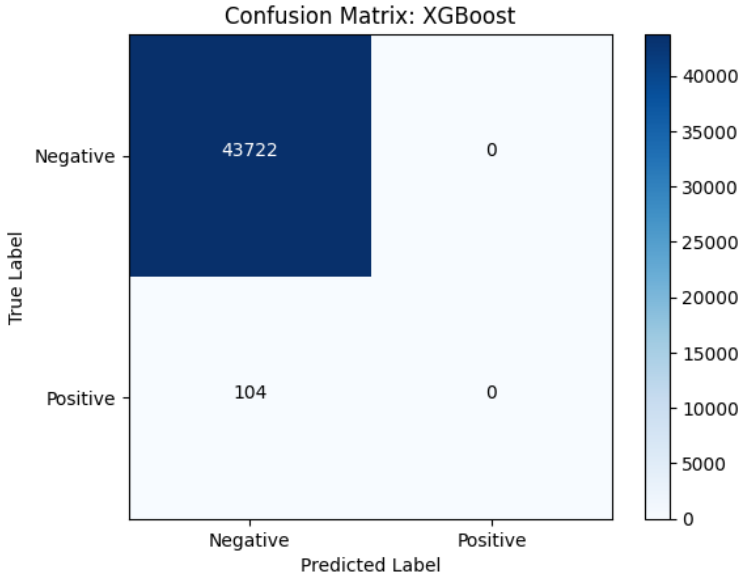
Para evaluar los modelos se priorizaron las siguientes métricas

- **Área bajo la curva ROC (AUC-ROC)**: Mide la capacidad del modelo para distinguir entre clases. Es útil cuando el conjunto de datos es desbalanceado y hay que evaluar el rendimiento en diferentes umbrales de clasificación.
- **Matriz de Confusión**: Proporciona una descripción detallada del rendimiento del modelo, mostrando el número de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos

Métricas/Modelo	**Árbol de Decisión**	**Random Forest**	**XGBoost**
Área bajo la curva ROC	0.5	0.5	0.81

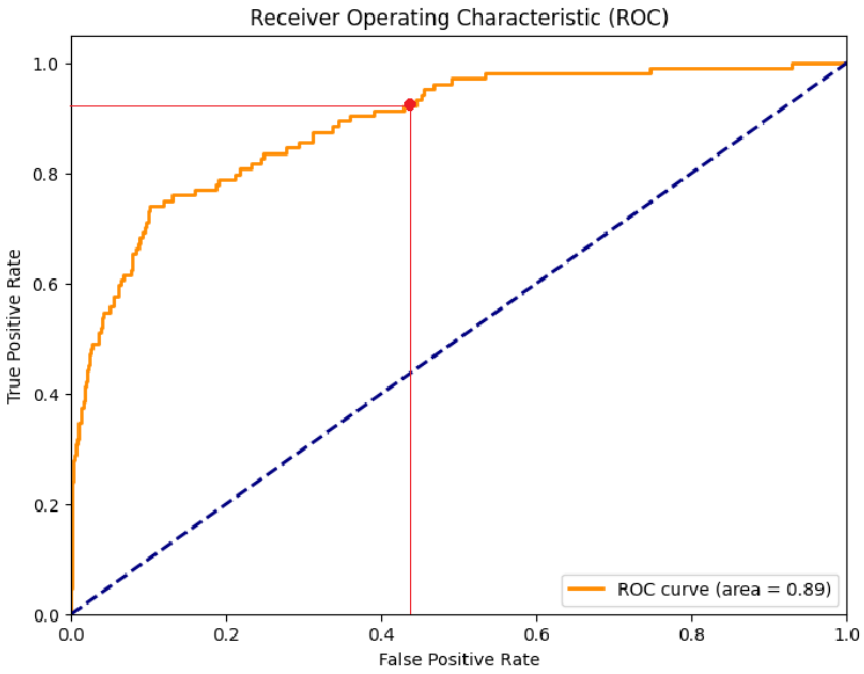






Como podemos observar, el modelo XGBoost está considerando que ninguno de los registros es un caso de fraude. Esto se debe a que el modelo está determinando si los casos son positivos o negativos en base a la probabilidad de que sean positivos o negativos (tpr, fpr). Considerando el desbalance del DataSet trabajado, la probabilidad de que el registro sea negativo será siempre ampliamente superior a la probabilidad de que sea positivo. Es por esto que, como paso final, necesitamos evaluar el threshold con el que el modelo está trabajando.

Para hacer esto, empezaremos por hacer una inspección visual de la curva ROC y determinar cuál es el porcentaje deseable de positivos que debemos detectar en relación al porcentaje de falsos positivos que estamos dispuestos a tolerar. Cabe aclarar que, debido a la lógica en la que se basa el modelo y la curva ROC representativa, un mayor porcentaje de aciertos implica también un mayor porcentaje de falsos positivos. Por eso es que determinaremos un valor ‘deseable’, pero esto puede ajustarse en el futuro sin modificar el modelo.



Considerando el modelo de negocio planteado, es más grave no detectar un caso de fraude que asumir como fraude casos que no lo son y poder descartarlos posteriormente. Es decir, decidimos permitir un Error de tipo I (falso positivo) alto – alrededor del 40% - para poder asegurar un Error de tipo II (falso negativo) bajo – alrededor de 7%.

Esta decisión está basada en la suposición de que los falsos positivos pueden ser detectados y descartados posteriormente sin consecuencias problemáticas. Es decir, el modelo predictivo será un primer paso de detección de fraude, pero, considerando la complejidad del Data Set y las limitaciones naturales previamente mencionadas, necesitará de una segunda instancia donde se haga una evaluación para determinar si los casos detectados como positivos son efectivamente fraude o simplemente 'movimientos sospechosos'. Considerando esto, se prefiere tener que emplear recursos en evaluar algunas operaciones para determinar que no sean fraudes - por ejemplo, contactándose con el cliente - que tener que responder por operaciones fraudulentas no detectadas. Incluso el trabajo posterior para la evaluación de los falsos positivos puede dar inicio a la creación de otra base de datos que sirva para una retroalimentación del modelo inicial o para un segundo modelo que trabaje de forma consecutiva con el primero.

Esta decisión podría ser inversa. Por ejemplo, podría determinarse los niveles de error tipo I y II a establecer en base al costo que cada uno tenga asociado. Si contásemos con esta información, podríamos crear una función que las relacione para calcular el costo total y buscar los valores de error que lo minimizaran. Como esto no es posible dado que no contamos con esta información, optamos por seguir la guía establecida por las preguntas planteadas al inicio y priorizar detectar los casos para poder así frenar el avance de estas prácticas y alcanzar, también, una mejor imagen positiva de la empresa en cuanto al cuidado de las operaciones de sus clientes.

Iteraciones de optimización

Considerando que, como fue explicado en el inciso anterior, hay variables que aportan mucho menos valor al modelo que otras, se procedió a evaluar el modelo con menos variables y evaluar su desempeño.

Se comprobó que, si bien las variables dejadas para el final suman poco al modelo, el modelo que incluye todas las variables es el que demuestra mejor capacidad predictiva. También resultó importante retener todas las variables debido a la naturaleza del algoritmo utilizado (XGBoost), considerando que esto implica que se produce una reducción en el sesgo, una mayor robustez y que habilita la interacción no lineal (es decir, la interacción entre variables independientes para la mejora del modelo).

Es importante resaltar que esto puede implicar un costo computacional mayor, pero es un análisis que excede el alcance de este trabajo. En nuestro caso, el modelo corrió sumamente rápido, por lo cual no se hizo una evaluación del costo computacional de las distintas posibilidades.

Métricas finales del modelo optimizado

Para evaluar el rendimiento del modelo predictivo se utilizó el método de área bajo la curva ROC. Esta es una medida utilizada para evaluar la capacidad predictiva de un modelo de clasificación binaria. La curva ROC es una representación gráfica que muestra la relación entre la tasa de verdaderos positivos (TPR) y la tasa de falsos positivos (FPR) para diferentes umbrales de clasificación.

Se corrió mediante 5 folds y el valor promedio fue de 0.8.

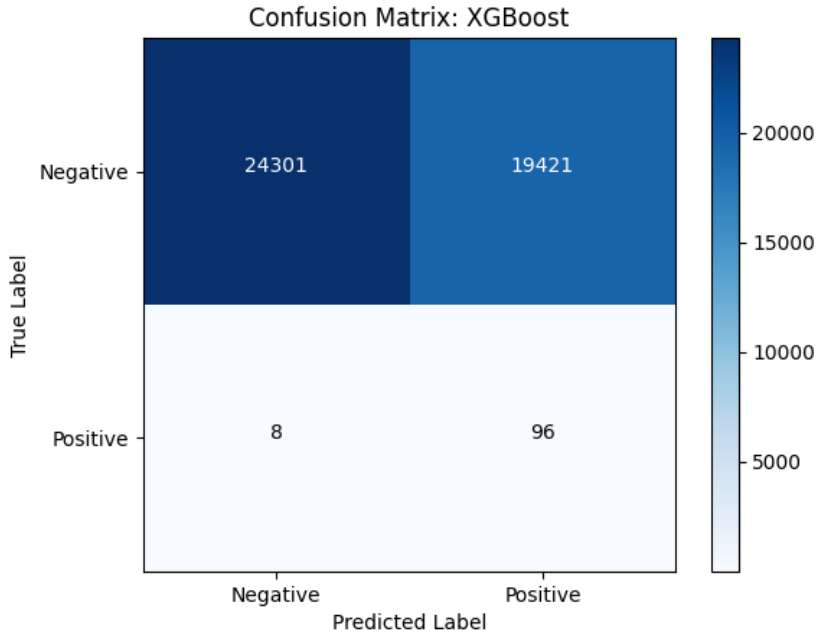
Por otro lado, como fue previamente explicado, se utilizó la matriz de confusión para establecer la cantidad de verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Podemos ver la matriz final y los porcentajes que cada uno de estos representa.

<u>Positivos</u>

- Verdaderos 93%
- Falsos 7%

<u>Negativos</u>

- Verdaderos = 56%
- Falsos = 44%



Futuras Líneas

Si bien consideramos que el modelo creado tiene una buena capacidad predictiva (explicado anteriormente en base al ROC siendo un 0.8), en líneas futuras, a medida que siga creciendo el DataSet, será importante seguir retroalimentándolo. Esto podrá permitir que el modelo aprenda respecto a valores que hoy en día se consideran outliers o traer outliers nuevos de los cuales el modelo se pueda nutrir (lo cual es una de las principales características que diferencia al algoritmo XGBoost de los otros – los outliers, lejos de complicarlo, lo fortalecen).

Por otro lado, como fue mencionado previamente, el modelo tiene una tendencia alta a tener falsos positivos. Por lo que podría crearse un segundo modelo que trabajara exclusivamente con los casos detectados como positivos (tanto falsos como reales) para que pueda distinguir entre ellos y tener los dos modelos trabajando en serie para permitir una mejor purga y reducir el trabajo manual que se necesitará al principio.

En tercer lugar, los falsos negativos serán probablemente comunicados por clientes cuando vean sus resúmenes de tarjetas, por lo que también será necesario volverlos a incorporar a la base de datos para que el modelo siga aprendiendo a detectarlos.

Conclusiones

Como conclusión al trabajo y en respuesta a nuestras preguntas originales –

- **Sí** – se puede predecir las operaciones fraudulentas en base al Data Set proporcionado mediante la creación de un modelo.

- Las principales variables que nos ayudarán a detectar las operaciones fraudulentas son las previamente destacadas de acuerdo al aporte que tienen hacia el modelo predictivo: **V3**, **V2**, **V14** y **V10**. Pero, como también fue aclarado previamente, el resto de las variables tiene que estar incluido en el modelo, a pesar de que su aporte individual sea bajo, dado que su aporte global es importante.

- Si bien existen múltiples modelos predictivos, dada la naturaleza de los datos y del tipo de problema (clasificación) que estamos intentando resolver, el modelo que mejor se ajusta es el **XGBoost**.

Considerando la naturaleza del problema de negocio, es preferible tomar una operación normal por fraudulenta e investigarla para descartarla (falso positivo) que asumir que una operación fraudulenta no lo es e incurrir en las consecuencias del caso (falso negativo). Por lo que el modelo estará orientado a minimizar los errores de tipo 1, es decir falso negativo, buscando un error menor al 7%, y será más laxo con los errores de tipo 2, falso negativo, teniendo que manejar, en un principio, un error del 44% aproximadamente.