Proyecto Detección de Fraudes

Iván Bozo, Nicolás Muñoz, Lukas Flores

Julio 2023

1 Introducción

En la era digital contemporánea, el fraude financiero ha emergido como un desafío significativo. La proliferación de estafas, robos de identidad y fraudes con tarjetas de crédito no solo ha causado daños económicos a las víctimas, sino que también ha infligido pérdidas millonarias a las instituciones financieras. Estos actos delictivos han erosionado la confianza en el sistema financiero global, subrayando la necesidad de soluciones robustas y efectivas.

Dentro de este contexto, la ciencia de datos y la minería de datos han tomado un papel protagonista en la búsqueda de soluciones. La detección y prevención de fraudes financieros se ha establecido como un campo de estudio crucial, donde el aprendizaje automático ofrece herramientas prometedoras para abordar el problema. El propósito central de este informe es evaluar y comparar el rendimiento de varios clasificadores en la detección de fraudes utilizando el conjunto de datos "Synthetic Financial Datasets For Fraud Detection" de Kaggle. Este conjunto, generado sintéticamente, simula transacciones financieras, brindando una plataforma ideal para analizar cómo diferentes modelos abordan la detección de fraudes en un entorno controlado.

El desafío principal radica en desarrollar un modelo que pueda identificar transacciones fraudulentas en este conjunto de datos, que presenta características numéricas y categóricas y una variable objetivo binaria 'is Fraud'. Se explorarán múltiples técnicas de aprendizaje, incluyendo enfoques supervisados y no supervisados, teniendo en cuenta el desequilibrio de clases inherente al conjunto de datos.

Además de evaluar la eficacia de los clasificadores, este informe se adentrará en los enfoques y metodologías subyacentes, proporcionando una visión detallada de las estrategias empleadas en la detección de fraudes. A través de este análisis comparativo, buscamos no solo entender la efectividad de cada clasificador, sino también identificar las mejores prácticas en el ámbito de la detección de fraudes.

2 Marco Teórico

2.1 Detección de Fraudes Financieros

La detección de fraudes financieros es una tarea crítica en la era digital. A medida que las transacciones electrónicas se vuelven más prevalentes, es esencial identificar patrones anómalos que sugieran estafas, robos de identidad o malversaciones, con el objetivo de proteger tanto a los consumidores como a las instituciones financieras.

2.2 Aprendizaje Automático en la Detección de Fraudes

El aprendizaje automático, al analizar y aprender de grandes conjuntos de datos, se ha posicionado como una herramienta esencial en la detección de fraudes.

2.2.1 Aprendizaje Supervisado

Se basa en modelos que se entrenan con datos previamente etiquetados.

- **Bosque Aleatorio:** Método de ensamble que utiliza múltiples árboles de decisión. Las métricas de evaluación comunes incluyen exactitud, recall, precisión y F1 score.

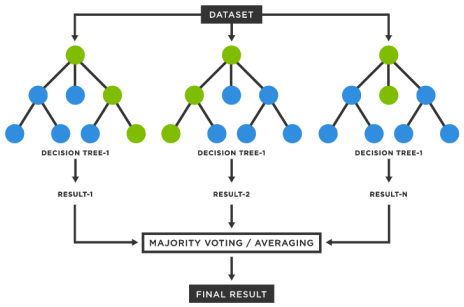

Figure 1: Ejemplo del Bosque Aleatorio.

- **K-Vecinos (KNN):** Clasifica una entrada basada en las etiquetas de sus vecinos más cercanos en el conjunto de entrenamiento. Se evalúa típicamente usando exactitud, recall, precisión y F1 score.

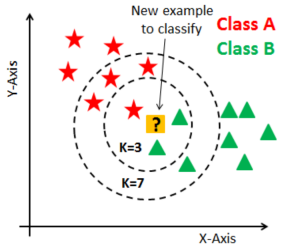

Figure 2: Ejemplo de K-Vecinos.

- **Perceptrón Multicapa:** Red neuronal artificial que aprende funciones no lineales. Las métricas de evaluación incluyen exactitud, recall, precisión y F1 score.

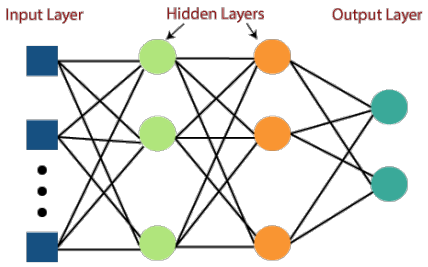

Figure 3: Ejemplo de Perceptrón Multicapa.

2.2.2 Aprendizaje No Supervisado

Identifica patrones en datos sin etiquetas previas.

- **K-means:** Algoritmo de clustering que agrupa datos en K grupos distintos. Se evalúa utilizando métricas como la Inercia, el Índice Calinski-Harabasz y el Silhouette Score.

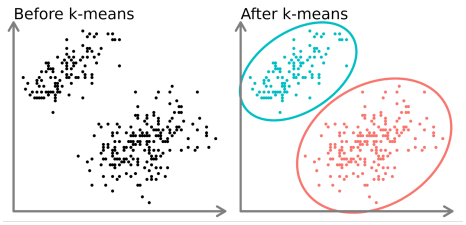

Figure 4: Ejemplo de K-means.

2.2.3 Aprendizaje Semi-Supervisado

Redes Neuronales Artificiales (RNA): Se han consolidado como herramientas esenciales para descubrir patrones y relaciones complejas en grandes conjuntos de datos. Una de las arquitecturas más fundamentales y ampliamente adoptadas dentro de las RNA es la red neuronal feedforward.

- **Red Neuronal Feedforward**: Esta arquitectura se caracteriza por tener flujos de información que avanzan en una sola dirección, desde la entrada hasta la salida, sin retroalimentaciones ni bucles. Su diseño permite modelar relaciones no lineales y adaptarse a una amplia variedad de problemas.

- **Inicialización**: La elección de cómo se inicializan los pesos en una red neuronal es crucial para su entrenamiento y convergencia. Técnicas como la inicialización sigmod buscan proporcionar un punto de partida óptimo para el proceso de aprendizaje, asegurando que las activaciones no se desvanezcan ni se saturen en las primeras etapas.

- **Capa de Entrada**: Sirve como la puerta de entrada a la red, recibiendo los datos y pasándolos a las capas subsiguientes. La forma y estructura de esta capa se determinan según la naturaleza y dimensiones de los datos con los que se trabaja.

- **Capas Densas**: Estas capas, también conocidas como "completamente conectadas", son el núcleo computacional de la red. Cada neurona en una capa densa está interconectada con todas las neuronas de las capas adyacentes. Las funciones de activación, como la ReLU, se utilizan para introducir no linealidades en el modelo, permitiendo que la red capture relaciones más complejas.

- **Dropout**: En el contexto de la minería de datos, la regularización es esencial para garantizar que el modelo generalice bien a datos no vistos. Dropout es una técnica de regularización que busca prevenir el sobreajuste al desactivar aleatoriamente un subconjunto de neuronas durante el entrenamiento, promoviendo así una distribución más robusta de las características aprendidas.

- **Capa de Salida**: Esta capa produce la predicción final de la red. Dependiendo del problema, puede tener una función de activación específica, como la sigmoide para clasificación binaria, que mapea las salidas a un rango específico.

- **Compilación**: Una vez que se ha definido la estructura de la red, es necesario especificar cómo se entrenará. Esto implica seleccionar una función de pérdida adecuada al problema en cuestión y un optimizador, que determinará cómo se ajustan los pesos de la red para minimizar dicha función de pérdida.

2.3 Self-training

El *Self-training* es una técnica de aprendizaje semisupervisado que busca combinar las ventajas del aprendizaje supervisado y no supervisado. La idea central es utilizar un modelo inicial entrenado con datos etiquetados para predecir etiquetas en un conjunto de datos no etiquetados. Estas predicciones, cuando se tienen con alta confianza, se consideran como "etiquetas pseudo" y se reincorporan al conjunto de entrenamiento.

- **Principio Básico**: Se inicia con un modelo entrenado exclusivamente con datos etiquetados. Este modelo se utiliza para hacer predicciones sobre un conjunto de datos no etiquetados.
- **Selección de Datos**: De las predicciones realizadas, se seleccionan aquellos datos para los cuales el modelo muestra una alta confianza en su predicción. Estas predicciones se convierten en "etiquetas pseudo".
- **Reentrenamiento**: Los datos con etiquetas pseudo se combinan con el conjunto original de datos etiquetados. El modelo se reentrena con este conjunto ampliado.
- **Iteración**: El proceso puede repetirse, cada vez incorporando más datos no etiquetados con alta confianza en el conjunto de entrenamiento.
- **Ventajas**: El *Self-training* permite aprovechar la información contenida en los datos no etiquetados, potencialmente mejorando la precisión y robustez del modelo sin requerir etiquetado manual adicional.
- **Consideraciones**: Es esencial ser cauteloso al seleccionar qué datos no etiquetados se reincorporarán, ya que las etiquetas incorrectas pueden afectar negativamente el rendimiento del modelo.

En esencia, el *Self-training* es una estrategia que busca maximizar la utilidad de los datos disponibles, tanto etiquetados como no etiquetados, para mejorar la eficacia de los modelos en la minería de datos.

2.4 Métricas de Evaluación

Para evaluar la eficacia y precisión de los modelos en la tarea de detección de fraudes, se utilizan diversas métricas. A continuación, se detallan las métricas empleadas:

2.4.1 Para aprendizaje supervisado y semi-supervisado

- **Exactitud (Accuracy):** Es la proporción de predicciones correctas entre el total de predicciones realizadas. Representa una medida general de la eficiencia del modelo.

- **Recall o Sensibilidad:** Indica la proporción de positivos reales que fueron identificados correctamente. Es especialmente importante en contextos donde los falsos negativos tienen un alto costo, como la detección de fraudes.
- **Precisión:** Representa la proporción de identificaciones positivas que fueron realmente correctas. Es útil para entender cuántos de los identificados como positivos son verdaderos positivos.
- **F1 Score:** Es la media armónica de la precisión y el recall. Proporciona un balance entre ambas métricas, siendo especialmente útil cuando las clases están desbalanceadas.

2.4.2 Para aprendizaje no-supervisado

- **Silhouette:** La métrica Silhouette mide cuán similar es un objeto a su propio grupo (cohesión) en comparación con otros grupos (separación). Los valores oscilan entre -1 y 1. Un valor alto indica que el objeto está bien emparejado con su propio grupo y mal emparejado con grupos vecinos.
- **Inercia:** Conocida también como suma de cuadrados dentro del grupo, mide la cohesión de los grupos. Es la suma de las distancias al cuadrado de las muestras al centro de su grupo más cercano. Siempre es un valor no negativo, y valores más bajos son mejores.
- **Calinski-Harabasz:** Este índice compara la dispersión entre grupos con la dispersión dentro de los grupos. Es el cociente entre la suma de cuadrados entre grupos y la suma de cuadrados dentro del grupo. Valores más altos indican una mejor agrupación.

2.5 Desafío del Desequilibrio de Clases

El desequilibrio de clases es un problema recurrente en la detección de fraudes, donde las transacciones legítimas suelen superar ampliamente a las fraudulentas. Para abordar este problema miraremos las siguientes tecnicas:

2.5.1 Submuestreo (Undersampling):

El undersampling es una técnica de muestreo de datos que se utiliza para reducir el tamaño de un conjunto de datos. Esto se hace eliminando algunas de las muestras del conjunto de datos. El undersampling se utiliza a menudo para mejorar el rendimiento de los modelos de aprendizaje automático en conjuntos de datos desequilibrados, que son conjuntos de datos en los que hay una gran diferencia en el número de muestras de cada clase.
Ventajas:

- **Reducción del tiempo de entrenamiento:** Al reducir el tamaño del conjunto de datos, el tiempo y los recursos necesarios para entrenar modelos se disminuyen significativamente.

- **Simplicidad:** El submuestreo es una técnica directa y fácil de implementar.
- **Eliminación de ejemplos redundantes:** Puede ayudar a eliminar ejemplos repetitivos o muy similares en la clase mayoritaria que no aportan información adicional.

Desventajas:

- **Pérdida de información:** Al eliminar ejemplos de la clase mayoritaria, se corre el riesgo de perder información potencialmente valiosa.
- **Riesgo de subrepresentación:** Si no se realiza correctamente, el submuestreo puede llevar a que la clase mayoritaria esté subrepresentada, lo que puede afectar negativamente el rendimiento del modelo.
- **Variabilidad:** Diferentes rondas de submuestreo pueden producir resultados diferentes, lo que puede llevar a una variabilidad en el rendimiento del modelo.
- **No adecuado para conjuntos de datos pequeños:** En conjuntos de datos ya pequeños, el submuestreo puede reducir aún más el tamaño, lo que puede no ser adecuado para entrenar un modelo efectivo.

2.5.2 Sobremuestreo con SMOTE(Synthetic Minority Over-sampling Technique):

A diferencia del undersampling, el oversampling es una técnica de muestreo de datos que aumenta el tamaño de un conjunto de datos duplicando algunas de las muestras del conjunto de datos. El oversampling se utiliza a menudo para mejorar el rendimiento de los modelos de aprendizaje automático en conjuntos de datos desequilibrados.

Ventajas:

- **Generación de ejemplos sintéticos:** En lugar de replicar ejemplos existentes, SMOTE crea ejemplos sintéticos basados en la interpolación entre ejemplos reales, lo que puede mejorar la generalización del modelo.
- **Reducción del riesgo de sobreajuste:** Al generar ejemplos sintéticos en lugar de simplemente duplicar ejemplos existentes, SMOTE puede reducir el riesgo de sobreajuste en comparación con el sobremuestreo tradicional.
- **Flexibilidad:** SMOTE permite ajustar el grado de generación de ejemplos sintéticos, lo que da a los practicantes control sobre cuánto sobremuestreo quieren realizar.
- **Combinable con otras técnicas:** SMOTE se puede combinar con otras técnicas de muestreo para lograr un equilibrio de clases más refinado.

Desventajas:

- **Generación de ruido:** Al crear ejemplos sintéticos, SMOTE puede generar datos que no representan patrones reales, introduciendo ruido.
- **Aumento del tiempo de entrenamiento:** Aumentar el tamaño del conjunto de datos puede resultar en un mayor tiempo de entrenamiento.
- **No siempre es efectivo:** En algunos casos, SMOTE puede no mejorar el rendimiento del modelo y, en cambio, puede empeorarlo.
- **Complejidad:** SMOTE requiere la elección de parámetros, como la cantidad de vecinos a considerar, lo que puede afectar la calidad de los ejemplos sintéticos generados.

2.5.3 Muestreo Mixto :

Esta técnica busca aprovechar las ventajas de ambos métodos para crear un conjunto de datos balanceado que sea representativo y que reduzca el riesgo de sobreajuste.
Ventajas:

- **Equilibrio de clases:** Combina las ventajas del sobremuestreo y el submuestreo para equilibrar las clases sin perder demasiada información ni introducir demasiada redundancia.
- **Flexibilidad:** Permite a los practicantes ajustar el grado de submuestreo y sobremuestreo según las necesidades del problema específico.
- **Mejora del rendimiento del modelo:** Al equilibrar las clases, los modelos suelen tener un mejor rendimiento en la clase minoritaria sin comprometer demasiado la precisión en la clase mayoritaria.

Desventajas:

- **Pérdida de información:** El submuestreo puede llevar a la eliminación de ejemplos potencialmente útiles de la clase mayoritaria.
- **Sobreajuste:** El sobremuestreo puede causar sobreajuste, ya que replica ejemplos existentes, lo que puede hacer que el modelo memorice esos ejemplos en lugar de generalizar.
- **Costo computacional:** Aumentar el tamaño del conjunto de datos mediante sobremuestreo puede aumentar el tiempo y los recursos necesarios para entrenar modelos.

3 Metodología

3.1 Preprocesamiento de Datos

El primer paso en nuestro análisis fue la selección y limpieza de los datos. A partir del conjunto de datos "Synthetic Financial Datasets For Fraud Detection", se eliminaron las columnas `nameOrig`, `nameDest` e `isFlaggedFraud`, ya que no aportaban información relevante para la detección de fraudes. Además, se seleccionaron únicamente las transacciones categorizadas como `cash_out` y `transfer` en la columna `type`, ya que en el muestreo mixto con una cantidad baja de datos se detecto que los fraudes estaban predominantemente asociados a estas categorías como en fig 6, por lo que ademas se filtraron los datos que no fueran `cash_out` o `transfer` para solo tener datos de interes para los modelos.

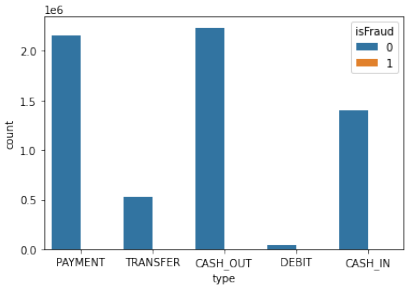

Figure 5: Observacion de fraudes en todo el conjunto (6.3millones).

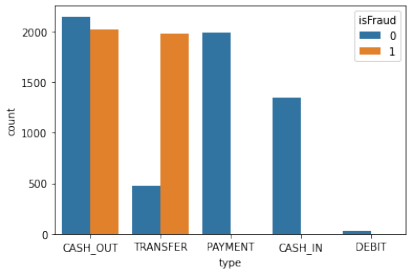

Figure 6: Observacion de fraudes en 10mil datos con muestreo mixto.

3.2 Desbalance en las clases

Para abordar el desbalance en las clases, se aplicaron técnicas de balanceo como el muestreo mixto y SMOTE en los datos destinados a modelos de aprendizaje supervisado y no supervisado. En el caso del aprendizaje semi-supervisado, se llevó a cabo un balanceo de clases con diferentes proporciones entre fraudes y no fraudes.

Tal como se observa en la figura 5, la cantidad de fraudes en el conjunto de datos es mínima. En específico, corresponde al 0.1% (8.213) de los datos totales (6.354.407). Para abordar esto, se aplicaron las siguientes técnicas:

- **Muestreo Mixto:** Se seleccionaron 10.000 datos, de los cuales 4.000 son fraudes y el resto no lo son. Esta cantidad de datos fue elegida con el propósito de entender el comportamiento con un volumen reducido de datos. Encontramos hallazgos interesantes, como se muestra en la fig6. En este subconjunto del conjunto de datos, contamos con 5.297 datos para entrenamiento y 1.325 para prueba, es decir, una división de 80/20.

- **SMOTE (Synthetic Minority Over-sampling Technique):** Al importar la biblioteca "imblearn.over_sampling", podemos aplicar esta técnica de forma casi directa. Inicialmente, tomamos una muestra de 50.000 datos. Posteriormente, filtramos los datos que corresponden únicamente a 'CASH_OUT' o 'TRANSFER', resultando en 18.456 datos para entrenamiento (13.842) y validación (4.614). Es importante mencionar que al generar ejemplos sintéticos, la validación debe realizarse con datos reales para que refleje la distribución auténtica de los datos.

- **Submuestreo:** Para el aprendizaje semi-supervisado, se trabajó en un archivo distinto con el objetivo de simplificar la visualización del código. Se aplicó el mismo preprocesamiento que en los modelos anteriores. Esto nos proporcionó un total de 2.770.409 datos para dividir en 'X' e 'Y' y aplicar la técnica de submuestreo de la función 'get_data'. En los argumentos 'x_training' e 'y_training', se toman los datos de entrenamiento y sus etiquetas. Con 'examples_per_class', elegimos la cantidad de datos que tendrá la clase objetivo para que esté balanceada, es decir, 50/50 en términos porcentuales. Al haber filtrado previamente y seleccionado solo los datos de interés, obtenemos un conjunto de 2.770.409 datos. Al designar una parte para el conjunto de entrenamiento, nos quedamos con 1.933.537 datos en los cuales el modelo busca equilibrar entre fraudes y no fraudes. Sin embargo, al contar solo con 5.749 datos de fraude que la función identifica en los datos que pasamos en los parámetros 'X' e 'Y', los datos de entrenamiento no quedan 50/50. En su lugar, tenemos un 22,3% de datos de fraude y el resto no lo son. A pesar de ello, esta proporción sigue siendo útil para experimentar con el modelo, presentando un desbalance menos acentuado que el del conjunto de datos original.

3.3 Modelos Utilizados

3.3.1 Aprendizaje Supervisado:

Se emplearon tres modelos distintos: Bosque Aleatorio, K-Vecinos y Perceptrón Multicapa. Todos los modelos se entrenaron con datos previamente procesados y se evaluaron usando métricas como exactitud, recall, precisión y F1 score.

- **Bosque Aleatorio:** Se diseñó una función multifuncional que, al ser invocada, realiza diversas tareas. Esta función acepta cuatro parámetros: los conjuntos de entrenamiento y prueba, tanto para las características como para las etiquetas. La función instancia el clasificador con 100 árboles de decisión, lo entrena, realiza predicciones y evalúa su rendimiento. Finalmente, se visualiza la matriz de confusión correspondiente.

- **K-Vecinos:** Se adaptó un proceso similar al del Bosque Aleatorio, pero utilizando las funciones específicas para el modelo K-Vecinos. Durante la instanciación, se especificó considerar los tres vecinos más cercanos.

- **Perceptrón Multicapa:** Se construyó un modelo con dos capas densas. La primera capa tiene 64 perceptrones con función de activación relu, mientras que la segunda capa, destinada a la clasificación binaria, tiene un perceptrón con función de activación sigmoid. Tras compilar el modelo con la función de pérdida *binary crossentropy* y el optimizador *adam*, se entrenó especificando 10 épocas y un tamaño de lote de 32. Las predicciones se obtuvieron y evaluaron de manera similar a los modelos anteriores, añadiendo la métrica de pérdida.

3.3.2 Aprendizaje No Supervisado:

Se implementó la función `train_kmeans` que acepta los datos de entrenamiento 'X' y un número de clusters 'n_clusters'. Una vez entrenado el modelo, se evalúa con la función `evaluate_kmeans`, que toma como argumentos los datos 'X' y el modelo entrenado 'kmeans_model'. Las métricas utilizadas para evaluar la calidad de los clusters son la inercia, el índice Calinski-Harabasz y el coeficiente de silueta. Para visualizar los resultados, se realizó una reducción de dimensionalidad con PCA y se graficó en 2D, como se puede observar en la figura 7.

3.3.3 Aprendizaje Semi-Supervisado:

El proceso de *self-training* se llevó a cabo siguiendo una serie de pasos estructurados. A continuación, se detalla el proceso:

- **Inicialización:** Se inicializan dos listas, `fscore` y `cantidad_de_datos`, para registrar los puntajes F1 y la cantidad de datos etiquetados en cada iteración, respectivamente.
- **Preparación de Datos:**
  - **Conjunto de Entrenamiento**: Se formó un conjunto balanceado de entrenamiento con 25,749 ejemplos, de los cuales 5,749 son transacciones fraudulentas y 20,000 son transacciones legítimas. Las dimensiones específicas del conjunto son `train x: (25749, 7)` y `train y: (25749,)`.
  - **Conjunto de Validación**: Se estableció un conjunto de validación con 2,000 ejemplos por clase, resultando en dimensiones de `val x: (2000, 7)` y `val y: (2000,)`.
  - **Datos sin Etiquetas**: El volumen de datos sin etiquetas fue de `unlabeled x: (1911537, 7)`.
- **Iteración**: La función realiza iteraciones determinadas por el argumento **veces**.
  - **Información de la Iteración**: Se muestra información sobre la iteración actual, incluyendo la cantidad de datos etiquetados y no etiquetados.

- **Entrenamiento del Modelo**: El modelo se entrena con el conjunto de datos etiquetados actuales durante 100 épocas.
- **Evaluación del Modelo**: Se emplea el modelo entrenado para realizar predicciones en el conjunto de prueba. Posteriormente, se calcula y registra el puntaje F1 de estas predicciones.
- **Predicción de Datos No Etiquetados**: Se utilizan las predicciones del modelo para etiquetar el conjunto no etiquetado. Los datos con predicciones más confiables se añaden al conjunto etiquetado.
- **Actualización de Conjuntos**: Los datos auto-etiquetados se retiran del conjunto no etiquetado y se incorporan al conjunto etiquetado.
- **Informe de Clasificación**: Se genera un informe que muestra métricas como precisión, recall y F1-score para cada clase.

- **Visualización**: Al concluir las iteraciones, se presenta una gráfica que relaciona el puntaje F1 con la cantidad de datos etiquetados en cada iteración.

La preparación y distribución de los conjuntos de datos reflejan la naturaleza desafiante de la detección de fraudes. Sin embargo, el enfoque semi-supervisado adoptado demostró ser efectivo en manejar este desbalance y en aprovechar la gran cantidad de datos sin etiquetas disponibles.

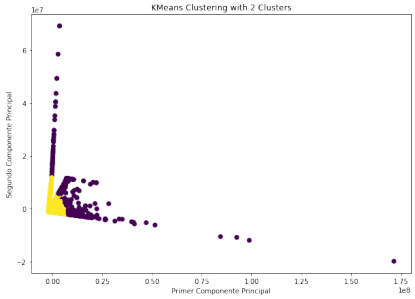

Figure 7: Visualizacion de kmeans con Muestro Mixto y pca=2.

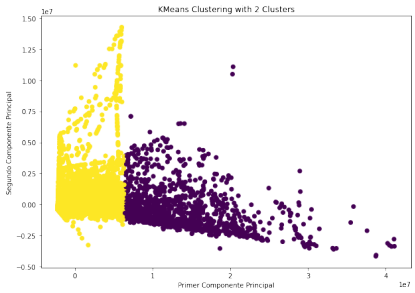

Figure 8: Visualizacion de kmeans con SMOTE y pca=2.

3.4 Métricas

3.4.1 Modelos Supervisados

Métricas Utilizadas: Para estos modelos, se emplearon métricas consolidadas para determinar su eficacia:

- **F1-Score:** Proporciona una evaluación conjunta de la precisión y exhaustividad.

- **Precisión (Precision):** Estima la proporción de identificaciones positivas que fueron correctas.
- **Exhaustividad (Recall):** Cuantifica cuántos de los casos positivos reales fueron identificados adecuadamente.
- **Exactitud:** Calcula el porcentaje total de predicciones correctas.

3.4.2 Modelos Semi-Supervisados

Métricas Utilizadas: En este enfoque, la evaluación se basó exclusivamente en:

- **F1-Score:** Para determinar la armonía entre la precisión y la exhaustividad del modelo.

3.4.3 Modelos No Supervisados

Métricas Utilizadas: Para los modelos no supervisados, la evaluación recurrió a métricas específicas para analizar la cohesión y diferenciación de los clusters:

- **Inercia:** Refleja la dispersión interna de los clusters.
- **Índice de Calinski-Harabasz:** Evalúa la relación entre la dispersión dentro del cluster y la dispersión entre clusters.
- **Coeficiente de Silhouette:** Mide cuán similar es un objeto respecto a su propio cluster en contraste con otros clusters.

4 Resultados y Análisis

4.1 Aprendizaje supervisado

4.1.1 Bosque aleatorio:

Con muestreo mixto se logró conseguir un modelo bastante excelente, con todas sus métricas sobre el 98%. Lo que más le dificulta a este modelo es hacer que sus predicciones positivas sean todas acertadas, pero incluso esto lo hace muy bien. Por otro lado, el modelo entrenado con la técnica de SMOTE no estuvo del todo bien, si bien predijo correctamente una proporción mayor de clases que el modelo anterior, pocas fueron de la clase positiva, lo cual no es muy útil cuando nuestro objetivo son estas clases.

Comparación Resultados del Bosque Aleatorio		
Métricas	Muestreo Mixto	SMOTE
Exactitud	98.64%	99.48%
Recall	99.38%	67.86%
Precisión	98.40%	55.88%
F1 Score	98.88%	61.29%

Table 1: Resultados

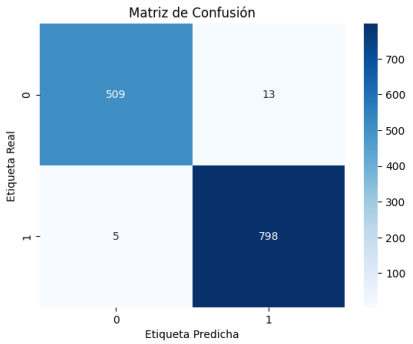

Figure 9: Matriz de confusión del bosque aleatorio con técnica de muestreo mixto.

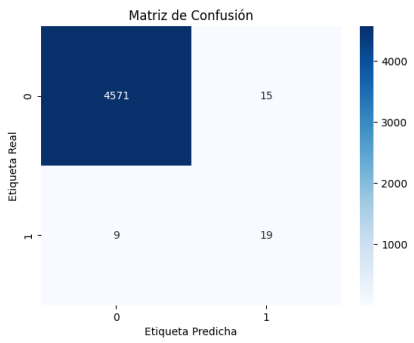

Figure 10: Matriz de confusión del bosque aleatorio con técnica de SMOTE.

4.1.2 K vecinos más cercanos:

Este algoritmo, en el caso del muestreo mixto, también resultó ser bastante bueno para crear modelos. Todas las métricas estuvieron sobre el 94%, siendo lo más bajo la precisión, con un 94.10%. Por otro lado, la técnica de SMOTE

fue similar al caso anterior, tuvo una gran exactitud, pero muy malos resultados en las demás métricas.

Comparación Resultados de K-vecinos más Cercanos		
Métricas	Muestreo Mixto	SMOTE
Exactitud	94.64%	97.62%
Recall	97.26%	64.29%
Precisión	94.10%	15.25%
F1 Score	95.65%	24.66%

Table 2: Resultados

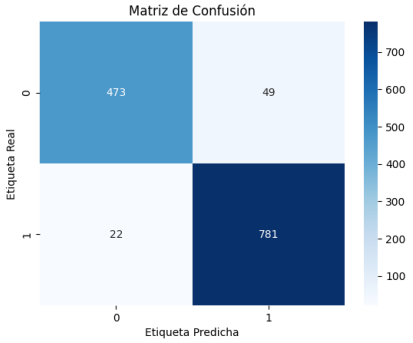

Figure 11: Matriz de confusión de k-vecinos más cercanos con técnica de muestreo mixto.

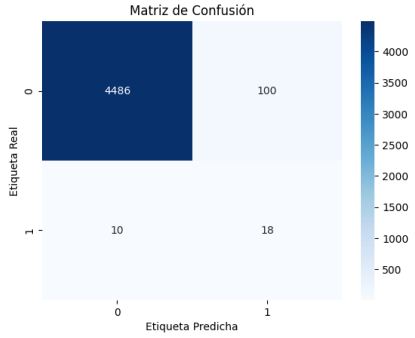

Figure 12: Matriz de confusión de k-vecinos cercanos con técnica de SMOTE.

4.1.3 Perceptrón multicapa:

Nuevamente la tecnica del muestreo mixto tuvo buenos resultados, todas sus metricas estuvieron sobre el 94%. Y nuevamente la técnica de SMOTE falló en la precisión, pero tuvo un gran rendimiento en el F1 score.

Comparación Resultados del Perceptrón Multicapa		
Métricas	Muestreo Mixto	SMOTE
Exactitud	95.32%	96.23%
Recall	97.76%	82.14%
Precisión	94.69%	11.98%
F1 Score	95.30%	97.60%

Table 3: Resultados

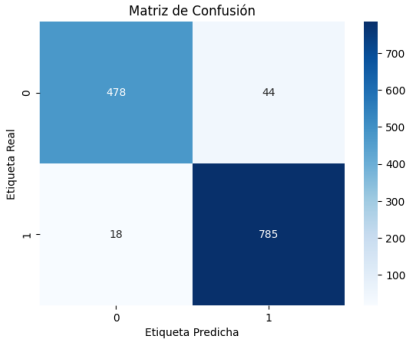

Figure 13: Matriz de confusión del perceptrón multicapa con técnica de SMOTE.

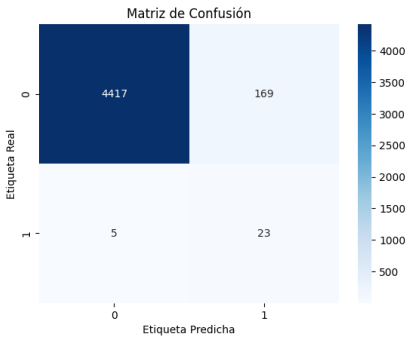

Figure 14: Matriz de confusión del perceptrón multicapa con técnica de SMOTE.

4.2 Resultados del Aprendizaje No Supervisado con K-means

Tras la implementación del algoritmo K-means, se obtuvieron resultados significativos en la detección de patrones dentro del conjunto de datos.

4.2.1 K-means con Muestro Mixto

La inercia resultante fue de 1.5 aprox, lo que sugiere que posee cohesión razonablemente buena dentro de los clusters. El índice Calinski-Harabasz alcanzó un valor de 2065.02, indicando un valor alto y por lo tanto que posee buena agrupación, mientras que el coeficiente de silueta fue de 0.79, lo que refleja una estructura de cluster bien definida en el entrenamiento de 5297 datos. La visualización de los clusters en 2D, gracias a la reducción de dimensionalidad con PCA, reveló una separacion entre los grupos de fraude y no fraude, la figura 7 ilustra estos resultados.

4.2.2 K-means con SMOTE

La inercia resultante fue de 1.39 aprox, lo que sugiere que al igual que con muestreo mixto posee cohesión razonablemente buena dentro de los clusters.

El índice Calinski-Harabasz alcanzó un valor de 35834.02, indicando un valor mas alto en comparacion con el muestreo mixto y por lo tanto que posee buena agrupación, mientras que el coeficiente de silueta fue de 0.82, lo que refleja una estructura de cluster bien definida en el entrenamiento de 19278 datos. Al igual que para muestreo mixto la visualización de los clusters en 2D, gracias a la reducción de dimensionalidad con PCA, reveló una separacion entre los grupos de fraude y no fraude la figura 8

4.3 Resultados del Aprendizaje Semi-Supervisado

El enfoque de *self-training* demostró ser efectivo en la ampliación del conjunto de datos etiquetados.

4.3.1 Modelo con Datos Desbalanceados:

- El modelo fue entrenado 4 veces con un conjunto de 25,749 datos, de los cuales 20,000 son transacciones legítimas y 5,749 son fraudulentas.
- Los F1 scores obtenidos en cada iteración son: [99.55785832752821, 99.56149418727954, 99.57724736361764, 99.81667666449088].
- Estas métricas sugieren una capacidad robusta del modelo para identificar transacciones fraudulentas.
- La evolución de esta métrica a lo largo del entrenamiento se visualiza en la figura 15.

4.3.2 Modelo con Datos Balanceados:

- El modelo fue entrenado 10 veces con un conjunto de 4,000 datos, equitativamente distribuidos entre transacciones legítimas y fraudulentas (2,000 de cada tipo).
- Los F1 scores obtenidos en cada iteración son: [99.788, 99.784, 99.783, 99.782, 99.794, 99.745, 99.739, 99.753, 99.768, 99.756].
- Estas métricas sugieren que el modelo entrenado con datos balanceados también posee una capacidad robusta para identificar transacciones fraudulentas.
- La evolución de esta métrica a lo largo del entrenamiento se visualiza en la figura 16.

En términos comparativos, el modelo entrenado con clases balanceadas alcanzó un F1 score de 99.794%. A pesar de esto, el modelo entrenado con datos desbalanceados mostró una ligera ventaja, alcanzando un pico de 99.816% en su capacidad de predicción de casos fraudulentos.

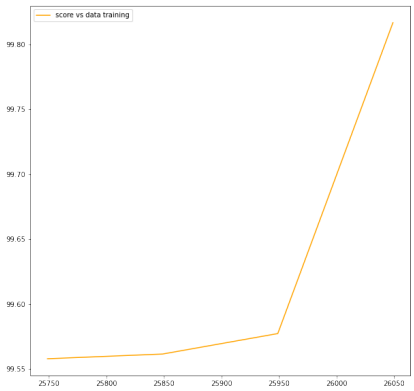

Figure 15: Score vs data training clases desbalanceadas.

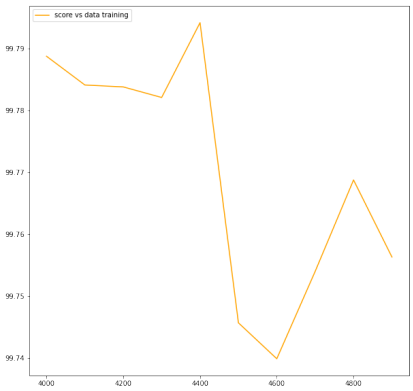

Figure 16: Score vs data training clases balanceados.

5 Conclusiones

- **Resumen de Resultados**:
  - **Aprendizaje supervisado:** se compararon tres algoritmos de clasificación (Bosque Aleatorio, K-vecinos más cercanos y Perceptrón Multicapa) utilizando dos técnicas de muestreo (muestreo mixto y SMOTE) en un problema de clasificación binaria con clases desbalanceadas. Se observó que el muestreo mixto proporcionó mejores resultados en todos los algoritmos, superando a la técnica de SMOTE en todas las métricas evaluadas (exactitud, recall, precisión y F1-Score). El modelo de Bosque Aleatorio con muestreo mixto mostró el rendimiento más consistente y mejor equilibrado entre todas las métricas evaluadas.
  - **Aprendizaje no supervisado:** con el algoritmo K-means, se utilizó el muestreo mixto y SMOTE para reducir el desequilibrio en los datos antes de aplicar el algoritmo. Ambas técnicas proporcionaron resultados significativos en la detección de patrones dentro del conjunto de datos. Sin embargo, el muestreo mixto presentó una inercia ligeramente mayor y un índice Calinski-Harabasz más bajo que SMOTE. La visualización en 2D reveló una separación clara entre los grupos de fraude y no fraude en ambos casos.
  - **Aprendizaje semi-supervisado:** se utilizó la técnica de self-training para ampliar el conjunto de datos etiquetados. Se entrenaron dos modelos: uno con datos desbalanceados y otro con datos balanceados. Ambos modelos demostraron una capacidad robusta para identificar transacciones fraudulentas, con F1-Scores superiores al 99%. El modelo entrenado con clases balanceadas mostró una leve ventaja en el F1-Score, alcanzando un pico de 99.794%, mientras que el

modelo entrenado con datos desbalanceados alcanzó un máximo de 99.816

En general, el muestreo mixto fue la técnica más efectiva para mejorar el rendimiento de los modelos en el problema de clasificación desbalanceada. Además, se observó que el algoritmo de Bosque Aleatorio con muestreo mixto mostró el mejor rendimiento y equilibrio entre las métricas evaluadas.

- **Impacto y Relevancia**:

  El impacto de estos resultados en el campo de estudio y en la problemática abordada es significativo. En el área de la seguridad financiera y la detección de fraudes, tener modelos más precisos y robustos puede tener un impacto directo en la reducción de pérdidas económicas para las instituciones financieras y en la protección de los usuarios contra actividades fraudulentas. Además, estos resultados pueden contribuir a mejorar la eficacia y eficiencia de los sistemas de detección de fraudes en tiempo real, lo que puede beneficiar tanto a las empresas como a los consumidores.

- **Limitaciones**:

  Debido al desbalanceo de clases, nos vimos en la tarea de modificar nuestro conjunto de datos. Nos encontramos varias soluciones de balanceo, cada una con sus ventajas y desventajas, sin embargo fue complicado identificar el método adecuado, a consecuencia de esto, utilizamos 2 métodos que nos parecieron los más correctos. Si bien la complejidad de la implementación aumentó, se aseguraron resultados excelentes.

- **Recomendaciones**:

  - **Análisis profundo de los datos:** estos pueden tener distintos problemas, tales como la duplicación o ausencia de datos, la irrelevancia de características, el tamaño del dataset, cantidad de clases, etc. Si no se analizan adecuadamente puede ser perjudicial para los modelos.
  - **Métodos de balanceo:** para el caso de detección de fraudes, es común encontrarse con clases desbalanceadas, por lo que hay que hacer uso de métodos de muestreo para balancearlos, de otra forma afectará negativamente al rendimiento de los modelos. Al momento de balacear los datos, debemos examinar la ventajas y desventajas de los distintos métodos, tales como el efecto en el tiempo de entrenamiento de los modelos, la complejidad del método, como se altera la información al aplicar un método, etc.
  - **Métricas a utilizar:** al momento de evaluar los modelos, se debe analizar cuales métricas es conveniente utilizar. Si estamos tratando con un problema de clases desbalanceadas es muy útil la métrica de f1 score para modelos de aprendizaje supervisado, porque de esta manera podemos medir que tan bueno es el modelo identificando la clase minoritaria.

- **Reflexión sobre Direcciones e Investigaciones Futuras**

  Los resultados obtenidos en este estudio demuestran el potencial del aprendizaje semi-supervisado en la detección de fraudes mediante técnicas de minería de datos. A partir de los hallazgos actuales, se pueden identificar varias direcciones prometedoras para investigaciones futuras:

  - **Expansión de Datos**: Aunque el aprendizaje semi-supervisado ha mostrado ser eficaz con el conjunto de datos actual, sería interesante explorar cómo se comporta con conjuntos de datos más grandes y diversos. Esto podría mejorar la robustez del modelo y su capacidad para detectar fraudes en diferentes contextos.
  - **Combinación de Técnicas**: La integración de técnicas de aprendizaje supervisado y no supervisado con el aprendizaje semi-supervisado podría ofrecer un enfoque híbrido que aproveche lo mejor de cada método. Esta combinación podría mejorar aún más la precisión y la capacidad de generalización del modelo.
  - **Aplicación en Otros Dominios**: Los métodos utilizados en este estudio podrían adaptarse y aplicarse a otros dominios que enfrenten problemas similares, como la detección de anomalías en sistemas de seguridad o la identificación de patrones inusuales en datos médicos.
  - **Evaluación de Nuevas Métricas**: Aunque las métricas de rendimiento actuales han sido efectivas, podría ser útil explorar otras métricas que ofrezcan una perspectiva más completa del rendimiento del modelo, especialmente en escenarios donde los datos de fraude son extremadamente desequilibrados.
  - **Exploración de Nuevas Tecnologías**: Con el avance constante de la tecnología y la inteligencia artificial, siempre hay nuevas herramientas y técnicas emergentes que podrían incorporarse en futuras investigaciones para mejorar la detección de fraudes.
  - **Aspectos Éticos y de Privacidad**: A medida que se desarrollan y aplican modelos más avanzados, es esencial considerar las implicaciones éticas y de privacidad. Las futuras investigaciones podrían centrarse en cómo garantizar que la detección de fraudes se realice de manera ética y respetando la privacidad de los individuos.

6 Bibliografía

References

[1] P. Tan, M. Steinbach, V. Kumar. *Introduction to Data Mining.* Addison-Wesley. 2006.

[2] Edgar Lopez-Rojas· *Synthetic Financial Datasets For Fraud Detection.* Editorial o revista, 2017. https://www.kaggle.com/datasets/ealaxi/paysim1

[3] Ian H. Witten, Eibe Frank, Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (Morgan Kaufmann Series in Data Management Systems)..* 2011.