Universidad Nacional Mayor de San Marcos
Universidad del Perú. Decana de América
Facultad de Ciencias Matemáticas
Escuela Profesional de Computación Científica

Comparación de técnicas de balanceo de datos para la clasificación de fraude en transacciones bancarias

TESIS

Para optar el Título Profesional de Licenciada en Computación Científica

AUTOR

Marjorie Dayanne MONTALVO CUSI

ASESOR

Mg. Luis Javier VÁSQUEZ SERPA

Lima, Perú

2023



Reconocimiento - No Comercial - Compartir Igual - Sin restricciones adicionales

https://creativecommons.org/licenses/by-nc-sa/4.0/

Usted puede distribuir, remezclar, retocar, y crear a partir del documento original de modo no comercial, siempre y cuando se dé crédito al autor del documento y se licencien las nuevas creaciones bajo las mismas condiciones. No se permite aplicar términos legales o medidas tecnológicas que restrinjan legalmente a otros a hacer cualquier cosa que permita esta licencia.

Referencia bibliográfica

Montalvo, M. (2023). *Comparación de técnicas de balanceo de datos para la clasificación de fraude en transacciones bancarias*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ciencias Matemáticas, Escuela Profesional de Computación Científica]. Repositorio institucional Cybertesis UNMSM.

Metadatos complementarios

Datos de autor	
Nombres y apellidos	Marjorie Dayanne Montalvo Cusi
Tipo de documento de identidad	DNI
Número de documento de identidad	72398419
URL de ORCID	
Datos de asesor	
Nombres y apellidos	Luis Javier Vásquez Serpa
Tipo de documento de identidad	DNI
Número de documento de identidad	43389380
URL de ORCID	https://orcid.org/0000-0002-5414-6764
Datos del jurado	
Presidente del jurado	
Nombres y apellidos	Frank Duberlee Alvarez Huertas
Tipo de documento	DNI
Número de documento de identidad	45173557
Miembro del jurado 1	
Nombres y apellidos	Henry Zorrilla Masias
Tipo de documento	DNI
Número de documento de identidad	41190698
Datos de investigación	
Línea de investigación	A.3.4.1. Optimización numérica
Grupo de investigación	No aplica.
Agencia de financiamiento	Sin financiamiento.
Ubicación geográfica de la investigación	Universidad Nacional Mayor de San Marcos y sus coordenadas geográficas País: Perú Departamento: Lima Provincia: Lima

	Distrito: Cercado de Lima Urbanización: (según corresponda) Manzana y lote: (según corresponda) Calle: Av. Carlos Germán Amezaga #375 - Cercado de Lima Latitud: -12.056021389629086 Longitud: -77.08385441690996
Año o rango de años en que se realizó la investigación	Obligatorio. Ejemplo: Setiembre 2022 - octubre 2023 2022 - 2023
URL de disciplinas OCDE	Ciencias de la computación https://purl.org/pe-repo/ocde/ford#1.02.01 Ciencias de la información https://purl.org/pe-repo/ocde/ford#1.02.02

ACTA DE SUSTENTACIÓN DE TESIS PARA LA OBTENCIÓN DEL TÍTULO PROFESIONAL DE LICENCIADO(A) EN COMPUTACIÓN CIENTÍFICA (PROGRAMA DE TITULACIÓN PROFESIONAL 2023)

En la UNMSM – Ciudad Universitaria – Facultad de Ciencias Matemáticas, siendo las 08:00 horas del sábado 28 de octubre del 2023, se reunieron los docentes designados como Miembros del Jurado Evaluador (PROGRAMA DE TITULACIÓN PROFESIONAL 2023): Dr. Frank Duberlee Alvarez Huertas (PRESIDENTE), Mg. Henry Zorrilla Masias (MIEMBRO) y el Mg. Luis Javier Vásquez Serpa (MIEMBRO ASESOR), para la sustentación de la Tesis titulada: "**COMPARACIÓN DE TÉCNICAS DE BALANCEO DE DATOS PARA LA CLASIFICACIÓN DE FRAUDE EN TRANSACCIONES BANCARIAS**", presentado por la señorita **Bachiller MARJORIE DAYANNE MONTALVO CUSI**, para optar el Título Profesional de Licenciada en Computación Científica.

Luego de la exposición de la Tesis, el Presidente invitó a la expositora a dar respuesta a las preguntas formuladas.

Realizada la evaluación correspondiente por los Miembros del Jurado Evaluador, la expositora mereció la aprobación sobresaliente con mención con un calificativo promedio de .diecinueve. (19).

A continuación, los Miembros del Jurado Evaluador dan manifiesto que la participante **Bachiller MARJORIE DAYANNE MONTALVO CUSI**, en vista de haber aprobado la sustentación de su Tesis, será propuesta para que se le otorgue el Título Profesional de Licenciada en Computación Científica.

Siendo las 08:45 horas se levantó la sesión firmando para constancia la presente Acta.

Dr. Frank Duberlee Alvarez Huertas
PRESIDENTE

Mg. Henry Zorrilla Masias
MIEMBRO

Mg. Luis Javier Vásquez Serpa
MIEMBRO ASESOR

CERTIFICADO DE SIMILITUD

Yo …LUIS JAVIER VÁSQUEZ SERPA……. en mi condición de asesor acreditado con la Resolución Decanal N°…001666-2023-D-FCM/UNMSM…….. de la tesis, cuyo título es: "COMPARACIÓN DE TÉCNICAS DE BALANCEO DE DATOS PARA LA CLASIFICACIÓN DE FRAUDE EN TRANSACCIONES BANCARIAS"

presentado por la bachiller …MARJORIE DAYANNE MONTALVO CUSI

para optar el .Título Profesional de Licenciado en Computación Científica de la Facultad de Ciencias Matemáticas,

CERTIFICO que se ha cumplido con lo establecido en la Directiva de Originalidad y de Similitud de Trabajos Académicos, de Investigación y Producción Intelectual. Según la revisión, análisis y evaluación mediante el software de similitud textual, el documento evaluado cuenta con el porcentaje de ..7.% de similitud, nivel **PERMITIDO** para continuar con los trámites correspondientes y para su **publicación en el repositorio institucional.**

Se emite el presente certificado en cumplimiento de lo establecido en las normas vigentes, como uno de los requisitos para la obtención del grado/ título/ especialidad correspondiente.

Firma del Asesor _____



DNI: 43389380

Nombres y apellidos del asesor:

Luis Javier Vásquez Serpa

DEDICATORIA

A mis padres Henry y Guadalupe
por apoyarme siempre.

A mis hermanos, abuelos, tíos y
primos por alentarme.

A mi novio por darme muchas
fuerzas de seguir adelante.

AGRADECIMIENTOS

Agradezco a Dios por guiarme hasta esta etapa de mi vida.
A mi alma mater por haberme formado como profesional.
A mis profesores por todas las enseñanzas compartidas.
A mis amigos de la universidad por todas las experiencias vividas.
A mi familia por nunca dudar de mi.
.

RESUMEN

El presente trabajo de investigación plantea comparar técnicas de balanceo de datos entrenando un algoritmo específico de Machine Learning, a fin de determinar cuál modelo obtiene la mayor puntuación en las métricas de evaluación de modelos de clasificación de fraude en transacciones bancarias

Para llevar a cabo esta evaluación, se implementan tres técnicas de balanceo de datos: Oversampling, Undersampling y SMOTE, en conjunción con el algoritmo de redes neuronales profundas (Deep Neural Network), utilizando datos sintéticos generados por PaySim. Este conjunto de datos simulados emula transacciones bancarias basadas en una muestra de registros financieros existentes extraídos de un mes de operaciones de un servicio de dinero móvil desplegado en un país africano.

Todo este análisis y modelado se realizó haciendo uso del lenguaje de programación de Python ejecutado en Colab PRO. Esta elección del entorno Python permitió la implementación y el entrenamiento de modelos de redes neuronales profundas en un entorno flexible y accesible. El enfoque de utilizar datos sintéticos basados en el mundo real y aplicar técnicas avanzadas de aprendizaje automático subraya la relevancia de abordar el desafío de desequilibrio en conjuntos de datos para mejorar la precisión y confiabilidad de los resultados en aplicaciones financieras y más allá.

Palabras clave: Modelos de aprendizaje automático, Python, técnicas de balanceo de datos, clasificación de fraude, Redes neuronales profundas, PaySim.

ABSTRACT

In this research we propose a comparison between data balancing techniques and a Machine Learning algorithm, to discover which model has the highest score in the evaluation metrics of fraud classification models in banking transactions.

To carry out this evaluation, different data balancing techniques were used, such as Oversampling, Undersampling and SMOTE, together with the Deep Neural Networks (DNN) algorithm, using synthetic data generated by PaySim. This simulated data set emulates banking transactions using a sample of existing financial records taken from operations of a financial service by a mobile during one month. This service was deployed in a country from Africa.

All of this analysis and modeling was done using the Python programming language executed in Colab PRO. This choice of the Python environment allowed the implementation and the construction and training of deep neural network models in a flexible and accessible environment. The approach of using synthetic data based on the real world and applying advanced machine learning techniques underscores the relevance of addressing the challenge of imbalance in data sets to enhance the accuracy and reliability of results in financial and other applications.

Keywords:
Machine learning models, Python, data balancing techniques, fraud classification, Deep Neural Networks, PaySim

Tabla de contenido

1. INTRODUCCIÓN 10
1.1. Introducción 10
1.2. Planteamiento del Problema 12
1.2.1. Formulación del problema 12
1.2.2. Problema general: 12
1.2.3. Problemas específicos 13
1.3. Justificación de la Investigación 13
1.4. Objetivos de la Investigación 14
1.4.1. Objetivo General 14
1.4.2. Objetivos Específicos 14
1.5. Importancia y alcance de la Investigación 14
1.6. Limitaciones de la Investigación 15
2. REVISIÓN DE LITERATURA 16
2.1. Marco Teórico 16
2.1.1. Desbalanceo de data 16
2.1.2. Machine Learning 16
2.1.3. Matriz de confusión 17
2.2. Antecedentes del Problema 17
2.2.1. Antecedentes Locales 18
2.2.2. Antecedentes Nacionales 18
2.2.3. Antecedentes Internacionales 19
2.3. Bases Teóricas 20
2.3.1. Redes Neuronales Profundas 20
2.3.2. Aprendizaje no supervisado 20
2.3.3. Librerías y funciones de Python 21
2.4. Definición de términos 23
2.4.1. Oversampling 23
2.4.2. Undersampling 23
2.4.3. SMOTE (Synthetic Minority Oversampling Technique) 24
2.4.4. Precisión 25
2.4.5. Recall 25
2.4.6. F- Score 25
3. HIPÓTESIS Y VARIABLES 26
3.1. Hipótesis General 26
3.2. Hipótesis Específicas 26
3.3. Variables 26
4. MATERIALES Y MÉTODOS 27
4.1. Área de estudio 27

4.2. Diseño de la investigación 27
4.3. Población y muestra 28
4.4. Procedimientos, Técnicas e instrumentos de recolección de información 28
4.5. Análisis estadístico: 28
5. RESULTADOS 30
5.1. Presentación y análisis de los resultados 30
5.1.1. Análisis de la data: 30
5.1.2. Resultados del modelo de redes neuronales profundas 32
5.1.3. Resultados de métricas de clasificación 40
6. CONCLUSIONES Y RECOMENDACIONES 41
6.1. Conclusiones 41
6.2. Recomendaciones: 42
7. BIBLIOGRAFÍA 43
ANEXOS 48
Anexo 1: Análisis de data 48
Anexo 2: División de data 48
Anexo 3: Modelos de DNN con data desbalanceada 49
Anexo 4: Modelos de DNN con data balanceada 50

ÍNDICE DE TABLAS

Tabla 1: Tabla de variables ……27
Tabla 2: Tabla de campos ……30
Tabla 3: Tabla de resultado de métricas……40

ÍNDICE DE FIGURAS

Figura 1: Matriz de confusión Chawla (2002)……...17
Figura 2: Deep learning in digital pathology. (2018, 13 febrero)……20
Figura 3: How to Use Sklearn train_test_split in Python (2022)…… 22
Figura 4: How to Use Sklearn train_test_split in Python (2022)…… 22
Figura 5: Undersampling and oversampling: an old and a new approach (2022)……..23
Figura 6: Undersampling and oversampling: an old and a new approach (2022)...24
Figura 7: SMOTE(Synthetic Minority Over-sampling Technique) (2023)……..25
Figura 8: Buckland and Gey (1994)……25
Figura 9: Buckland and Gey (1994)……25
Figura 10: Buckland and Gey (1994)……26
Figura 11: Clasificación de fraude……31
Figura 12: Tipos de transacciones en data fraudulenta y no fraudulenta……31
Figura 13: Mapa de calor……..32
Figura 14: Data de entrenamiento……...32
Figura 15: Data de pruebas……..33
Figura 16: Red neuronal profunda (DNN) con keras……33
Figura 17: Entrenamiento de redes neuronales profundas sin balanceo de data utilizando Keras……..34
Figura 18: Resultado de métricas del modelo 1……35
Figura 19: Cantidad de data balanceada con Undersampling……36
Figura 20: Entrenamiento de redes neuronales profundas con keras con data balanceada por oversampling……..36
Figura 21: Resultado de métricas del modelo 2……..37
Figura 22: Cantidad de data balanceada con Oversampling……..37
Figura 23: Entrenamiento de redes neuronales profundas con keras con data balanceada por undersampling……38
Figura 24: Resultado de métricas del modelo 3……38
Figura 25: Cantidad de data balanceada con Smote……39
Figura 26: Entrenamiento de redes neuronales profundas con keras con data balanceada por SMOTE……39
Figura 27: Resultado de métricas del modelo 4……40

1. INTRODUCCIÓN

1.1. Introducción

Actualmente, debido a la transformación digital, muchas organizaciones hacen uso de transacciones financieras por canales digitales.

En 2020 el valor de transacciones financieras on line superó los 38 mil millones de USD en America Latina y El Caribe. Con proyección a los 100 mil millones en el 2025 (Statista Research Department, 13 mar 2023).
Sin embargo, este aumento de transacciones online también ha generado un aumento de delitos cibernéticos. Por ejemplo, en Estados Unidos, en 2019 se presentaron 467,361 denuncias por delitos en internet y en 2020 el número ascendió a 791,790, representando un incremento del 69% respecto del año anterior. Del mismo modo, en 2021 y 2022 se presentaron un total de 847.376 y 800.944 denuncias respectivamente, lo que refleja un aumento del 7% con respecto a 2020. Esto significó que las personas perdieran más de 24.900 millones de dólares por ciber delitos en los últimos 4 años (IC3, 2022).

Existen motivos que generan la desconfianza de los clientes al llevar a cabo transacciones financieras a través de internet. Estos motivos incluyen el temor a que los datos de sus tarjetas de crédito o débito, como el número de tarjeta, el CVV y la fecha de vencimiento, se vean comprometidas. La preocupación radica en que, si estos datos caen en manos de terceros, podrían utilizarse para llevar a cabo transacciones financieras en línea, lo que podría causar perjuicios económicos al usuario. Además, esto también impactaría negativamente en la reputación y prestigio de las instituciones financieras involucradas.

Debido a estas preocupaciones, se ha vuelto crucial para todas las organizaciones, de cualquier tamaño, abordar el desafío de prevenir y detectar el fraude en todas sus formas. Los bancos, empresas de seguridad y usuarios de tarjetas de crédito requieren con gran demanda de modelos estadísticos para la detección de fraudes. (Chan, P. K., Fan, W., Prodromidis, A. L., & Stolfo, S.J., 1999). La preservación de

la confianza del cliente se ha convertido en un objetivo fundamental, dada la importancia de mantener la integridad en las transacciones financieras por internet.

En la actualidad, se observa una tendencia a crear modelos estadísticos orientados a la prevención del fraude mediante la minería de datos. (Deng, X., Josepgh, V. R., Sudjianto, A., & Wu, C. F. J. 2009). Sin embargo, la mayoría de estas investigaciones son realizadas con data no equilibrada.

En el contexto de la clasificación, surgen situaciones en las que la cantidad de ejemplos por clase no está distribuida de manera equitativa. Una de las clases o grupos suele ser dominante, lo que significa que tiene un número mucho mayor de individuos en comparación con la otra clase. Esta gran diferencia en la distribución de la muestra es una situación habitual en diversos contextos, como el fraude financiero o en estudios donde la mayor cantidad de elementos de la muestra no exhiben características particulares.

No obstante, esta disparidad representa un desafío al tratar de identificar las clases minoritarias, ya que muchos algoritmos de aprendizaje no están diseñados para lidiar con este tipo de características. Para abordar este problema, una solución común es aplicar técnicas de balanceo de datos o resampling. Estas técnicas buscan alterar la distribución original de la muestra con el fin de contrarrestar los efectos de la desventaja y mejorar la capacidad del modelo para identificar y aprender de las clases minoritarias.

Esta investigación se compone de siete capítulos. En el primero, se abordan cuestiones relacionadas con el desequilibrio de datos, se plantean los problemas que surgen de esta cuestión, además de proporcionar la justificación, el alcance y la importancia de la investigación.

En el segundo capítulo, se exploran los conceptos y términos que serán utilizados durante la investigación, se exponen los antecedentes tanto a nivel nacional como internacional de tesis previas relacionadas.

En el tercer capítulo, se formulan las hipótesis de investigación, así como también se definen las variables.

En el cuarto capítulo, se detalla la fuente de datos, el diseño de la investigación, la muestra, las técnicas y procedimientos empleados durante su desarrollo.

En el quinto capítulo se presentan los resultados adquiridos durante el análisis y el entrenamiento de los modelos, asimismo, la comparativa de los resultados de las métricas de clasificación.

Finalmente, el último capítulo, se exponen las conclusiones finales y se ofrece futuras recomendaciones.

1.2. Planteamiento del Problema

1.2.1. Formulación del problema

Pulgar et al. (2018), evaluó cómo los datos desequilibrados impactan al desempeño predictivo de las convolutional neural networks o redes neuronales convolucionales (CNN), verificó que, en la mayoría de los casos, el desequilibrio tiene un efecto adverso en el rendimiento predictivo. Al tener data desbalanceada esta investigación tiende a comparar modelos con data balanceada usando oversampling, undersampling y SMOTE para la clasficiación de fraude frente a transacciones bancarias donde entrenaremos un modelo de Machine Learning, Redes neuronales profundas, a fin de comparar la precisión y recuperación y así decidir con cuál modelo se obtiene una mejor puntuación en las métricas de clasificación.

1.2.2. Problema general:

¿Cuál es la técnica de balanceo de datos más adecuada para obtener un modelo clasificación de fraude en transacciones financieras online con el uso de redes neuronales profundas?

1.2.3. Problemas específicos

- ¿Es Oversampling la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas?
- ¿Es Undersampling la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas?
- ¿Es SMOTE la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas?

1.3. Justificación de la Investigación

La presente investigación tiene justificación social ya que actualmente nuestro país viene enfrentando una ola de delincuencia cibernética afectando así a numerosos usuarios que realizan transacciones online.

Esta investigación presenta una justificación metodológica ya que para la presente investigación se emplean distintos procedimientos y técnicas que permitirán obtener y comparar diferentes resultados de modelos empleando machine learning en clasificación de fraude en transacciones financieras.

La presente investigación pretende ser un referente ante la implementación de nuevos modelos de machine learning para la clasificación de datos empleando diversas técnicas de balanceo de datos.

Por lo antes mencionado, en este trabajo se realiza la investigación incentivada por la amplia gama de aplicaciones que se pueden realizar con un modelo óptimo de machine learning empleando data balanceada, dando así un soporte a diferentes campos de investigación.

1.4. Objetivos de la Investigación

1.4.1. Objetivo General

Identificar la técnica de balanceo de datos más adecuada para obtener un modelo con mejor puntuación de métricas de clasificación (*precision* y *recall)* para la clasificación de fraude en transacciones financieras online utilizando el modelo de clasificación supervisada redes neuronales profundas.

1.4.2. Objetivos Específicos

- Determinar si Oversampling es la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas.
- Determinar si Undersampling es la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas.
- Determinar si SMOTE es la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas.

1.5. Importancia y alcance de la Investigación

La razón principal que motivó a realizar la presente investigación es aportar a la investigación en temas de fraude cibernético a nivel nacional y criterios para entrenar modelos de machine learning con data excesivamente desbalanceada en algunos casos incluso con una diferencia de hasta 99% de data mayoritaria contra un 1% de la data minoritaria.

Si solo tuviéramos 1 transacción fraudulenta de un total de 1000, y diseñáramos un modelo estadístico que maximizara la precisión (porcentaje de aciertos totales), este modelo podría predecir todas las transacciones como no fraudulentas. Esto nos daría una precisión del 99.9 %, lo cual podría parecer una métrica bastante impresionante. No obstante, al realizar un análisis más detenido, nos daríamos cuenta de que el modelo no realiza predicciones efectivas. (Alvaro, 2020)

Esto conlleva a un importante investigación teórica y práctica para emplear técnicas de balanceo de datos, modelos de machine learning y métricas de clasificación (*precision y recall*) contribuyendo así con futuras investigaciones científicas, temas académicos y sociales frente a situaciones con data desbalanceada con un enfoque destacado en tecnologías emergentes en el ambito de la detección y prevención de fraude cibernético en transacciones financieras con el uso de machine learning, específicamente aplicando redes neuronales profundas.

1.6. Limitaciones de la Investigación

Fundamentalmente, se presentó limitación en el acceso restringido a datos reales relacionados con cuentas y transacciones de clientes a nivel nacional. Debido a esta limitación, se llevó a cabo la investigación utilizando datos sintéticos generados por PaySim. PaySim es una simulación de transacciones de dinero móvil con el objetivo de generar un conjunto de datos transaccionales sintéticos que se puede utilizar para la investigación sobre la detección de fraude. (Lopez-Rojas et al.,2016).
Este conjunto de datos simulados imita transacciones de dinero móvil y se basa en una muestra extraída de transacciones reales que ocurrieron durante un mes en los registros financieros de un servicio de dinero móvil desplegado en un país africano. Se utilizó esta data sintética debido a que en Perú, la Ley N° 26702 “Ley General del Sistema Financiero y del Sistema de Seguros y orgánica de la Superintendencia de Banca y Seguros” (9 de febrero de 1996) indica la prohibición que tienen las empresas del sector financiero y bancario para suministrar información sensible de operaciones pasivas de sus clientes, con el fin de salvaguardar la seguridad de sus clientes.

2. REVISIÓN DE LITERATURA

2.1. Marco Teórico

2.1.1. Desbalanceo de data

El desbalanceo de clases se refiere a una situación en la que una o varias clases (clases minoritarias) están representadas en la muestra en menor proporción en comparación con otras (clases mayoritarias). (Arnejo, 2017) Esta situación puede manfiestarse en distintos grados, actualmente no hay un punto de referencia establecido que determine cuándo se tiene base de datos desequilibrada.

Específicamente, el desequilibrio de clases puede llevar a un buen rendimiento del modelo en la clase mayoritaria, pero presente dificultades para la clase minoritaria. Esto ocurre porque el modelo detecta una mayor presencia de la primera clase en la muestra, lo que puede afectar el equilibrio a la hora de realizar clasificaciones. El modelo puede sesgarse hacia la clase mayoritaria, lo que implica que tiende a categorizar la mayoría de las observaciones en esa clase, mientras ignora o subestima las clases minoritarias. (Mora, 2023)

La problemática del desequilibrio en la distribución de clases se ha vuelto más prominente con el uso de algoritmos de machine learning en situaciones del mundo real. Abordar este problema es crucial para lograr resultados confiables y precisos en tareas de clasificación, especialmente cuando se trata de clases minoritarias con importancia crítica.

2.1.2. Machine Learning

Tiene el fin de permitir que las máquinas y las computadoras aprendan de manera automática. En otras palabras, busca que las máquinas sean capaces de reconocer patrones complejos dentro de una amplia data, ya sea a través de ejemplos, experiencia o instrucciones predefinidas. La esencia del aprendizaje radica en desarrollar algoritmos que puedan examinar los datos y prever comportamientos futuros, adaptándose al incorporar nueva información y ajustando los resultados en consecuencia.

Los campos fundamentales asociados con el Machine Learning incluyen la Inteligencia Artificial (IA), el procesamiento del lenguaje natural, la minería de datos (DM), las ciencias de la computación, las matemáticas y la estadística. Estos campos trabajan en conjunto para desarrollar algoritmos y metodologías que posibiliten a las máquinas aprender, adaptarse y mejorar su rendimiento en diversas tareas y aplicaciones.

2.1.3. Matriz de confusión

Es una tabla que facilita la representación visual del desempeño de un algoritmo de clasificación. En la Figura 1 según Chawla et al. (2002). En dicha matriz, las columnas representan las clases predichas, mientras que las filas corresponden a las clases reales. En este contexto, TN denota la cantidad de muestras negativas clasificadas de forma correcta (Verdaderos Negativos), FP indica la cantidad de muestras negativas erróneamente clasificadas como positivas (Falsos Positivos), FN se refiere a muestras positivas clasificadas erróneamente como negativas (Falsos Negativos) y TP representa muestras positivas clasificadas de forma correcta (Verdaderos Positivos).

	Predicted Negative	Predicted Positive
Actual Negative	TN	FP
Actual Positive	FN	TP

Figura 1: Matriz de confusión Chawla (2002)

2.2. Antecedentes del Problema

2.2.1. Antecedentes Locales

RANTES, Mónica y CRUZ, Lizbeth (2010). Desarrollaron su tesis de grado en Lima, cuyo principal propósito de esta investigación fue validar que el método de agrupación (clustering) (K-vecinos más cercanos, Ubicación de los mejores centros o clusters de datos) es efectivo para detectar operaciones efectuadas con tarjetas de crédito con un comportamiento anómalo. La investigación fue de tipo aplicada y de nivel evaluativa. El diseño fue no experimental.
La población de objeto de estudio fue operaciones bancarias reales de un banco de Lima, Perú. La muestra fue de 3000 operaciones bancarias.
Los resultados evidenciaron que el modelo propuesto es efectivo para detectar operaciones efectuadas con tarjetas de crédito con un comportamiento anómalo.

AGUILAR, Dennys y CAMARGO, Julio (2021). Realizaron su tesis de grado en Lima. Esta investigación tuvo como objetivo demostrar que un sistema inteligente basado en un modelo que integra Máquinas de Soporte Vectorial (SVM), Redes Neuronales (NN) y Random Forest (RF) desarrollado in house proporciona una mejor precisión. El estudio fue de tipo aplicada y de nivel evaluativa. El diseño fue experimental.
La población de objeto de estudio fueron clientes de un banco de Perú. La muestra fue de 24 420 clientes.

2.2.2. Antecedentes Nacionales

Panta, Juan (2021) con su tesis de grado desarrollada en la ciudad de Piura, examinó la utilización de un modelo de regresión logística en conjunto con técnicas de balanceo de datos para la clasificación de resultados de postulantes a la Facultad de Ciencias de la Universidad Nacional de Piura.
El propósito central del estudio fue comparar el desempeño de modelos de regresión logística aplicados tanto con técnicas de balanceo de datos como sin ellas, con el fin de determinar cuál de estos enfoques proporciona una mejor clasificación. La investigación adoptó un enfoque aplicado y se situó en el ámbito de la evaluación. El diseño utilizado fue experimental.

El grupo de interés en esta investigación lo constituyeron los postulantes del proceso de admisión a la Facultad de Ciencias de la Universidad Nacional de Piura. La muestra estudiada consistió en 184 postulantes.

Para abordar el estudio, se emplearon tres modelos de regresión logística. El primer modelo se aplicó a datos desbalanceados, el segundo modelo se ajustó con la técnica de sobremuestreo (oversampling) para equilibrar los datos, y el tercer modelo utilizó SMOTE. Luego de comparar los 3 modelos señalaron que el modelo que empleó la técnica SMOTE brindó resultados superiores en comparación con los otros modelos en términos de clasificación precisa de los resultados de los postulantes.

2.2.3. Antecedentes Internacionales

González, E. (2018). Realizó su tesis de grado en Bogotá, Colombia. Esta investigación tuvo como objetivo comparar modelos probabilísticos de minería de datos utilizando el lenguaje R en detección de fraude. La investigación fue de tipo aplicada y de nivel evaluativa. El diseño fue no experimental.

La población de objeto de estudio fueron transacciones efectuadas con tarjetas de crédito durante el mes de setiembre de 2013 por titulares de tarjetas de Europa. La muestra fue de 284,807 transacciones.

Los resultados evidenciaron que los modelos de Random Forest, Redes neuronales, y Máquinas de soporte Vectorial son óptimos para la detección de fraude online con tarjetas de crédito.

Wei, W., Li, J., Cao, L., Ou, Y. & Chen, J. (2013). Publicaron un artículo en el World Wide Web.

Este artículo tuvo como objetivo comparar un novedoso algoritmo de Minería de datos (Redes neuronales sensible a costos CNN) para la detección de Fraude con métodos tradicionales de detección de fraude. El estudio fue de tipo aplicada y de nivel evaluativa. El diseño fue no experimental.

La población de objeto de estudio fueron transacciones bancarios en línea de un importante banco australiano. La muestra fue de 8.009.500 transacciones.

Los resultados evidenciaron que el sistema de alerta desarrollado por los autores basado en un algoritmo de Minería de datos (Redes neuronales sensible a costos

CNN) es mejor en eficiencia y precisión que un método tradicional de detección de fraude.

2.3. Bases Teóricas

2.3.1. Redes Neuronales Profundas

Una red neuronal se compone de múltiples capas de nodos, estas reciben entradas de otras capas y generan una salida, avanzando gradualmente hasta obtener un resultado final. Las redes neuronales tradicionales generalmente constan de 2 o 3 capas ocultas, mientras que las Redes de Aprendizaje Profundo pueden llegar a tener hasta 150 capas ocultas. La complejidad de una red neuronal aumenta con el número de capas ocultas que posea. (Mercier, 2022).

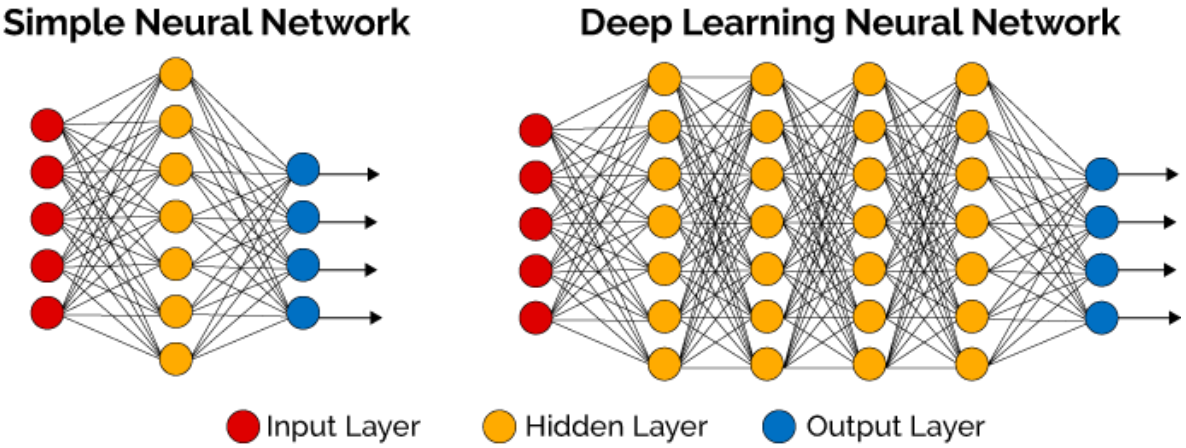


Figura 2: Deep learning in digital pathology. (2018, 13 febrero)

2.3.2. Aprendizaje no supervisado

Este tipo de aprendizaje implica el uso de algoritmos de machine learning para analizar y clasificar datos que carecen de etiquetas. Estos algoritmos tienen la capacidad de identificar conjuntos de datos o patrones subyacentes sin requerir de intervención humana. La habilidad para descubrir semejanzas y diferencias en los datos los hace especialmente adecuados para tareas como la exploración de datos, clasificación de clientes e identificación de imágenes. (IBM, s. f.)

2.3.3. Librerías y funciones de Python

Las bibliotecas o librerías son conjuntos de funciones desarrolladas en un lenguaje de programación específico. Estas bibliotecas tienen una interfaz definida y un propósito claro cuando se llaman.

A menudo, son creadas por profesionales con experiencia en un área particular de tecnología. Estos expertos, debido a su conocimiento especializado, identifican problemas específicos en su campo. Esto les permite optimizar su tiempo de trabajo al centrarse en aspectos críticos. (Galarza, 2023)

- Función train_test_split: Divide la data en dos grupos distintos: un grupo para el entrenamiento del modelo y el otro para las pruebas de rendimiento (entrenamiento y prueba). Esta función está estrechamente relacionada con el concepto de estratificación, que implica dividir los datos de manera que se conserve la proporción de una variable específica, generalmente la variable objetivo. Cuando no se especifica explícitamente la estratificación en "train_test_split", la función intenta realizar la división de manera equilibrada, evitando divisiones extremas como, por ejemplo, un 90% para entrenamiento y un 10% para prueba. Para no depender únicamente de la lógica interna de la función, es común proporcionar el parámetro de estratificación para indicar qué variable debe ser estratificada y en base a cuál variable debe hacerse. De esta manera, la proporción de la variable objetivo se conservará en ambos conjuntos, entrenamiento y prueba.

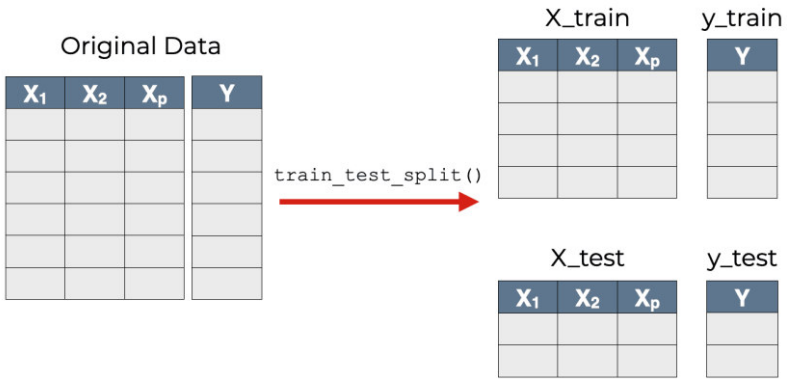


Figura 3: How to Use Sklearn train_test_split in Python (2022)

Ebner (2022), indica que cuando desarrollamos un modelo de aprendizaje automático, generalmente debemos ingresar una agrupación de datos en un algoritmo de aprendizaje automático. Esta agrupación de datos se denomina grupo de entrenamiento, que se usa para entrenar al algoritmo y el grupo de datos de prueba se utiliza después de entrenar el algoritmo para probar que el modelo funciona bien.

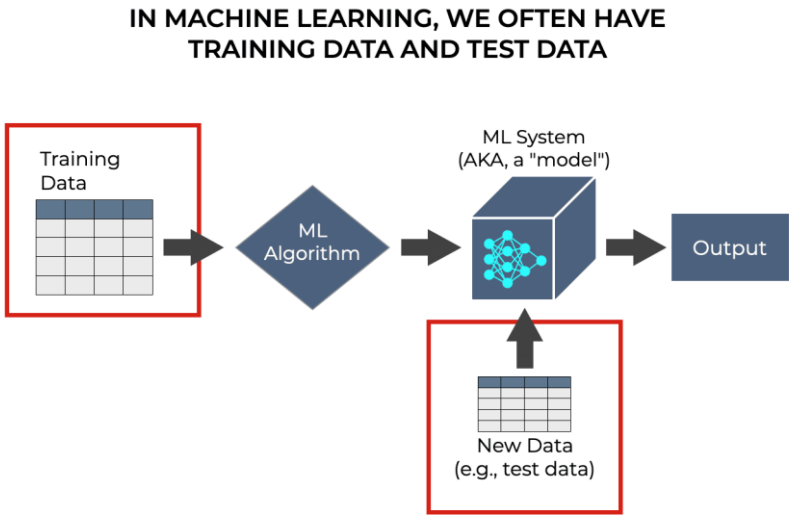


Figura 4: How to Use Sklearn train_test_split in Python (2022)

- Keras es una API utilizada en el desarrollo de redes neuronales, escrita en Python. Keras se ejecuta sobre framework de deep learning TensorFlow. Hoy en día, Keras es una de las APIs más populares para la creación y prueba de redes neuronales. Usando Keras se construye un modelo como una secuencia o un gráfico independiente. Esta API proporciona módulos completamente configurables que pueden ser combinados entre sí para crear nuevos modelos. La modularidad tiene la ventaja de permitir la adición de nuevas funcionalidades en forma de módulos independientes. Esto hace que Keras sea muy flexible para la investigación e innovación. La orientación de Keras hacia la intuición, la modularidad y la facilidad de expansión la hace idónea para trabajar con Python. Según sus creadores, Keras está diseñada para ser más legible por los humanos que por las máquinas, siguiendo las

mejores prácticas para minimizar la carga cognitiva. Keras simplifica la adición de nuevos módulos, clases y funciones, y las plantillas son definidas en el propio código Python, en lugar de requerir archivos de configuración de modelos separados. Alexandre (2023)

2.4. Definición de términos

2.4.1. Oversampling

Consiste en aumentar el número de filas de las otras clases para igualarlas a la clase mayoritaria es un proceso en el que se duplican las muestras de las clases minoritarias o menos representadas. Las desventajas de esta técnica son el tiempo excesivo de procesamiento de datos y al realizar un alto volumen de réplicas de algunas muestras, afectando excesivamente la distribución, llevando a un aumento significativo e incontrolado de las probabilidades de ocurrencia de eventos de baja frecuencia. Al-Serw (2021)

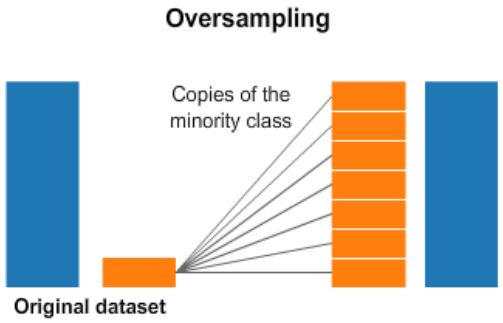


Figura 5: Undersampling and oversampling: an old and a new approach (2022)

2.4.2. Undersampling

Equilibra la distribución de datos mediante la eliminación de instancias de la clase mayoritaria para igualar en cantidad a la clase minoritaria. Por más que sea una técnica sencilla y de rápido procesamiento de datos, es posible eliminar elementos de la muestra que pueden ser críticos para el entrenamiento de la clasificación. Al-Serw (2021)

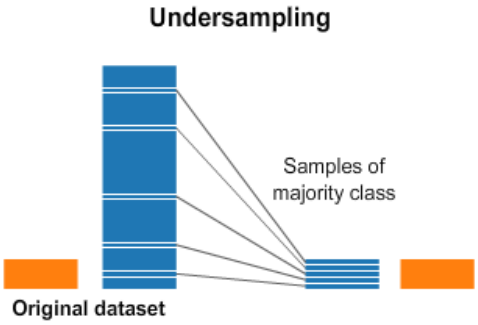


Figura 6: Undersampling and oversampling: an old and a new approach (2022)

2.4.3. SMOTE (Synthetic Minority Oversampling Technique)

La técnica de sobremuestreo de minorías sintéticas, abreviada como SMOTE, fue introducida por Nitesh Chawla y su equipo en su artículo de 2002 llamado " SMOTE: Synthetic Minority Over-sampling Technique.". La forma en que SMOTE opera implica la selección de ejemplos que se encuentran cerca en el espacio de características, trazando una línea entre estos ejemplos y generando una nueva muestra en algún punto a lo largo de esa línea.

El método SMOTE comienza eligiendo al azar un dato perteneciente a la clase minoritaria y determina sus k vecinos más cercanos que también pertenecen a la misma clase minoritaria. Luego, para crear una instancia sintética, selecciona al azar uno de esos k vecinos cercanos, al que llamaremos "b", y conecta los puntos "a" y "b" para formar un segmento de línea en el espacio de características. Los datos sintéticos se generan como una mezcla convexa de los dos datos seleccionados, "a" y "b". (Página 47, Imbalanced Learning, 2013).

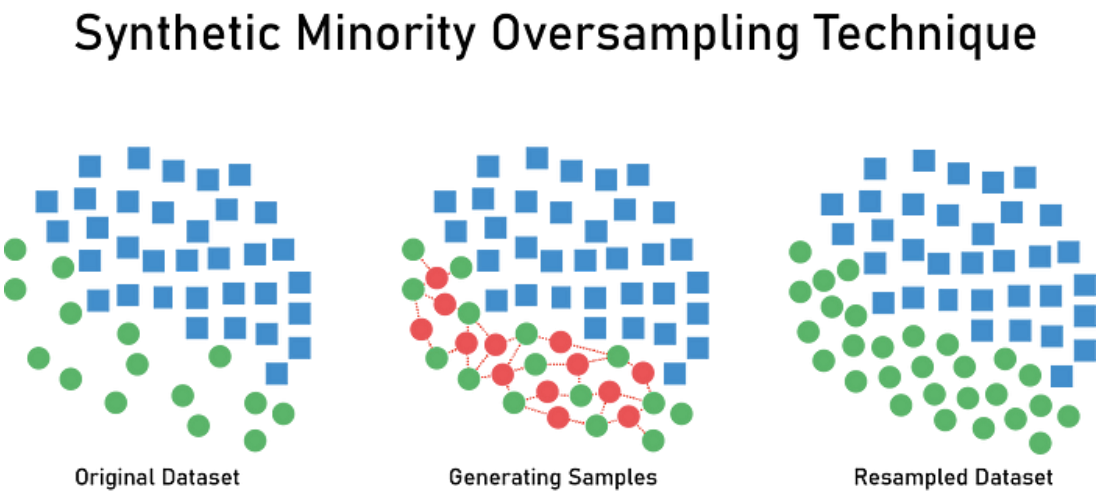


Figura 7: SMOTE(Synthetic Minority Over-sampling Technique) (2023)

2.4.4. Precisión

La precisión busca responder a la pregunta: ¿Qué proporción de las identificaciones positivas fue realizada correctamente?

$$precision = \frac{TP}{TP + FP}$$

Figura 8: Buckland and Gey (1994)

2.4.5. Recall

La recuperación responde a: ¿Qué cantidad de positivos reales fue identificada de manera correcta?

$$recall = \frac{TP}{TP + FN}$$

Figura 9: Buckland and Gey (1994)

2.4.6. F- Score

F es un indicador que acopla las ventajas y desventajas de la precisión y la recuperación, y genera un solo número que refleja la "bondad" de un clasificador en presencia de clases raras. En la figura 9 se observa que es una formula general donde Betha es un factor real positivo, donde considera que la recuperación es Betha veces tan importante como la precisión. El F más usado es F1, tomando Betha el valor de 1.

$$F - value = \frac{(1+\beta^2) * recall * precision}{\beta^2 * recall + precision}$$

Figura 10: Buckland and Gey (1994)

3. HIPÓTESIS Y VARIABLES

3.1. Hipótesis General

El uso de técnicas de balanceo de datos proporcionará mejores resultados para obtener un modelo clasificación de fraude en transacciones financieras con el uso de redes neuronales profundas

3.2. Hipótesis Específicas

- Oversampling es la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas.
- Undersampling es la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas.
- SMOTE es la técnica de balanceo de datos más adecuada para obtener el mejor modelo de clasificación con el uso de redes neuronales profundas.

3.3. Variables

Variables de investigación: Técnicas de balanceo de data

Tabla 1: Tabla de variables

Variables	Definición conceptual	Definición operacional	Dimensiones	Indicadores
Técnicas de balanceo	Técnicas para poder tratar los datos desbalanceados y poder predecir correctamente las clases minoritarias del conjunto de datos	Las técnicas de balanceo de datos equilibran la data de clase minoritaria de la mayoritaria por medio de diferentes técnicas de muestreo.	Oversampling	Precisión
				Recall
				F1
			Undersampling	Precisión
				Recall
				F1
			SMOTE	Precisión
				Recall
				F1

Fuente: Elaboración propia.

4. MATERIALES Y MÉTODOS

4.1. Área de estudio

Enfocada a estudios de clasificación de fraude en transacciones financieras online utilizando técnicas de balanceo de data y modelo de redes neuronales profundas.

4.2. Diseño de la investigación

Esta investigación adopta un enfoque cuantitativo y comparativo al contemplar un análisis en los resultados de puntuaciones de métricas de clasificación entre modelo de redes neuronales profundas sin balanceo de data, redes neuronales profundas con balanceo de data aplicando las técnicas: Oversampling, Undersampling y SMOTE, a fin de determinar el modelo que mejor puntuación tenga en las métricas de clasificación (*precisión*, *recall y f1)*

4.3. Población y muestra

La población está compuesta por datos sintéticos de transacciones financieras generados por Paysim. La muestra, por otro lado, consiste en datos sintéticos de un mes de transacciones financieras basadas en transacciones reales de un banco en África.

4.4. Procedimientos, Técnicas e instrumentos de recolección de información

Se obtiene la data Synthetic Financial Datasets for fraud Detection. (2017) de un repositorio libre online (Kaggle), recuperado de https://www.kaggle.com/datasets/ealaxi/paysim1 en un archivo de formato .csv que contiene la data sintética de transacciones financieras durante un mes, registradas por horas.

Se realiza el análisis de data como: tipos de datos, datos nulos, variables cualitativas y cuantitativas, conteo de datos, clasificación de tipo de datos. Se realizó un código en lenguaje Python para el análisis de data y gráficas, utilizando librerías como: pandas.plotting, numpy, mpl_toolkits.mplot3d, sklearn.preprocessing, matplotlib.pyplot, seaborn. (Ver Anexo 1)

Se agrupan los datos en dos partes distintas: el grupo de entrenamiento y el grupo de prueba, haciendo uso de la función "train_test_split en Python. (Ver Anexo 2).

Se ha desarrollado un código en Python para la implementación de un modelo de Redes Neuronales Profundas (DNN) utilizando la librería Keras. Este código permitirá construir y entrenar una red neuronal profunda para la clasificación de fraude en transacciones bancarias, aprovechando la flexibilidad y facilidad de uso que ofrece Keras. Además de evaluar la puntuación de las métricas de clasificación entre modelos con data balanceada y desbalanceada. (Ver Anexo 3 y 4)

4.5. Análisis estadístico:

Se emplean datos de las diversas columnas para llevar a cabo un estudio exhaustivo de cada una. Se examina tanto la cantidad de datos clasificados como fraude y no fraude. Esto se realiza mediante la representación visual de los datos a

través de gráficos de barras, así como también utilizando mapas de calor para identificar patrones y relaciones.
En el caso del modelo de aprendizaje automático, específicamente las redes neuronales profundas, se realiza el proceso de entrenamiento utilizando la librería Keras. Keras simplifica la construcción del modelo al trabajar con capas< de entrenamiento. Estas capas permiten que la red neuronal aprenda patrones y relaciones en los datos.

Para evaluar la precisión y confiabilidad de los modelos, se recopilan métricas de clasificación relevantes. Estas métricas brindan información sobre el rendimiento del modelo en términos de su capacidad para clasificar correctamente entre las distintas categorías, como fraude y no fraude. Las puntuaciones obtenidas en estas métricas son utilizadas para comparar y determinar cuál de los modelos es más efectivo en la solución del problema.

Todo este proceso de análisis, entrenamiento y evaluación se desarrolla en el lenguaje de programación Python, aprovechando sus librerías y herramientas diseñadas para analizar datos y desarrollo de modelos de aprendizaje automático.

5. RESULTADOS

5.1. Presentación y análisis de los resultados

Se utilizan gráficos y tablas para reportar estos resultados de manera visual y concisa.

5.1.1. Análisis de la data:

Se realiza una matriz de los campos que figura en la data. Teniendo 11 campos

Tabla 2: Tabla de campos

N°	COLUMNA	TIPO DE DATO	DETALLE	EJEMPLO
1	step	int64	Hora correspondiente a cada hora del mes iniciando desde 1 que pertenece a la primera hora del mes y finalizando en 743, la última hora.	1;125;743
2	type	int64	Tipo de transacción PAYMENT: Pagos de servicios TRANSFER: Transferencias a terceros CASH_IN: Ingresos CASH_OUT: Retiros DEBIT: Pagos con tarjeta débito	PAYMENT TRANSFER CASH_IN CASH_OUT DEBIT
3	amount	float64	Monto de transacción	82685.93;114712.48
4	nameOrig	float64	ID del cliente	C589363823;C140404585
5	oldbalanceOrg	float64	Monto incial del cliente	10892
6	newbalanceOrig	float64	Monto después de la transacción	9446.94
7	nameDest	float64	ID del destinatario	M1218519094, C1688019098
8	oldbalanceDest	float64	Monto inicial del destinatario	306239.03; 0,0
9	newbalanceDest	float64	Monto del destinatario después de la transacción	97263.78; 0,0
10	isFraud	int64	Es fraude 0, no es Fraude 1, si es Fraude	0,1
11	isFlaggedFraud	int64	Es alertado como fraude 0, no es Fraude 1, si es Fraude	0,1

Fuente: Elaboración propia.

Se realizó un conteo de la data clasificada, validando que no existan registros de data nula, teniendo un total de 6,362,620 de registros de transacciones realizadas durante un mes, de las cuales se contabilizan que 6,354,407 transacciones No son fraudulentas representando así un 99.87% del total, mientras que se contabilizan que 8213 transacciones son fraudulentas, representando así 0.13% del total de transacciones bancarias registradas durante un mes. Notamos así que tenemos data desbalanceada.

```
0    6354407
1       8213
Name: isFraud, dtype: int64
```

Figura 11: Clasificación de fraude

Se generó un gráfico con el propósito de categorizar los datos en dos grupos: fraudulentos y no fraudulentos. El enfoque se centró en analizar el tipo de transacción, revelando que la mayor parte de las transacciones no fraudulentas corresponden a pagos y retiros en efectivo. Por otro lado, en el caso de las transacciones fraudulentas, la mayoría involucra transferencias y retiros en efectivo. No obstante, es importante destacar que este patrón no necesariamente es un criterio absoluto para identificar casos de fraude.

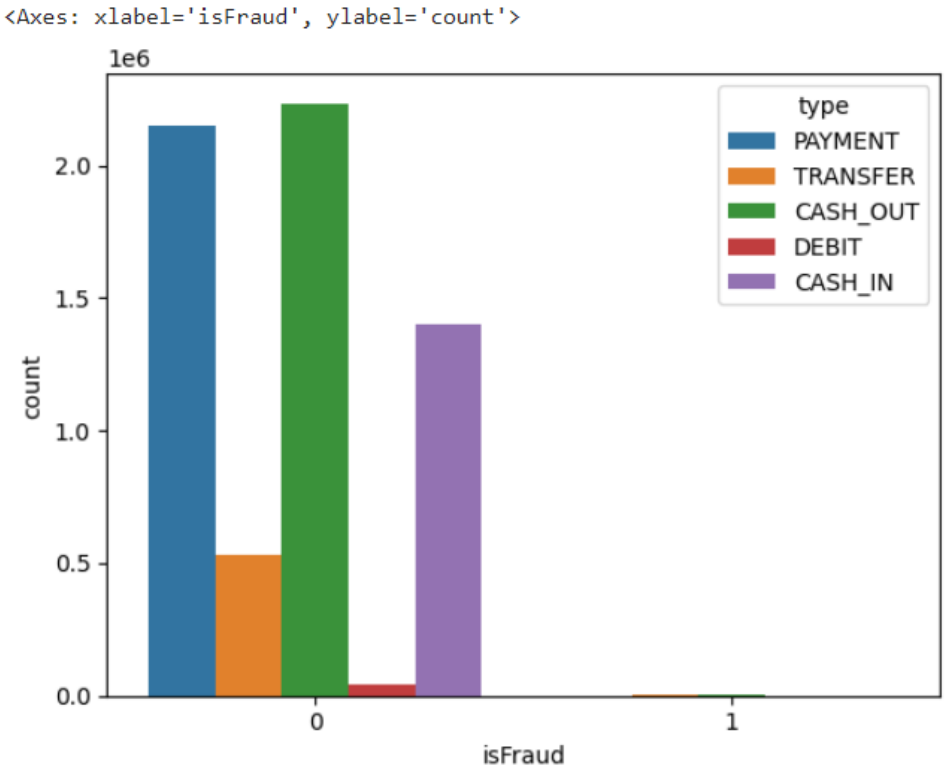


Figura 12: Tipos de transacciones en data fraudulenta y no fraudulenta

Se realiza un mapa de calor donde se aprecia la temperatura de acuerdo con el comportamiento de los campos.

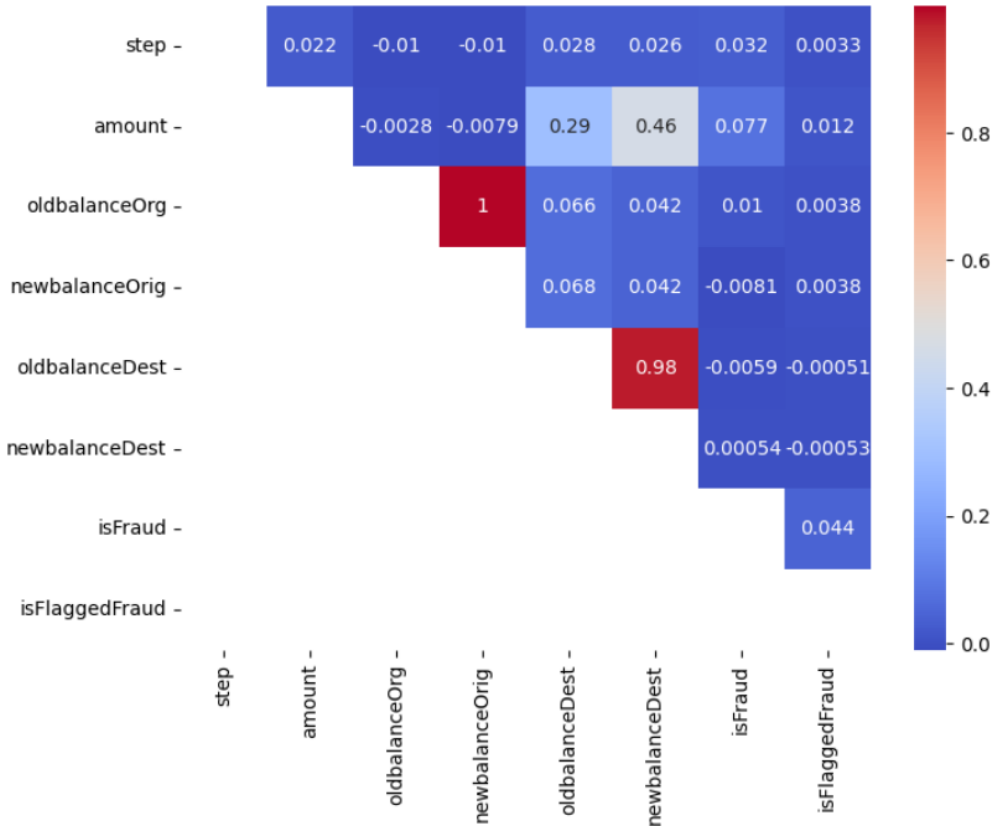


Figura 13: Mapa de calor

5.1.2. Resultados del modelo de redes neuronales profundas

Seleccionamos las columnas de entrada y salida. Las columnas de entrada abarcan todas excepto la columna "IsFraud", las cuales almacenamos en la variable X. La columna de salida es "IsFraud", que se almacena en la variable y. Para dividir la data desequilibrada en grupos de entrenamiento y prueba, aplicamos la función train_test_split. La división se realiza de manera que el 80% de las muestras se destinarán al entrenamiento empleando redes neuronales profundas, mientras que el 20% restante se reservará para realizar pruebas.

```
y_train.value_counts()
```

```
0.0    5083526
1.0       6570
Name: isFraud, dtype: int64
```

Figura 14: Data de entrenamiento

```
y_test.value_counts()
```

```
0.0    1270881
1.0       1643
Name: isFraud, dtype: int64
```

Figura 15: Data de pruebas

Utilizamos Keras para configurar las capas de la red neuronal profunda. Importamos Keras de TensorFlow y empleamos un modelo Sequential de Keras para construir las capas de la red de forma secuencial, una detrás de otra. Inicializaremos la capa de entrada, la capa oculta o intermedia y la capa de salida, que en conjunto formarán el modelo definitivo de la red neuronal profunda. A continuación, describiremos las funciones de cada una de estas capas:

Empezamos con la función *Sequential* para establecer la base del modelo. La red neuronal profunda incorpora varias capas densas en el siguiente orden:

- Primera capa densa: Capa de entrada. Tiene 7 neuronas debido al número de columnas de entrada. Empleamos la función de activación relu debido a que la salida de esta capa debe caer entre 0 y valores positivos infinitos.

- Segunda capa densa: Capa oculta o intermedia. Dispone de 15 neuronas para mejorar aún más el desempeño. Nuevamente, aplicamos la función de activación relu.

- Tercera capa densa: Capa de salida. Incorpora una única neurona que generará los resultados de la red neuronal. En este caso, aplicamos la función de activación sigmoid, adecuada para limitar la salida de la red neuronal dentro del rango de 0 a 1.

```
dnn_model = keras.Sequential([
    keras.layers.Dense(7, input_shape=(7,), activation='relu'),
    keras.layers.Dense(15, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
```

Figura 16: Red neuronal profunda (DNN) con keras

De esta forma, el modelo DNN se compone de estas capas, cada una con su función específica en el proceso de procesamiento y predicción.

5.1.2.1. Modelo 1: Redes neuronales profundas sin balanceo de data

Se realizó el modelo de redes neuronales profundas sin balancear la data solo con la data de entrenamiento y se obtuvo el siguiente resultado:

```
Epoch 1/100
101802/101802 [==============================] - 272s 3ms/step - loss: 5.4719 - accuracy: 0.9974
Epoch 2/100
101802/101802 [==============================] - 270s 3ms/step - loss: 0.6176 - accuracy: 0.9985
Epoch 3/100
101802/101802 [==============================] - 270s 3ms/step - loss: 0.7316 - accuracy: 0.9985
Epoch 4/100
101802/101802 [==============================] - 270s 3ms/step - loss: 0.3337 - accuracy: 0.9985
Epoch 5/100
101802/101802 [==============================] - 270s 3ms/step - loss: 0.3839 - accuracy: 0.9985
Epoch 6/100
101802/101802 [==============================] - 271s 3ms/step - loss: 0.6822 - accuracy: 0.9983
Epoch 7/100
101802/101802 [==============================] - 271s 3ms/step - loss: 0.2783 - accuracy: 0.9987
Epoch 8/100
101802/101802 [==============================] - 271s 3ms/step - loss: 0.9857 - accuracy: 0.9987
Epoch 9/100
101802/101802 [==============================] - 271s 3ms/step - loss: 0.1405 - accuracy: 0.9987
Epoch 10/100
101802/101802 [==============================] - 271s 3ms/step - loss: 0.1728 - accuracy: 0.9987
                                    ⋮
Epoch 90/100
101802/101802 [==============================] - 273s 3ms/step - loss: 0.5104 - accuracy: 0.9987
Epoch 91/100
101802/101802 [==============================] - 273s 3ms/step - loss: 0.4040 - accuracy: 0.9987
Epoch 92/100
101802/101802 [==============================] - 272s 3ms/step - loss: 0.6252 - accuracy: 0.9987
Epoch 93/100
101802/101802 [==============================] - 272s 3ms/step - loss: 0.1668 - accuracy: 0.9987
Epoch 94/100
101802/101802 [==============================] - 272s 3ms/step - loss: 0.2858 - accuracy: 0.9986
Epoch 95/100
101802/101802 [==============================] - 273s 3ms/step - loss: 0.0388 - accuracy: 0.9987
Epoch 96/100
101802/101802 [==============================] - 273s 3ms/step - loss: 0.1131 - accuracy: 0.9987
Epoch 97/100
101802/101802 [==============================] - 272s 3ms/step - loss: 0.0441 - accuracy: 0.9987
Epoch 98/100
101802/101802 [==============================] - 273s 3ms/step - loss: 0.1882 - accuracy: 0.9987
Epoch 99/100
101802/101802 [==============================] - 272s 3ms/step - loss: 0.0693 - accuracy: 0.9987
Epoch 100/100
101802/101802 [==============================] - 272s 3ms/step - loss: 0.0796 - accuracy: 0.9987
<keras.callbacks.History at 0x7c7b2c426f80>
```

Figura 17: Entrenamiento de redes neuronales profundas sin balanceo de data utilizando Keras

La ejecución de este entrenamiento tomó más de 12 horas, utilizando RAM de GPU

ampliada y casi 100 unidades de procesamiento extra que brinda COLAB PRO. La puntuación de exactitud final generada por la red neuronal profunda es 0.9987. Al evaluar el desempeño de un modelo que emplea un conjunto de datos desequilibrado, no es apropiado emplear la puntuación de exactitud, ya que esto podría llevar al modelo a favorecer la clase mayoritaria y predecir dicha clase constantemente, aun cuando la puntuación de exactitud sea alta.

Por lo tanto, optamos por emplear las métricas de recuperación, precisión y puntuación F1 para la evaluación. Estas métricas se enfocan en el desempeño de cada clase de manera individual después de que la red neuronal haya realizado sus predicciones.

Para hacer una predicción utilizamos las muestras de prueba y ejecutamos el reporte de clasificación de métricas

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	1270881
1.0	0.00	0.00	0.00	1643
accuracy			1.00	1272524
macro avg	0.50	0.50	0.50	1272524
weighted avg	1.00	1.00	1.00	1272524

Figura 18: Resultado de métricas del modelo 1

Notamos que la clase 0 tiene valores de puntuación más altos que la clase 1.
Es esencial que las puntuaciones sean equitativas para ambas clases, lo que asegurará la confiabilidad del modelo.

5.1.2.2. Modelo 2: Redes neuronales profundas con undersampling

Para llevar a cabo la técnica de submuestreo utilizaremos el método "sample" de Pandas. El proceso involucra los siguientes pasos:

- Almacenamos las instancias de la clase mayoritaria y minoritaria en variables distintas.

- Concatenamos las muestras submuestreadas de la clase mayoritaria con las muestras de la clase minoritaria. Esto crea un único DataFrame consolidado.
- Imprimimos la cantidad de data equilibrada. Se espera que ambas clases tengan ahora el mismo número de muestras.

De esta manera, conseguimos un conjunto de datos balanceado.

Este enfoque de submuestreo contribuye a reducir el desequilibrio entre las clases y favorece un trato justo para la modelización.

```
Number of data samples after under-sampling:
0    8213
1    8213
Name: isFraud, dtype: int64
```

Figura 19: Cantidad de data balanceada con Undersampling

El código entrenará la red neuronal profunda utilizando el grupo de datos de entrenamiento, tomando 100 capas de entrenamiento. Mostrará el siguiente resultado:

```
Epoch 90/100
411/411 [==============================] - 1s 2ms/step - loss: 14.4377 - accuracy: 0.8967
Epoch 91/100
411/411 [==============================] - 1s 2ms/step - loss: 10.6792 - accuracy: 0.8871
Epoch 92/100
411/411 [==============================] - 1s 2ms/step - loss: 5.7641 - accuracy: 0.8862
Epoch 93/100
411/411 [==============================] - 1s 2ms/step - loss: 4.5747 - accuracy: 0.9066
Epoch 94/100
411/411 [==============================] - 1s 2ms/step - loss: 5.1347 - accuracy: 0.9317
Epoch 95/100
411/411 [==============================] - 1s 2ms/step - loss: 5.5438 - accuracy: 0.9442
Epoch 96/100
411/411 [==============================] - 1s 2ms/step - loss: 4.5618 - accuracy: 0.9514
Epoch 97/100
411/411 [==============================] - 1s 2ms/step - loss: 5.3275 - accuracy: 0.9307
Epoch 98/100
411/411 [==============================] - 1s 2ms/step - loss: 5.7091 - accuracy: 0.9332
Epoch 99/100
411/411 [==============================] - 1s 2ms/step - loss: 3.9055 - accuracy: 0.9222
Epoch 100/100
```

Figura 20: Entrenamiento de redes neuronales profundas con keras con data balanceada por oversampling

La ejecución de este entrenamiento tomó unos miuntos La puntuación de exactitud final generada por la red neuronal profunda es 0.9222.

Sin embargo, debemos aplicar las métricas de clasificación.

	precision	recall	f1-score	support
0	0.87	0.99	0.93	1643
1	0.99	0.85	0.92	1643
accuracy			0.92	3286
macro avg	0.93	0.92	0.92	3286
weighted avg	0.93	0.92	0.92	3286

Figura 21: Resultado de métricas del modelo 2

5.1.2.3. Modelo 3: Redes neuronales profundas con oversampling

La técnica de sobremuestreo implica multiplicar las muestras de datos en la clase minoritaria para lograr un equilibrio. Para llevar a cabo esta técnica, seguiremos los siguientes pasos:

- Utilizamos el método "sample" de Pandas para realizar el sobremuestreo de la clase minoritaria, duplicando sus muestras.
- Después de aplicar el método de sobremuestreo, concatenamos ambas muestras. Esto crea un único DataFrame consolidado con muestras equilibradas de ambas clases.

```
Number of data samples after over-sampling:
1    6354407
0    6354407
Name: isFraud, dtype: int64
```

Figura 22: Cantidad de data balanceada con Oversampling

El código entrenará la red neuronal profunda utilizando el grupo de datos de entrenamiento, tomando 100 capas de entrenamiento mostrará el siguiente resultado:

```
Epoch 89/100
203342/203342 [==============================] - 580s 3ms/step - loss: 0.1069 - accuracy: 0.9670
Epoch 90/100
203342/203342 [==============================] - 578s 3ms/step - loss: 0.1215 - accuracy: 0.9573
Epoch 91/100
203342/203342 [==============================] - 580s 3ms/step - loss: 0.1022 - accuracy: 0.9702
Epoch 92/100
203342/203342 [==============================] - 580s 3ms/step - loss: 0.1264 - accuracy: 0.9526
Epoch 93/100
203342/203342 [==============================] - 579s 3ms/step - loss: 0.1365 - accuracy: 0.9515
Epoch 94/100
203342/203342 [==============================] - 578s 3ms/step - loss: 0.1160 - accuracy: 0.9633
Epoch 95/100
203342/203342 [==============================] - 578s 3ms/step - loss: 0.1171 - accuracy: 0.9634
Epoch 96/100
203342/203342 [==============================] - 578s 3ms/step - loss: 0.1036 - accuracy: 0.9691
Epoch 97/100
203342/203342 [==============================] - 586s 3ms/step - loss: 0.1052 - accuracy: 0.9695
Epoch 98/100
203342/203342 [==============================] - 585s 3ms/step - loss: 0.0863 - accuracy: 0.9796
Epoch 99/100
203342/203342 [==============================] - 590s 3ms/step - loss: 0.1159 - accuracy: 0.9641
Epoch 100/100
203342/203342 [==============================] - 591s 3ms/step - loss: 0.0948 - accuracy: 0.9707
<keras.callbacks.History at 0x7c8cc8cdb310>
```

Figura 23: Entrenamiento de redes neuronales profundas con keras con data balanceada por undersampling

La ejecución de este entrenamiento tomó más de 12 horas, utilizando RAM de GPU ampliada y casi 100 unidades de procesamiento extra que brinda COLAB PRO. La puntuación de exactitud final generada por la red neuronal profunda es 0.9707
Sin embargo, debemos aplicar las métricas de clasificación.

	precision	recall	f1-score	support
0	0.96	1.00	0.98	1270882
1	1.00	0.96	0.98	1270881
accuracy			0.98	2541763
macro avg	0.98	0.98	0.98	2541763
weighted avg	0.98	0.98	0.98	2541763

Figura 24: Resultado de métricas del modelo 3

5.1.2.4. Modelo 4: Redes neuronales profundas con SMOTE

SMOTE genera muestras sintéticas y únicas para la clase minoritaria, con el propósito de lograr un conjunto de datos balanceado. Para implementar esta técnica, seguiremos estos pasos:

- Importaremos la funcionalidad SMOTE desde la librería Imbalanced-learn.

La función SMOTE utiliza el sampling_strategy parámetro para equilibrar las clases. Evidenciamos que los datos están equilibrados:

```
0    6354407
1    6354407
Name: isFraud, dtype: int64
```

Figura 25: Cantidad de data balanceada con Smote

El código entrenará la red neuronal profunda utilizando el grupo de datos de entrenamiento tomando 100 capas de entrenamiento mostrará el siguiente resultado:

```
Epoch 89/100
203342/203342 [==============================] - 553s 3ms/step - loss: 0.0959 - accuracy: 0.9796
Epoch 90/100
203342/203342 [==============================] - 555s 3ms/step - loss: 0.0848 - accuracy: 0.9820
Epoch 91/100
203342/203342 [==============================] - 558s 3ms/step - loss: 0.0942 - accuracy: 0.9824
Epoch 92/100
203342/203342 [==============================] - 558s 3ms/step - loss: 0.1141 - accuracy: 0.9816
Epoch 93/100
203342/203342 [==============================] - 559s 3ms/step - loss: 0.0800 - accuracy: 0.9831
Epoch 94/100
203342/203342 [==============================] - 558s 3ms/step - loss: 0.0935 - accuracy: 0.9800
Epoch 95/100
203342/203342 [==============================] - 562s 3ms/step - loss: 0.0848 - accuracy: 0.9840
Epoch 96/100
203342/203342 [==============================] - 562s 3ms/step - loss: 0.0956 - accuracy: 0.9834
Epoch 97/100
203342/203342 [==============================] - 558s 3ms/step - loss: 0.0912 - accuracy: 0.9789
Epoch 98/100
203342/203342 [==============================] - 552s 3ms/step - loss: 0.0844 - accuracy: 0.9818
Epoch 99/100
203342/203342 [==============================] - 560s 3ms/step - loss: 0.0878 - accuracy: 0.9819
Epoch 100/100
203342/203342 [==============================] - 569s 3ms/step - loss: 0.0849 - accuracy: 0.9830
<keras.callbacks.History at 0x7d08604d5a20>
```

Figura 26: Entrenamiento de redes neuronales profundas con keras con data balanceada por SMOTE

La ejecución de este entrenamiento tomó más de 14 horas, utilizando RAM de GPU ampliada y casi 100 unidades de procesamiento extra que brinda COLAB PRO. La puntuación de exactitud final generada por la red neuronal profunda es 0.9830
Sin embargo, debemos aplicar las métricas de clasificación.

	precision	recall	f1-score	support
0	0.98	1.00	0.99	1270882
1	1.00	0.98	0.99	1270881
accuracy			0.99	2541763
macro avg	0.99	0.99	0.99	2541763
weighted avg	0.99	0.99	0.99	2541763

Figura 27: Resultado de métricas del modelo 4

5.1.3. Resultados de métricas de clasificación

Tenemos un resultado de las puntuaciones en las métricas de clasificación por cada modelo.

Tabla 3: Tabla de resultado de métricas

		DNN SIN BALANCEO DE DATA	DNN CON UNDERSAMPLING	DNN CON OVERSAMPLING	DNN CON SMOTE
PRECISION	0	1.00	0.87	0.96	0.98
	1	0.00	0.99	1.00	1.00
RECALL	0	1.00	0.99	1.00	1.00
	1	0.00	0.85	0.96	0.98
F1	0	1.00	0.93	0.98	0.99
	1	0.00	0.92	0.98	0.99
ACCURACY		0.9987	0.9222	0.9707	0.9830

Fuente: Elaboración propia.

6. CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

Se ha adquirido conocimientos sobre cómo abordar conjuntos de datos desequilibrados en el contexto de redes neuronales profundas. Aplicamos técnicas como submuestreo, sobremuestreo y SMOTE. Iniciamos el proceso al construir un modelo de red neuronal profunda utilizando datos desequilibrados y obtuvimos métricas de rendimiento. Nuestro enfoque se centró en desarrollar un modelo de clasificación para predecir transacciones fraudulentas.

Posteriormente, implementamos las tres técnicas mencionadas para lograr un equilibrio en el conjunto de datos. Evaluamos nuevamente el rendimiento del modelo después de aplicar estas técnicas y observamos mejoras en términos de precisión, recuperación y la puntuación F1 de la red neuronal profunda. Estas estrategias resultaron en un modelo más efectivo para abordar conjuntos de datos con desequilibrio de clases y mejorar su capacidad para predecir si la transacción financiera es fraude o no, de manera más precisa y confiable.

Se concluye observando los resultados que:

El modelo de Redes neuronales profundas sin data balanceada muestra puntuaciones que difieren muy ampliamente en la puntuación de las métricas de clasificación, a pesar de haber obtenido el mayor puntaje en exactitud (accuracy).

El modelo de redes neuronales profundas con undersampling cuya ejecución tomó minutos y no fue necesaria ampliar la RAM ni utilizar unidades de procesamiento extra, muestra puntuaciones que difieren en menor cantidad a comparación del modelo anterior, sin embargo su puntaje de exactitud (accuracy) disminuye ligeramente a comparación del puntaje en el modelo anterior.

El modelo de redes neuronales profundas con oversampling muestra puntuaciones muy cercanas a comparación de los modelos anteriores, sin embargo su puntaje de exactitud (accuracy) es mayor al modelo de DNN con Undersampling sin embargo es menor que el modelo de DNN sin balanceo de data.

Finalmente, el modelo de redes neuronales profundas balanceada con SMOTE muestra puntuaciones bastante cercanas y hasta iguales a persar de que su puntaje de exactitud (accuracy) no supera al modelo de DNN sin data balanceada, lo que

logra un equilibrio más pronunciado entre las clases y, por ende, una mayor fiabilidad en sus resultados.

Estos hallazgos subrayan la importancia de considerar y abordar el desequilibrio de clases al entrenar modelos de aprendizaje automático, ya que dicho equilibrio impacta en la capacidad del modelo para predecir de manera precisa y confiable en escenarios del mundo real.

6.2. Recomendaciones:

- Se sugiere poner a prueba el código (Ver anexo 2 y 3) utilizando conjuntos de datos de transacciones reales para validar su rendimiento en escenarios más cercanos a la realidad.

- Aplicar las técnicas de balanceo a otros modelos de machine learning y comparar las métricas de clasificación obtenidas de estos diferentes modelos puede brindar una comprensión más completa de cuál enfoque es más efectivo para resolver el problema en cuestión. Esta comparación ayudaría a seleccionar el modelo más adecuado en función de su rendimiento en diferentes aspectos, como precisión, recuperación, puntajes F1, etc. Diversificar los enfoques y probar diferentes modelos contribuirá a una evaluación más sólida al momento de elegir el modelo de machine learning que mejor se ajuste a las necesidades y requisitos específicos del problema.

- Implementar una interfaz para mejor visualización y comparación de resultados entre las diferentes técnicas de balanceo de datos utilizando distintos modelos de machine learning.

.

7. BIBLIOGRAFÍA

Articulos cientificos:

1. SANTAMARIA, Wilfredy. 2006 Técnicas de Minería de Datos Aplicadas en la Detección de Fraude. Disponible en: https://www.researchgate.net/publication/240724702_Tecnicas_de_Mineria_de_Datos_Aplicadas_en_la_Deteccion_de_FraudeEstado_del_Arte
2. Pulgar, Francisco; Rivera, Antonio; Charte, Francisco y Del Jesús, María, 2018, Análisis del impacto de datos desbalanceados en el rendimiento predictivo de redes neuronales convolucionales. Disponible en: https://simidat.ujaen.es/sites/default/files/biblio/2018-CAEPIA-DesbalanceoCNNs.pdf

Articulos cientificos en ingles:

3. Chan, P.K., Fan, W., Prodromidis, A.L. and Stolfo, S.J., 1999. Distributed Data Mining in Credit Card Fraud Detection. IEEE Intelligent Systems and Their Applications, 14, 67-74. Disponible en: https://doi.org/10.1109/5254.809570

4. LUENGO, Julián; FERNÁNDEZ, Alberto; GARCÍA Salvador y HERRERA Francisco , España, 2010 Addressing data complexity for imbalanced data sets: analysis of SMOTE-based oversampling and evolutionary undersampling Springer

5. H. He and E. A. Garcia, "Learning from Imbalanced Data," in IEEE Transactions on Knowledge and Data Engineering, vol. 21, no. 9, pp. 1263-1284, Sept. 2009, doi: 10.1109/TKDE.2008.239.

6. Chawla, Bowyer, Hall, & Kegelmeyer, 2002, “SMOTE: Synthetic Minority Over-sampling Technique” in Journal of Artificial Intelligence Research 16 (2002)

7. Wei, W., Li, J., Cao, L., Ou, Y. & Chen, J. (2013). Effective detection of sophisticated online banking fraud on extremely imbalanced data.. World Wide Web, 16, 449-475.
8. Lopez-Rojas, Edgar Alonso & Elmir, Ahmad & Axelsson, Stefan. (2016). PAYSIM: A FINANCIAL MOBILE MONEY SIMULATOR FOR FRAUD DETECTION.

Leyes y Normas:

9. Ley N° 26702 “Ley General del Sistema Financiero y del Sistema de Seguros y orgánica de la Superintendencia de Banca y Seguros” (9 de febrero de 1996) Disponble en: https://www2.congreso.gob.pe/sicr/cendocbib/con4_uibd.nsf/7B3154074498CD5E05257F030072F042/$FILE/26702.pdf

Tesis:

10. GARCÍA, Joaquín. 2021 Comparativa de técnicas de Balanceo de Datos: Aplicación a un caso real para la predicción de fuga de clientes. Trabajo de fin de Máster en análisis de datos para la inteligencia de negocios. Universidad de Oviedo. España. Disponibile en: https://digibuo.uniovi.es/dspace/bitstream/handle/10651/60629/TFM_Joaqu%C3%ADnGarc%C3%ADaAbad.pdf?sequence=4

11. SALAMANCA, Javier y VELÁSQUEZ, Paulina. 2018 MACHINE LEARNING APLICADO A DATOS FINANCIEROS. Informe de avance del proyecto para optar al título profesional de Ingeniero de ejecución en informática. Pontificie Universidad Católoca de Valparíso. Chile. Disponible en: http://opac.pucv.cl/pucv_txt/txt-7500/UCC7987_01.pdf

12. RANTES, Mónica y CRUZ, Lizbeth. 2010. Detección de fraudes usando técnicas de CLUSTERING. Tesis para optar el Título Profesional de Ingeniero de Sistemas. Universidad Nacional Mayor de San Marcos. Lima.

Disponible en: https://cybertesis.unmsm.edu.pe/bitstream/handle/20.500.12672/2644/Rantes_gm.pdf?sequence=1&isAllowed=y

13. AGUILAR, Dennys y CAMARGO, Julio (2021). Sistema inteligente basado en redes neuronales, máquina de soporte vectorial y random forest para la predicción de deserción de clientes en microcréditos de bancos. Tesis para optar el Título Profesional de Ingeniero de Sistemas. Universidad Nacional Mayor de San Marcos. Lima. Disponible en: https://hdl.handle.net/20.500.12672/16390

14. Panta, Juan (2021). Modelo de regresión logística aplicado con técnicas de balanceo de datos para clasificar el ingreso de los postulantes a la Facultad de Ciencias de la Universidad Nacional de Piura. Tesis para optar el Título Profesional de Licenciado en Estadística. Universidad Nacional de Piura. Facultad de Ciencias, Piura Disponible en: http://repositorio.unp.edu.pe/handle/20.500.12676/2689

15. CARMONA, Maricela y LONDOÑO, Leydi. 2021 Modelos de machine learning para la detección de fraude financiero. Tesis o trabajo de investigación presentada(o) como requisito parcial para optar al título de: Especialista en Analítica y Ciencia de Datos. Universidad de Antioquia. Colombia. Disponible en: https://bibliotecadigital.udea.edu.co/bitstream/10495/20164/1/CarmonaMaricela_2021_DeteccionFraudeFinanciero.pdf

16. González, E. (2018). Detección de fraude en tarjetas de crédito mediante técnicas de minería de datos. (Trabajo de pregrado). Universidad Santo Tomás. Bogotá, Colombia. Disponible en: http://repository.usta.edu.co/handle/11634/12529

Libros:

17. MEHLIG, Bernhard. 2021 Machine Learning with Neural Networks Universidad de Cambridge ISBN: 9781108860604

18. VASILEV, Ivan. 2019 Python Deep Learning second edition ISBN 978-1-78934-846-0

Paginas web institucionales:

19. Statista. 2023 *Los pagos móviles en América Latina*. Disponible en: https://es.statista.com/temas/9208/los-pagos-moviles-en-america-latina/#topicOverview

20. Internet Crime Complaint Center(IC3) .2023 Denuncias por delitos en internet Disponible en: https://www.ic3.gov/

21. ¿Qué es el aprendizaje no supervisado? | IBM. (s. f.-b). Disponible en: https://www.ibm.com/es-es/topics/unsupervised-learning

22. Ebner, J. (2022, mayo 16). How to use Sklearn Train_Test_Split in Python - Sharp Sight. Disponible en: https://www.sharpsightlabs.com/blog/scikit-train_test_split/

23. IBM. (s. f.). ¿Qué es el aprendizaje no supervisado? Disponible en: https://www.ibm.com/es-es/topics/unsupervised-learning

24. Alexandre. (2023, 31 enero). Keras: Todo sobre la API de deep learning. Formation Data Science Disponible en: https://datascientest.com/es/keras-la-api-de-deep-learning

25. Al-Serw, N. A. (2021, 31 diciembre). Undersampling and oversampling: an old and a new approach. Disponible en: https://medium.com/analytics-vidhya/undersampling-and-oversampling-an-old-and-a-new-approach-4f984a0e8392

26. Dholakiya, P. (2023, 26 abril). SMOTE(Synthetic Minority Over-sampling Technique) Disponible en: https://medium.com/@parthdholakiya180/smote-synthetic-minority-over-sampling-technique-4d5a5d69d720
27. Alvaro. (2020, 19 enero). Tratamiento de clases desbalanceadas. Disponible en: https://machinelearningparatodos.com/tratamiento-de-clases-desbalanceadas/
28. Mora, M. (2023, 8 agosto). Validez y evaluación del modelo en análisis discriminante - Potenciando la precisión de tus decisiones empresariales. Disponible en: https://es.linkedin.com/pulse/validez-y-evaluaci%C3%B3n-del-modelo-en-an%C3%A1lisis-la-de-mora-caballero?trk=articles_directory#:~:text=En%20un%20conjunto%20de%20datos,o%20subestimando%20las%20clases%20minoritarias.
29. BotPress, (s.f.) ¿Qué es una red neuronal profunda?. Disponible en: https://botpress.com/es/blog/deep-neural-network#:~:text=Una%20red%20neuronal%20profunda%20(Deep,utilizando%20t%C3%A9cnicas%20de%20programaci%C3%B3n%20convencionales.
30. IntelDig. (2019, 25 abril). Redes neuronales profundas - tipos y características. Disponible en: https://www.codigofuente.org/redes-neuronales-profundas-tipos-caracteristicas/
31. *Deep learning in digital pathology*. (2018, 13 febrero). Disponible en: https://www.global-engage.com/life-science/deep-learning-in-digital-pathology/

ANEXOS

Anexo 1: Análisis de data

```
#importing libraries
import pandas as pd
from pandas.plotting import scatter_matrix
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt # plotting
import seaborn as sns
import os # accessing directory structure
```

```
print('Data does not have any NULL value.')
data.isnull().any()
sns.countplot(data, x="isFraud", hue="type")
```

```
color = sns.color_palette("Paired")
mask = np.tril(np.ones_like(data.corr(), dtype=bool))
plt.figure(figsize=(8, 6))
sns.heatmap(data.corr(), mask=mask, cmap= 'coolwarm', annot=True)
```

Anexo 2: División de data

```
import pandas as pd
import numpy as np
```

```
from sklearn.preprocessing import LabelEncoder

def label_encoder(column):
    le = LabelEncoder().fit(column)
    print(column.name, le.classes_)
    return le.transform(column)
```

```
df["type"] = label_encoder(df["type"])
```

```
type ['CASH_IN' 'CASH_OUT' 'DEBIT' 'PAYMENT' 'TRANSFER']
```

```
df=df.drop(columns=['nameOrig','nameDest','isFlaggedFraud'])
```

```
df.isFraud.value_counts()
```

```
0    6354407
1       8213
Name: isFraud, dtype: int64
```

```
X = df.drop('isFraud',axis='columns')
y = testLabels = df.isFraud.astype(np.float32)
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=15, stratify=y)
```

```
y_train.value_counts()
```

```
0.0    5083526
1.0       6570
Name: isFraud, dtype: int64
```

```
y_test.value_counts()
```

```
0.0    1270881
1.0       1643
Name: isFraud, dtype: int64
```

Anexo 3: Modelos de DNN con data desbalanceada

```
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import confusion_matrix , classification_report
```

```
dnn_model = keras.Sequential([
    keras.layers.Dense(7, input_shape=(7,), activation='relu'),
    keras.layers.Dense(15, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

dnn_model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
dnn_model.fit(X_train, y_train, epochs=100, batch_size=50)
dnn_pred = dnn_model.predict(X_test)

dnn_pred = np.round(dnn_pred)
```

```
39767/39767 [==============================] - 45s 1ms/step
```

```
from sklearn.metrics import confusion_matrix , classification_report

print(classification_report(y_test,dnn_pred))
```

Anexo 4: Modelos de DNN con data balanceada

- Metodo #2: DNN with Undersampling

```
# Class count
count_majority_class, count_minoriy_class = df.isFraud.value_counts()


df_majority_class = df[df['isFraud'] == 0]
df_minority_class = df[df['isFraud'] == 1]
```

```
# Undersample 0-class and concat the DataFrames of both class
df_class_undesample = df_majority_class.sample(count_minoriy_class)

df_balanced = pd.concat([df_class_undesample, df_minority_class], axis=0)

print('Number of data samples after under-sampling:')
print(df_balanced.isFraud.value_counts())
```

```
Number of data samples after under-sampling:
0    8213
1    8213
Name: isFraud, dtype: int64
```

```
X = df_balanced.drop('isFraud',axis='columns')
y = df_balanced['isFraud']


X_train_bus, X_test_bus, y_train_bus, y_test_bus = train_test_split(X, y, test_size=0.2, random_state=15, stratify=y)
```

```
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import confusion_matrix , classification_report
```

```
dnn_model = keras.Sequential([
    keras.layers.Dense(7, input_shape=(7,), activation='relu'),
    keras.layers.Dense(15, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])

dnn_model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
dnn_model.fit(X_train_bus, y_train_bus, epochs=100)
```

```
dnn_preds_bus = dnn_model.predict(X_test_bus)
dnn_preds_bus = np.round(dnn_preds_bus)
```

```
103/103 [==============================] - 0s 1ms/step
```

```
print(classification_report(y_test_bus,dnn_preds_bus))
```

- Metodo #3: DNN with Oversampling

```
# Oversample 1-class and concat the DataFrames of both classes
df_class_oversample = df_minority_class.sample(count_majority_class, replace=True)

df_balanced_os = pd.concat([df_class_oversample, df_majority_class], axis=0)

print('Number of data samples after over-sampling:')
print(df_balanced_os.isFraud.value_counts())
```

```
Number of data samples after over-sampling:
1    6354407
0    6354407
Name: isFraud, dtype: int64
```

```
X = df_balanced_os.drop('isFraud',axis='columns')
y = df_balanced_os['isFraud']


X_train_bos, X_test_bos, y_train_bos, y_test_bos = train_test_split(X, y, test_size=0.2, random_state=15, stratify=y)
```

```
dnn_model.fit(X_train_bos, y_train_bos, epochs=100, batch_size=50)
```

```
dnn_preds_bos = dnn_model.predict(X_test_bos)
dnn_preds_bos = np.round(dnn_preds_bos)
```

```
79431/79431 [==============================] - 105s 1ms/step
```

```
print(classification_report(y_test_bos,dnn_preds_bos))
```

- Método #4: DNN with SMOTE

```
! pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.10.1)
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.2.2)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn) (3.2.0)
```

```
from imblearn.over_sampling import SMOTE
```

```
X = df.drop('isFraud',axis='columns')
y = df['isFraud']
```

```
smote_technique = SMOTE(sampling_strategy='minority')
X_smt, y_smt = smote_technique.fit_resample(X, y)

y_smt.value_counts()
```

```
0    6354407
1    6354407
Name: isFraud, dtype: int64
```

```
X_train_smt, X_test_smt, y_train_smt, y_test_smt = train_test_split(X_smt, y_smt, test_size=0.2, random_state=15, stratify=y_smt)
```

```
dnn_model.fit(X_train_smt, y_train_smt, epochs=100, batch_size=50)
```

```
dnn_preds_smt = dnn_model.predict(X_test_smt)
dnn_preds_smt = np.round(dnn_preds_smt)
```

```
79431/79431 [==============================] - 96s 1ms/step
```

```
print(classification_report(y_test_smt,dnn_preds_smt))
```