



Universidad de las Ciencias Informáticas

FACULTAD 2

HERRAMIENTA PARA LA DETECCIÓN DE ANOMALÍAS EN PROBLEMAS DE OTORGAMIENTO DE CRÉDITOS BANCARIOS BASADOS EN BIG DATA

Trabajo Diploma para Ingeniero en Ciencias Informáticas

Autor: Odeynis Valdés Suárez

Tutor: Dr. C. Héctor Raúl González Diez

1 de septiembre de 2021, La Habana, Cuba

Año 63 de la Revolución

Resumen

El contenido de este trabajo está centrado en la demostración de la superioridad que existe entre los modelos de aprendizaje profundo con respecto a los modelos de aprendizaje automático dentro de la inteligencia artificial. Para ello se realiza una fundamentación teórica previa de los conceptos básicos relacionados con este campo de estudio, ya sean

y herramientas utilizadas para realizar las experimentaciones. Se presentarán las estructuras de los modelos para realizar las demostraciones, además de seleccionar los mejores modelos para el desarrollo de la herramienta para la detección de fraude de tarjeta de crédito.

Palabras clave: **modelos de aprendizaje profundo, modelos de aprendizaje automático, detección de fraude de tarjeta de crédito, inteligencia artificial.**

Abstract

The content of this work is focused on the demonstration of the superiority that exists between deep learning models and machine learning models within artificial intelligence. For this, a preliminary foundation of the basic concepts and tools used to carry out the experiments is carried out. The structures of the models will be presented to carry out the demonstrations, in addition to selecting the best models for the development of tools for detecting credit card fraud.

Keywords: **deep learning models, machine learning models, credit card fraud detection, artificial intelligence**

Contenido

1 Fundamentación teórica	8
2 Preparación de la minería de datos	19
2.1 Paso 1: Entendimiento, conocimientos previos e identificación de la meta	19
2.2 Paso 2: Selección del conjunto de datos	20
2.3 Paso 3: Limpieza y preprocesamiento	21
2.3.1 RandomUnderSampler	21
2.3.2 RandomOverSampler	21
2.3.3 SMOTE y ADASYN	22
2.3.4 Paso 4: Transformación de los datos	22
2.4 Paso 5: Método de minería de datos	24
2.5 Paso 6 y 7: Elección e implementación de los algoritmos de minería de datos . .	24
3 Evaluación y aplicación de los modelos DL	38
3.1 Experimento 1: Comparación entre la selección de datos para las pruebas	38
3.2 Experimento 2: Comparación entre los modelos de ML por estrategias para solucionar al desbalance de la información	39
3.3 Experimento 3: Comparación en el orden de los <i>dropout</i> con datos desbalanceados	41
3.4 Experimento 4: Comparación en los modelos DL teniendo en cuenta los resultados de las pruebas.	43
3.4.1 Primera parte: Modelos por epochs con la misma estrategia.	43
3.4.2 Análisis de la primera parte	48
3.4.3 Segunda parte: Estrategias y epochs por modelos	49
3.4.4 Análisis de la segunda parte	51
3.4.5 Análisis general del experimento 4	51

3.5	Experimento 5: Comparación de los modelos DL teniendo en cuenta los resultados del entrenamiento	52
3.5.1	Primera parte: Modelos por epochs con la misma estrategia	52
3.5.2	Análisis de la primera parte	57
3.5.3	Segunda parte: Estrategias y epochs por modelos	57
3.5.4	Análisis de la segunda parte	62
3.5.5	Análisis general del experimento 5	63
3.6	Análisis de los resultados de los experimentos 4 y 5	63
3.7	Experimento 6: Comparación entre los mejores modelos ML y DL, en cuanto a los resultados de las pruebas	64
3.8	Experimento 7: Comparación de todos los modelos ML y DL en el resultado de las pruebas	65
3.9	Resultados de la evaluación	69
3.10	Parte 9: Aplicación del conocimiento adquirido	71
3.10.1	Componentes de las soluciones al desbalance	71
3.10.2	Componentes de los modelos ML y DL	73
3.10.3	Integración de los componentes	73

Introducción

Las Tecnologías de la Información y las Comunicaciones (TICs) han penetrado en diversos sectores de la sociedad como la gestión de los activos, bienes y servicios que posee el ser humano. El modelo tradicional de los bancos ha tenido que cambiar para satisfacer las necesidades y exigencias de la creciente demanda de los clientes, tanto que el cliente puede saber su saldo disponible en una cuenta de banco, transferir dinero y realizar conversiones de monedas sin necesidad de realizar visitas físicas a los bancos, todo ello al alcance de un teléfono móvil conectado a internet. Para mantener el equilibrio entre la emisión de préstamos y el dinero en cuentas de sus clientes, han diversificado su negocio a través de actividades debido a los riesgos financieros de estar apalancados en sus balances ya que su negocio principal se centra en la captación de dinero o pasivo y el préstamo de ese dinero a clientes a un tipo de interés mayor.

Los bancos son un eslabón importante para la economía de un país al igual que para la población que recibe sus servicios. Las empresas emprendedoras pueden fortalecerse a través de préstamos concedidos por el banco, la población puede tener ahorros y un depósito de capital seguro, y con este capital el banco puede impulsar otros negocios. Pueden prestar dinero a personas para la reconstrucción de hogares, adquisición de equipos electrodomésticos, pago de facturas atrasadas, aunque esto pueda ser un arma de doble filo, es una opción importante para muchos. Mediante la revisión de las transacciones que se realizan en sus cuentas de créditos y débitos pueden detectar los fraudes y lavado de dinero.

El fraude de tarjetas crédito y débito encabeza la lista de fraudes bancarios por las diversas formas secretas que encuentran los criminales para acceder a la información de las tarjetas de crédito. Este tipo de fraude es detectado con métodos de inteligencia artificial como la detección de anomalías supervisada y técnicas de regresión.

La detección de anomalías ha cobrado una gran importancia en la comunidad científica actual, buscando siempre métodos para aplicaciones de *Deep Learning* (DL). Existen varias categorías de técnicas de detección de anomalías para *Machine Learning* (ML) y dependen del conjunto de datos de entrenamiento que se proveen, este último clasificado en supervisados, semi-supervisados y no supervisados. Debido a que el conjunto de datos presenta una gran desproporción de las muestras, se presenta el problema desbalanceado donde un tipo de datos del conjunto es considerablemente inferior a la otra. Para el problema desbalanceado se ha adoptado el sobre muestreo de la clase minoritaria para aliviar el problema, pero aún presenta algunas desventajas, una de las cuales es que no añade contenido informativo, limitando el mejoramiento de la habilidad de un clasificador para generalizar. Además, existe el aprendizaje federado que es una tecnología básica de inteligencia artificial que como objetivo tiene asegurar la seguridad de la información durante el intercambio de Big Data, incluyendo la información personal de los clientes.

Con el crecimiento de internet en Cuba y el uso del comercio electrónico por medio de las plataformas Enzona¹ y Transfermóvil² se inician nuevos caminos hacia la transformación digital en Cuba. Sin embargo, este tipo de facilidades requieren de mecanismos de seguridad para en caso de sustracción de la tarjeta o sus datos, garanticen la seguridad de las cuentas bancarias.

Un sistema de seguridad para este tipo de compras puede ser la detección de fraude mediante

¹Enzona es la plataforma cubana para el comercio y el gobierno electrónico, la gestión productiva, los servicios y la calidad de vida del pueblo.

²Transfermóvil es la aplicación Android usada por los clientes de ETECSA para facilitar pagos de servicios, compras en línea, consultas y trámites bancarios y la gestión de los servicios de telecomunicaciones.

el uso de algoritmos de aprendizaje automático, que se encargan de detectar si un tipo de movimiento realizado por la tarjeta se corresponde con un comportamiento normal de esa tarjeta concreta o no, en cuyo caso se deben tomar las medidas pertinentes. Una de las grandes ventajas de usar un algoritmo de aprendizaje es que cuantos más movimientos se realicen con la tarjeta, más precisión tendrá el algoritmo a la hora de determinar si el movimiento que se está realizando actualmente con la tarjeta es fraude o no.

Desde el punto de vista de la modelación del problema para la detección del fraude en operaciones bancarias mediante aprendizaje automático, existen varios problemas computacionales a tener en cuenta:

1. El volumen de operaciones que ocurren diariamente es excesivamente grande para que sea procesado por una computadora por lo que el problema debe ser tratado en entornos distribuidos.
2. El número de operaciones fraudulentas es muchísimo menor (aproximadamente 99% vs 1%) que las transacciones normales por lo que hay un desbalance del problema.
3. El flujo de operaciones que ocurren por unidad de tiempo es elevado.

Se define como problema de investigación ¿cómo mejorar la detección automatizada de fraude bancario en escenarios desbalanceados de *Big Data*?

Se presenta como objetivo general: desarrollar una herramienta que integre algoritmos para la detección automática de fraude bancario basado en detección de anomalías en escenarios de *Big data*. Desglosado en los siguientes objetivos específicos:

1. Caracterizar el marco teórico-conceptual del problema de la detección de fraudes bancarios, los enfoques basados en detección de anomalías, sustentados en escenarios de *Big Data*.
2. Desarrollar algoritmos basado en redes neuronales profundas, computación distribuida y tome en cuenta el problema de desbalance para la detección de anomalías en operaciones bancarias.
3. Validar la solución implementada mediante el diseño de experimentos sobre conjuntos de datos de referencia, comparando los resultados con otros algoritmos del estado del arte.

Las tareas de investigación que guiarán la investigación son:

- Identificar, conceptualizar y caracterizar la detección de fraudes mediante el enfoque de la detección de anomalías.
- Identificar y caracterizar los algoritmos usados para la detección de anomalías.
- Buscar algoritmos de *Deep Learning* para la detección de anomalías en fraudes bancarios.
- Buscar y desarrollar soluciones para los problemas desbalanceados de los datos.
- Desarrollar un programa haciendo uso de los algoritmos encontrados para la detección de anomalías.

- Validar la solución propuesta mediante experimentos con juegos de datos.
- Comparar los resultados obtenidos entre los algoritmos usados.

Para el Banco Central de Cuba, como entidad responsable de velar por la seguridad de los datos y activos, de las entidades estatales o particulares que se encuentran en los bancos cubanos, una herramienta que permita detectar el fraude bancario en tiempo casi real, sería estupendo para la salud financiera del país. Esto permitiría detectar las fugas de dinero que se realizan mediante transacciones en los bancos, detectar los robos de saldos que se realizan y no son percatados por los clientes. El país se beneficiaría grandemente en su saldo financiero al llevar un buen control de sus transacciones.

Se espera desarrollar una herramienta informática que use varios algoritmos para la detección de fraudes bancarios mediante la detección de anomalías en *Deep Learning*. Se pueden auditar las bases de datos de los bancos con estos algoritmos en busca de fraudes bancarios y se espera que el algoritmo se ejecute en tiempo casi real, así se lograría detectar los fraude en el momento y poder revertir o reducir el daño que puede ocasionar.

La estructura del documento se desglosa mediante capítulos. En el capítulo 1 se abarcará todas las bases teóricas relacionadas para el desarrollo del programa, referenciando a todas las investigaciones de la cual se extrajo la información. El capítulo 2 abarca todo lo relacionado con las primeras 7 fases de la metodología que se aplicará para la minería de datos, especialmente en la definición de las estructuras y ecuaciones que aplican los algoritmos seleccionados. El capítulo 3 consta de las últimas 2 fases de la metodología, las cuales abarcan la evaluación y resultados comparativos de los algoritmos teniendo en cuenta las métricas establecidas, además de la definición de la herramienta informática que se desarrolló.

Capítulo 1

Fundamentación teórica

Los bancos cumplen importantes funciones en la sociedad mediante principales funciones que son[1]:

- Canalización del ahorro a través de la demanda de una rentabilidad por la confianza del cliente de su depósito de capital en el banco.
- Seguridad en el depósito de capital. Los bancos guardan el dinero de las personas y tienen sistemas de seguridad muy potentes que permiten garantizar el dinero de sus clientes.
- Préstamos y crédito. Puede ser con disímiles motivos, tanto para la inversión de un negocio o la adquisición de bienes o maquinarias.
- Productos financieros.
- Control del dinero en circulación.
- Cumplimiento de las ratios mínimas de reservas para garantizar la liquidez de la masa de capital de sus clientes.
- Equilibrar el cociente entre expansión del crédito y volumen de depósitos.
- Ofrecer servicios de asesoramiento financiero y patrimonial.
- Permite aplazar pagos y uso de tarjetas de crédito y de débito.

Las tareas relacionadas con el banco que son realizadas por algoritmos de *Machine Learning* son: La detección y prevención de fraude, poner precios a las acciones, la gestión de riesgo, la gestión de crisis, la atribución del cliente, la deserción de clientes, los bancos minoristas, la retención de clientes, la aprobación de créditos y el marketing . El uso de la inteligencia artificial en las actividades bancarias mejora la seguridad de los datos y capitales de los clientes como la salud de las instituciones financieras. Estos se agrupan en cinco grandes grupos como: la mejora de la experiencia del cliente, los *chatbots*, las ofertas personalizadas, la retención de clientes y la detección de fraude bancario.

La detección de fraudes y lavado de dinero cumplen una tarea fundamental de los bancos, que le infunden una gran importancia dentro de la sociedad en que se encuentra. Mediante la inteligencia artificial, esta y las demás tareas de los bancos son optimizadas y mejoradas, entre los beneficios que aporta se encuentran[3]:

- Mayor automatización y mejoramiento de la productividad, permite a los empleados evitar realizar actividades repetitivas que consumen una gran cantidad de tiempo, poder dedicar tiempo a papeleos y la atención a los clientes.
- Servicio personalizado al cliente a un menor costo, usando la capacidad de *Big Data* se puede rastrear y almacenar las operaciones y tendencias del cliente para de esta forma crear y ayudar al cliente de una manera óptima según sus necesidades.
- Mayor precisión de riesgo de activos, teniendo un gran cúmulo de información del cliente evita una posible suplantación de identidad.
- Avanzada detección y prevención de fraude, este es el máximo beneficio de la inteligencia artificial para cualquier institución financiera por la histórica costumbre de los criminales de cometer ilegalidades financieras.

En el sistema bancario actual una significativa mayorí de las transacciones es realizadas mediante tarjetas de crédito o transacciones electrónicas bancarias. Esto permite tener un gran cúmulo de información digital para la detección de fraudes bancarios y para ello existen varias herramientas como la técnica de detección de anomalías. Esta es una técnica de ML que detecta los fraudes automáticamente casi en tiempo real mediante la identificación de correlaciones ocultas en la información.

La detección de anomalías mediante ML permite a través de un conjunto de datos como entrada, obtener reglas. Para la detección de anomalías el conjunto de datos para entrenamiento se clasifica en:

- Supervisado: se provee un conjunto de datos con una elevada cantidad de muestras anómalas o normales.
- Semi-supervisado: se provee un conjunto de datos con todas las muestras anómalas o normales, no existe la combinación.
- No supervisados: no se provee conjunto de datos.

Machine Learning está presente en una gran área del procesamiento de datos, incluido la identificación de fraude de tarjeta de crédito. La detección del fraude de tarjeta se centra en las acciones de transferencia realizadas en la misma. Para ellos es necesario un conjunto de datos para el entrenamiento que posea una mezcla de los dos tipos de transacciones, los fraudes y los normales. Por ende, se debe usar un aprendizaje supervisado, para ello existen diferentes técnicas como son[4]:

- ***Random Forest (RF)***: es una metodología usada solo para mejorar su prosperidad y precisión en algoritmos de ML, también puede ayudar a identificar las variables independientes adecuadas para la selección de funcionalidad del sistema.
- ***Naive Bayes Classifier (NBC)***: es un proceso estadístico basado en teoría predictiva que selecciona su mejor decisión según la probabilidad, también permite la implementación de conocimientos previos como lógica en afirmaciones impredecibles.

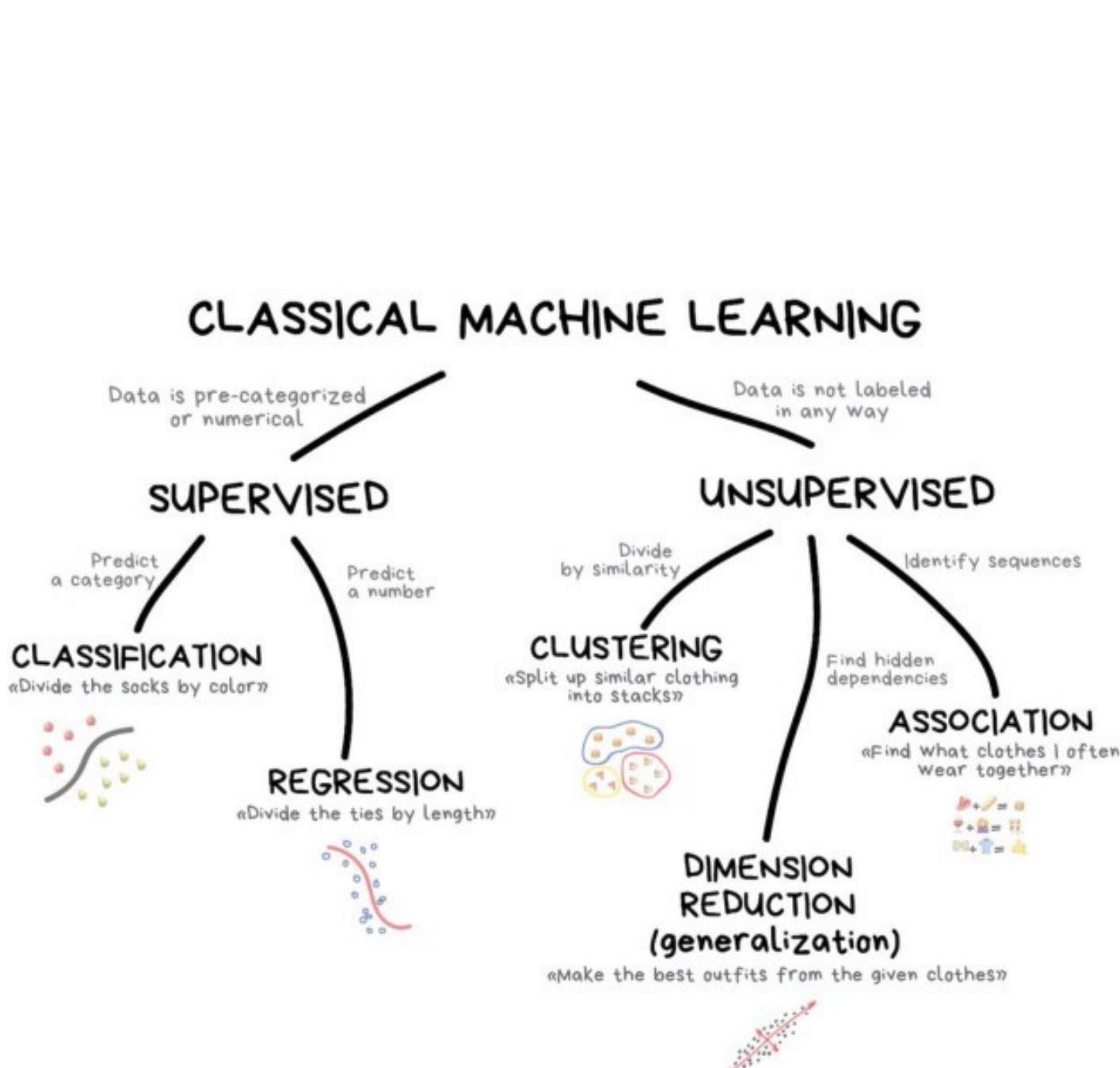


Figura 1.1: Clasificación de Machine Learning

- **Logistic Regression (LR)**: Es otro método decidido a tomar prestado de la profesión de datos estéticos por ML. Es también el proceso de referencia de asuntos concernientes a la categorización binaria. En los casos de detección de amenazas de fraude con tarjetas de crédito, se usa la clase de distribución de probabilidad como fraudulenta y no como fraude.
- **Support Vector Machine (SVM)**: Es un modelo supervisado, usa algoritmos de clasificación de aprendizaje para clasificar problemas importantes incluso en dos grupos e individuales. Pueden clasificar un nuevo documento partiendo de un número de datos nombrados a cada clasificación en un sistema SVM.
- **K-Nearest Neighbors (KNN)**: es un clasificador simple de implementar algoritmo supervisado de ML que puede ser usado para direccionar la clasificación respectiva como las dificultadas regresiones.
- **Classification Trees**: El árbol de clasificación marca, graba y asigna factores de clases separadas, es construido por un proceso llamado particionado recursivo binario.
- **Artificial Neural Network (ANN)**: Este es un método de ML basado en el sistema nervioso del cerebro. Mediante sus diseños y utilizando información histórica, pueden descubrir tendencias y clasificar la información entrante.
- **Restricted Boltzmann Machines (RBM)**: es un algoritmo especializado en la detección de fraudes. El algoritmo aprende de la probabilidad distribuida de los datos de entrada en un modelo no supervisado, no posee una capa neuronal de salida, solo una capa visible y otra oculta. Su importancia es que intenta identificar con qué probabilidad un objeto particular podría activar una característica particular [5].
- **Gradient Boosting (GBM)**: Es un algoritmo prominente de ML, siempre tiene que realizar la clasificación como actividades de regresión. El modelo consiste en una cantidad fundamental de diseños ensamblados como un árbol de decisión débil.
- **Isolation Forest (IF)**: es especialmente diseñado para la detección de anomalías sin importar el tamaño del conjunto de datos, consiste en la creación de varios árboles donde se van particionando al mismo tiempo que se van clasificando [6].
- **Local Outlier Factor (LOF)**: Este algoritmo fue hecho especialmente para la detección de anomalías basados en otros como DBSCAN¹ o KNN. Este algoritmo crea una K cantidad de vecindarios y mediante sus fórmulas calcula la distancia máxima que debe estar un valor de su vecino, si es mayor se considera una anomalía.

Unos de los mayores retos de la identificación de fraude con tarjeta de crédito es la enorme información que se almacenan de las operaciones de las tarjetas, además de que a visión general existe el problema de la información desbalanceada. La clasificación desbalanceada² tiene dos principales causas que son las muestras de datos y las propiedades del dominio [7], en este caso se refiere a la primera causa, donde la mayoría de las transacciones, cerca de un 99% no son fraude, siendo casi imposible detectar actividades fraudulentas. Existen técnicas para aliviar el problema del desbalance entre las clases de información.

¹The density-based spatial clustering application with noise (DBSCAN)

²Es un problema de modelado predictivo de clasificación donde la distribución de ejemplos de las clases no es igual.



Figura 1.2: Campos de la minería de datos

Una de las técnicas para la solución de problemas desbalanceados es el sub muestreo (*Un-*

dersample en inglés), esta técnica consiste en la selección aleatoria de ejemplos de la clase mayoritaria y su eliminación del conjunto de datos de entrenamiento. Una limitación de esta técnica es que los ejemplos se eliminan sin preocuparse por su decisión entre las clases, por lo tanto, es posible que se elimine información útil [8].

Otra forma de resolver este problema es el sobre muestreo (*Oversample*) de los ejemplos en la clase minoritaria, puede ser duplicando ejemplos de la clase minoritaria en el conjunto de datos de entrenamiento antes de ajustar un modelo o sintetizando nuevos ejemplos de la clase minoritaria [9]. El enfoque más utilizado para sintetizar nuevos ejemplos es *Synthetic Minority Oversampling TECnique* (SMOTE), funciona seleccionando ejemplos cercanos en el espacio de características, dibujando una línea entre los ejemplos en el espacio de características y dibujando una nueva muestra en un punto a lo largo de esta línea [10]. También aparece el *Adaptive Synthetic* (ADASYN) que es un algoritmo que genera datos sintéticos y su mayor ventaja es no copiar los mismos datos minoritarios [11].

En la pasada década se han realizado varias investigaciones sobre posibles soluciones acerca del problema desbalanceado tomando de base al algoritmo KNN. Estos grupos de métodos están basados en [12]:

- Estrategias de peso, estos métodos asignan pesos a las muestras de entrenamiento en el vecindario de una muestra testeada.
- Estructura geométrica local de datos.
- Lógica difusa, la pertenencia de cada clase es asignada a un ejemplo como una etiqueta de clase nítida, que puede preservar abundante información clasificada y entonces crear una clasificación completa.
- La falta de estimación de datos positivos.
- Métrica de distancia novedosa.
- El tamaño dinámico de los vecindarios.

Big Data es el conjunto inmenso y diverso de información que se incrementa constantemente, conforma también el volumen de información, la velocidad con que es creada o recolectada la misma. Puede ser categorizada en no estructurada o estructurada: los datos estructurados consisten en información administrada por la organización en base de datos u hojas de cálculo, mientras que los datos no estructurados es la información desorganizada que no se delimita por un formato o modelo [13]. Las fuentes de datos son las redes sociales, motores de búsquedas de internet, plataformas de comercio electrónico, cines en línea, entre otras formas de interacción en línea. Normalmente la información es no estructurada, es tan vasta que tomaría demasiado tiempo por los humanos para extraer información relevante y útil.

Los retos del *Big Data* están ampliamente definidos en las cinco Vs: valor, veracidad, variedad, velocidad y volumen. Valor se refiere a los beneficios asociados con el análisis de datos, la veracidad a la precisión de los datos; variedad es la cantidad de tipos de datos, como los estructurados, semi-estructurados o los no estructurados. El volumen es la cantidad de datos que es acumulada y la velocidad se refiere con la gran rapidez con que se generan los datos y sus muchas dimensiones [14]. Con mayor cantidad de dimensiones se crean dificultades para la detección de anomalías, porque cuando el número de atributos incrementa, la cantidad de datos necesarios para categorizar aumenta, concluyendo en la dispersión de datos.

Un subconjunto de técnicas de ML comúnmente usada para procesar *Big Data* es *Deep Learning* (DL), también conocido como *Deep Neural Networks* o *Neural Learning*, es un subconjunto de ML, utiliza niveles de jerarquía de ANN para llevar procesos de ML. Comparada con técnicas de ML como SVM y KNN, DL posee ventajas de los aprendizajes no supervisados, una fuerte capacidad de generalización, y un poderoso entrenamiento robusto para *Big Data* [15]. El DL posee grandes ventajas con respecto a ML, una de ellas es la redundancia de la extracción de características.

Los métodos tradicionales de ML (*Decision Tree* (DT), SVM y LR) eran los más populares, las extracciones de características suelen ser complicadas y requiere conocimiento detallado del dominio del problema. Este paso debe adaptarse, probarse y perfeccionarse en varias iteraciones para obtener resultados óptimos.

Los modelos de DL no necesitan la extracción de características, en estos casos se tienen ANN. Las capas pueden aprender una representación implícita de los datos sin procesar por sí mismas. Un modelo de DL produce una representación abstracta y comprimida de los datos sin procesar en varias capas de una ANN, luego son usadas para producir el resultado. Los modelos de DL requieren poco o ningún esfuerzo manual para realizar y optimizar el proceso de extracción de características, es decir, está integrada en el proceso que tiene lugar dentro de la ANN.

Otra gran ventaja del DL es que funciona con cantidades masivas de datos. Los modelos de DL tienden a aumentar su precisión con la creciente cantidad de datos de entrenamiento, mientras que los modelos tradicionales de ML dejan de mejorar después de un punto de saturación.

Se han creado varios algoritmos que permiten la detección de fraudes bancarios, a continuación, se desglosan algunos de los algoritmos:

- ***Convolutional Neural Network (CNN)***: CNN es un método DL altamente asociado con datos especiales, similar a ANN, posee la misma estructura de capa oculta en adición a la capa especial convolucional con diferente número de canales en cada capa [16].
- ***Recurrent Neural Network (RNN)***: RNN es un acercamiento dinámico de ML capaz de analizar los comportamientos temporales dinámicos de varias cuentas bancarias por la modelación de dependencias secuenciales entre transacciones consecutivas de los dueños de las tarjetas de crédito [17]. En otras palabras, RNN es una red neuronal con memoria que tiende a tener un corto término de memoria por el problema de desaparición de gradiente. *Backpropagation* es la columna vertebral de las redes neuronales, tanto que minimiza la pérdida ajustando pesos de red que son encontrados usando gradientes.
- ***Back-Propagation Neural Network (BPNN)***: BPNN es una multicapa *feedforward neural network (FNN)*, es uno de los modelos ANN ampliamente usados. La regla de aprendizaje es usar el método del descendiente más empinado y modificar repetidamente los pesos y bases de la red a través de una iteración reversa, entonces la suma de los cuadrados de los errores es minimizada [18].
- ***Long Short Term Memory Network (LSTM)***: LSTM es un tipo de RNN mejorado, resuelve el problema del corto término de memoria. Tiene celdas de estado que es la memoria de la red, que existe por cada paso, además de puertas en cada paso que controla el flujo de memoria que mantiene necesariamente y descarta información irrelevante [16].

- **Autoencoder (AE)**: AE es una clase especial de algoritmo DL, donde la salida es la misma que la entrada, pero tiene una capa media u oculta que contiene menos neuronas y comprime los datos. AE usa varias capas para codificar y para decodificar en la capa oculta [5]. Cuando los datos son decodificados, la salida es comparada con la entrada y si no son iguales, se activa un mecanismo de corrección hasta que se alcance el mínimo error. En este proceso son detectadas las anomalías, las cuales son en el caso de la problemática, las transacciones fraudulentas.
- **Denoising Autoencoder (DAE)**: DAE es una extensión de AE, que permite aprender más filtros robustos en las capas ocultas, reduce el riesgo de sobreajuste y previene de aprender una función de identificación simple [19].
- **Dropout Regularization**: *Dropout* es una técnica donde aleatoriamente selecciona neuronas que son ignoradas durante el entrenamiento, esto significa que sus contribuciones a la activación de neuronas vía abajo es temporalmente removida en el paso hacia adelante y ningún peso es actualizado a la neurona en el paso hacia atrás [20].

Varios investigadores han creado sus propias versiones de algoritmos DL mediante el aumento de neuronas o creación de híbridos. Por lo tanto, existen métricas para demostrar la efectividad de estos modelos por la cantidad de aciertos [21]. Estos son:

- **Accuracy**: es el número total de predicciones correctas dividido por el número total de predicciones.
- **Precision**: define cuan confiable es un modelo en responder si un punto pertenece a una clase.
- **Recall**: expresa cuan bien puede el modelo detectar a una clase.
- **F1 Score**: es dada por la media harmónica de *precision* y *recall*, combina ambas métricas en una sola.

Teniendo en cuenta estas métricas existen cuatro resultados posibles, las cuales se pueden interpretar como:

- **Alta precision y alto recall**: el modelo maneja perfectamente la clase.
- **Alta precision y bajo recall**: el modelo no detecta la clase muy bien, pero cuando lo hace es altamente confiable.
- **Baja precision y alto recall**: la clase detecta bien la clase, pero también incluye muestras de otras clases.
- **Baja precision y bajo recall**: El modelo no logra clasificar la clase correctamente.

Para el desarrollo y funcionamiento de los algoritmos en un sistema de cómputo es necesario la utilización de tecnologías y herramientas de programación. Una de las tecnologías a tener en cuenta son los lenguajes de programación, este debe: poseer bastantes y buenas librerías de ML y DL, debe tener un buen rendimiento en tiempo de ejecución, un buen soporte de herramientas, una comunidad de programadores y un ecosistema saludable de paquetes de soporte. Algunos

de los lenguajes que poseen estos requisitos son: Python³, C++⁴, Java⁵ y lenguajes de JVM⁶, JavaScript⁷, Swift⁸ y lenguaje R⁹.

Para el desarrollo de este proyecto se hará uso del lenguaje de programación Python en su versión 3.7.9, este lenguaje de programación se encuentra a la cabeza de los más utilizados para DL, posee una alta gamma de librerías, es un lenguaje sencillo y de alto rendimiento, posee bastantes comunidades de programadores que suben soluciones y repuesta sobre el tema. Este lenguaje lleva un destacado recorrido en el área del DL.

Las librerías que se usarán en el proyecto son las siguientes:

- Tensorflow.Keras: Keras es un API para personas que sigue buenas prácticas para reducir la carga cognitiva: ofrece consistencia y simple APIs, minimiza el número requerido de acciones de usuario para casos de usos comunes [22].
- Numpy: es una librería de Python usada para el trabajo con arreglos. También tiene funciones para trabajar en el dominio de álgebra linear y matrices [23].
- Panda: es un paquete de Python que provee una estructura diseñada de datos rápida, flexible y expresiva para hacer trabajo con la clasificación de datos [24].
- Sickit-learn: es una herramienta de código abierto, simple y eficiente para realizar análisis de datos predictivos [25].
- Matplotlib: es una librería comprensiva para la agrupación estática, animada y visualizaciones interactivas en Python [26].
- Imbalanced-learn: proporciona herramientas para tratar el problema de información desbalanceada. En esta librería se encuentran los algoritmos de UnderSample, OverSample, SMOTE y ADASYN [27].

El IDE a utilizar es el PyCharm 2019.2.1 (*Professional Edition*)[28], es especializado para aplicaciones de Python.

La minería de datos también conocida como *Knowledge Discovery in Database* (KDD), es el proceso para descubrir patrones útiles o conocimientos a partir de fuentes de datos. Los patrones deben ser válidos, potencialmente útiles y entendibles. La minería de datos es un campo multidisciplinario que incluye: aprendizaje automático, estadísticas, sistemas de bases de datos, AI, *Information Retrieval*, visualización de la información, etc. Entre las ventajas que ofrece se encuentran [29]:

- La minería de datos descubre información que no se esperaba obtener. Como muchos modelos diferentes son usados, algunos resultados inesperados tienden a aparecer. Las combinaciones de distintas técnicas otorgan efectos inesperados que se transforma en un valor añadido a la empresa.

³Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad del código.

⁴C++ es un lenguaje de programación diseñado para extender al lenguaje C y posee mecanismos que permiten la manipulación de objetos.

⁵Java es un lenguaje de programación y una plataforma informática.

⁶Java Virtual Machine (JVM) es una máquina virtual de proceso nativo, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial.

⁷JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript.

⁸Swift es un lenguaje de programación potente e intuitivo para iOS, iPadOS, macOS, tvOS y watchOS.

⁹R es un entorno y lenguaje de programación con un enfoque al análisis estadístico.

- Enormes bases de datos pueden ser analizadas mediante la tecnología de *data mining*.
- Los resultados son fáciles de entender.
- Contribuye a la toma de decisiones tácticas y estratégicas para detectar la información clave.
- Mejora la relación con el cliente.
- Los modelos son confiables. Los modelos son probados y comprobados usando técnicas estadísticas antes de ser usado, para que las predicciones que se obtienen sean confiables y válidas.
- En su mayoría, los modelos se generan y construyen de manera rápida. El modelado a veces se torna más fácil puesto que muchos algoritmos han sido probados previamente.

Existen metodologías para realizar la minería de datos, los cuales son [30]:

- *Knowledge Discovery in Databases* (KDD): esta metodología propone 5 fases: Selección, pre procesamiento, transformación, minería de datos y evaluación e implementación. Es un proceso iterativo e interactivo.
- CRISP-DM: *Cross-Industry Standard Process for Data Mining*, iniciativa financiada por la comunidad europea para desarrollar una plataforma para Minería de Datos.
- SEMMA: es el acrónimo a las 5 fases: *Sample, Explore, Modify, Model, Assess*. La metodología es propuesta por SAS Institute Inc.

Para este proyecto se aplicará la metodología KDD que es un análisis exploratorio, automático y modelado de grandes repositorios de datos. Es un proceso organizado de identificación válida, novel, útil, y entendibles patrones de un gran y complejo conjunto de datos [31]. Para ello se definen los pasos de esta metodología para orientar el desarrollo:

1. Entendimiento, conocimientos previos e identificación de la meta.
2. Selección de un conjunto de datos.
3. Limpieza y preprocessamiento de datos.
4. Transformación de los datos.
5. Colocar el objetivo del KDD a un método de minería de datos.
6. Elección de los algoritmos de minería de datos.
7. Implementación de los algoritmos de minería de datos.
8. Evaluación.
9. Aplicación del conocimiento adquirido.

En el próximo capítulo se abordará la metodología KDD secuencialmente atendiendo a los pasos definidos anteriormente.

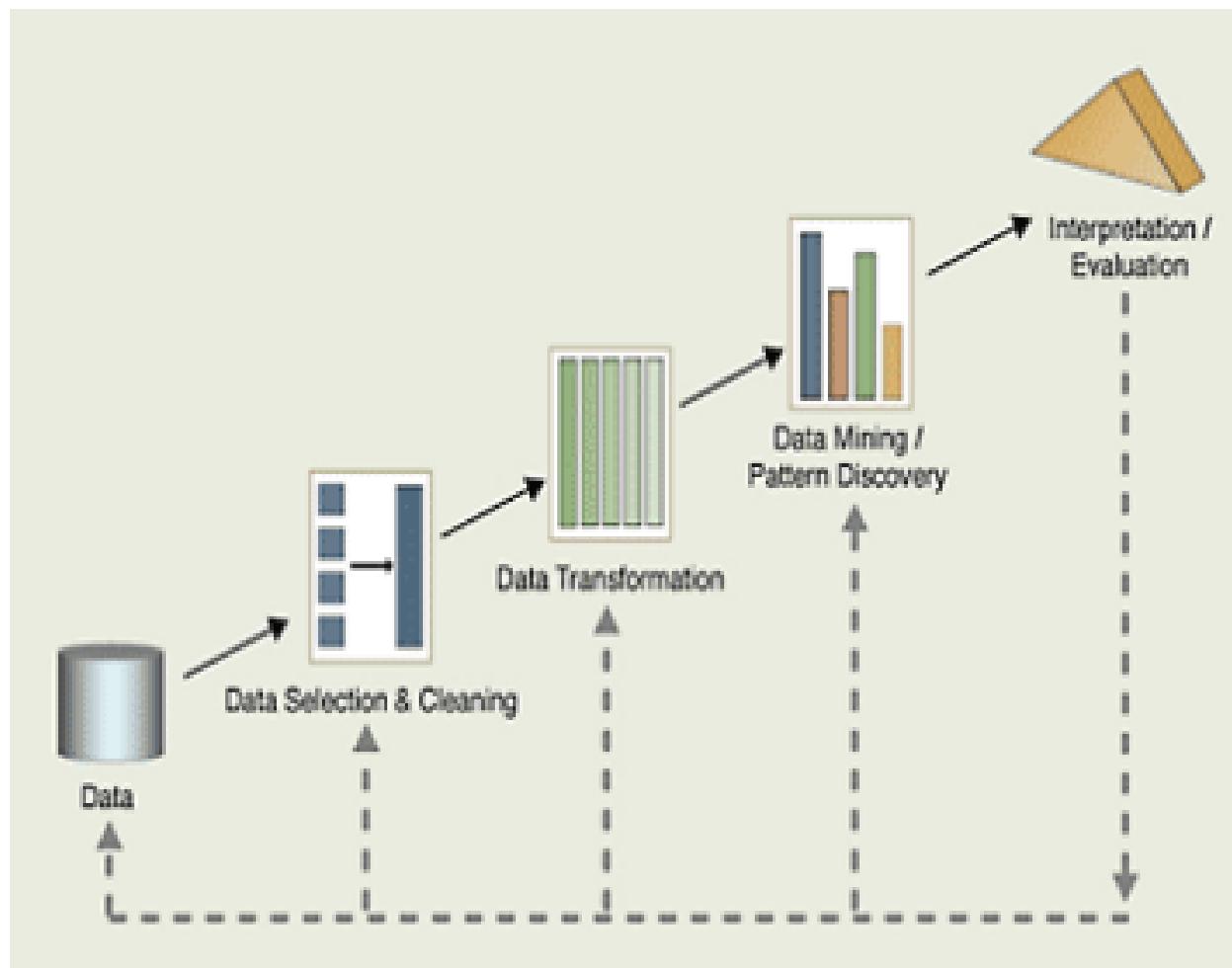


Figura 1.3: Secuencia KDD

Capítulo 2

Preparación de la minería de datos

Este capítulo abarca la información necesaria para entender todo el procedimiento que se llevará a cabo para realizar las pruebas, desde la selección del conjunto de datos hasta la estructura de los algoritmos para obtener los resultados. Para ello este capítulo se desglosa en secciones secuenciales para guiar al lector paso a paso para poder repetir el experimento.

2.1 Paso 1: Entendimiento, conocimientos previos e identificación de la meta

Este es el paso inicial preparatorio donde se desarrolla el entendimiento del dominio de la aplicación. Se definen las decisiones que se tomarán en los siguientes pasos, ya sea la transformación, los algoritmos, representación, etc. Además, se define el entorno en que se desarrollará la aplicación junto a las características, tecnologías y herramientas que se usan en el proyecto, las cuales fueron definidas en el capítulo anterior, por lo que en esta sección se procede a tomar las decisiones de los próximos pasos. El objetivo que se persigue con esta metodología es la predicción, básicamente en predecir satisfactoriamente la clase de los datos pasados como prueba.

En la selección del conjunto de datos no es necesario crear los datos ya que se utiliza un conjunto de datos extraídos de Kaggle. Debido a ello no es necesario extraer informac'on de varias fuentes e integrarlas en una estructura homogénea.

En el tercer paso se procede a realizar la limpieza de los datos en caso de ser necesario y luego se pasa al preprocesamiento. Para la limpieza es necesario detectar que los datos estén completos y homogéneos, en caso de que existan datos incompletos en una transacción, se procede a ser eliminada la transacción del conjunto de datos. En caso de no estar homogéneos los tipos de datos por atributos y se convierten los que no pertenezcan a ese tipo. El preprocesamiento de los datos se llevará a cabo porque existen un amplio desbalance de la información, dígase cuando la clase minoritaria represente menos de 10% del total de muestras del conjunto de datos, en cuyo caso se procederá a balancear mediante los algoritmos de balanceado de información mencionados en el capítulo anterior. Cada uno de los conjuntos obtenidos mediante el balance de la información será para entrenar los modelos incluyendo la versión desbalanceada.

En el cuarto paso se lleva a cabo la transformación de la información, ya sea la reducción de dimensiones o discretización. La reducción de dimensiones se llevará a cabo eliminando

atributos que no sean necesarios para el preprocesamiento o extrayéndolos para que no afecten o lancen resultados ficticios. En el caso de este proyecto los datos son valores numéricos reales, por lo tanto, no es necesario realizar la discretización.

En el quinto paso se decide que método de minería de datos usar, que es clasificación debido principalmente al objetivo y a las características que tiene este método.

En el sexto y séptimo paso se eligen e implementan los algoritmos para la minería de datos, los cuales son de DL por el enfoque de las métricas de comparación. Se establecerán la estructura de cada algoritmo a emplear, además de un enfoque matemático al funcionamiento de cada uno.

El octavo y noveno paso, los cuáles son los últimos, forman parte del capítulo siguiente, donde se realizarán la evaluación, comparación y representación de los resultados de los algoritmos seleccionados. Luego de escoger los mejores algoritmos, se aplicará el conocimiento adquirido.

2.2 Paso 2: Selección del conjunto de datos

El conjunto de datos S consta de 20 000 filas con 31 columnas, es decir que, se trabaja con 20 000 transacciones de las cuales se tienen 31 características. De estas 31 características existe información privada nombradas como V1, V2, V3, hasta V28, *Time* representa el instante de tiempo en que se realizó dicha transacción, *Amount* es la cantidad de la transacción realizada, y, por último, *Class* tiene dos posibles valores: 0 para las transacciones normales y 1 para las transacciones fraudulentas.

Del total de 20 000 muestras, 19 936 son transacciones normales y 64 son transacciones fraudulentas, representando los fraudes un 0.32% del conjunto de muestras, evidenciando un desbalance de la información y un porcentaje inferior al 10%. Por lo tanto, es necesario llevar a cabo un balanceado de la información en el preprocesamiento.

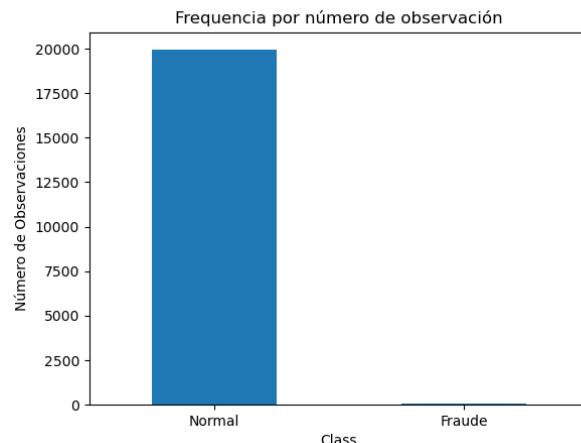


Figura 2.1: Desbalance de las clases

2.3 Paso 3: Limpieza y preprocessamiento

La primera acción a realizar es verificar si los datos están completos y homogéneos. Luego de la verificación se obtiene como resultado que los datos están completos y presenta su homogeneidad, como parte de este proceso de verificación se presentan valores anómalos que son el propósito de esta investigación por lo que se mantienen dentro del conjunto de datos. Una vez concluido se procede al preprocessamiento de la información.

Luego de la limpieza es necesario dividir los datos en dos conjuntos, uno de entrenamiento y otro para la prueba. Debido al gran desbalance de información que poseen los datos se le aplicarán las soluciones mencionadas en el capítulo anterior, aunque solo se le aplicarán al conjunto de entrenamiento. Todos los algoritmos para solucionar el desbalance de información que se presentarán pertenecen a la librería imbalanced-learn.

2.3.1 RandomUnderSampler

RandomUnderSampler aplica un algoritmo de selección de prototipos que selecciona ejemplos del conjunto de datos S . Entonces, S' es definido como $|S'| < |S|$ y $S' \in S$. Este algoritmo es una técnica de submuestreo controlada, es una forma de balancear los datos seleccionando aleatoriamente un subconjunto de datos de las clases.

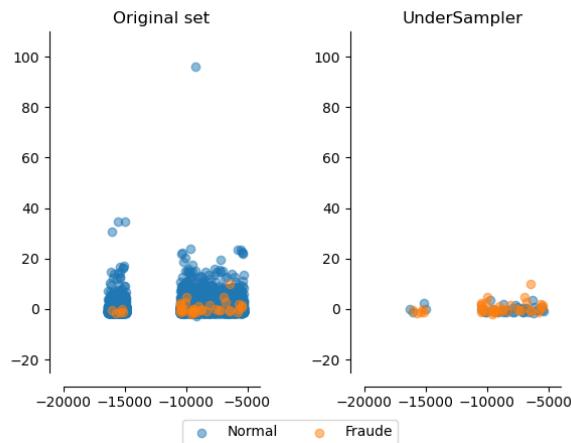


Figura 2.2: Aplicación de UnderSampler

UnderSampler a pesar de ser un algoritmo que soluciona el problema desbalanceado, tiene una gran desventaja y es que elimina información para el entrenamiento y el conjunto de datos se vuelve extremadamente pequeño, reduciendo la capacidad de predicción de los modelos.

2.3.2 RandomOverSampler

RandomOverSampler genera nuevos ejemplos en las clases que están subrepresentadas mediante el sobre muestreo aleatorio de la clase minoritaria, en otras palabras, multiplicar los datos de la clase minoritaria hasta lograr el equilibrio. Formalmente S' se define como $|S'| > |S| \text{ y } S \in S'$.

A diferencia de UnderSampler, este algoritmo no elimina información, pero no adiciona

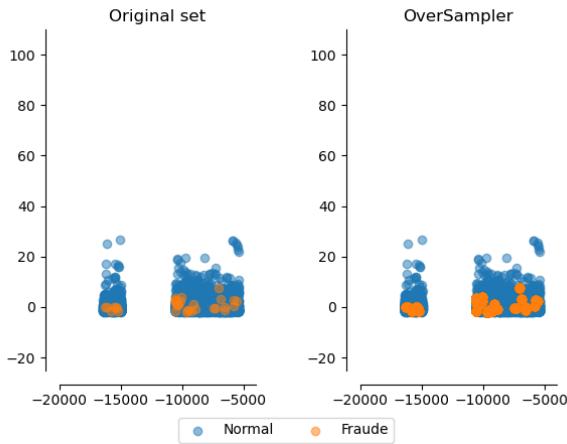


Figura 2.3: Aplicación de OverSampler

ninguna información nueva para el entrenamiento por solo repetir los datos de la clase minoritaria.

2.3.3 SMOTE y ADASYN

SMOTE y ADASYN generan nuevas muestras sintéticas por interpolación que difieren de las originales. La diferencia se centra en que ADASYN se enfoca en generar muestras sintéticas junto a las muestras originales que están clasificadas incorrectamente usando un clasificador KNN, mientras que la implementación de SMOTE no hará ninguna distinción entre muestras erróneas o correctas para ser clasificadas usando la regla KNN. Por lo tanto, ambos algoritmos usan el mismo algoritmo.

SMOTE, a partir de una muestra x_i de la clase minoritaria, se generará una nueva muestra x_{new} por su vecino más cercano x_{zi} , y esta nueva muestra es incluida al conjunto de datos, esto se repite hasta que los datos sean balanceados. Su ecuación es:

$$x_{new} = x_i + \lambda \times (x_{zi} - x_i) \quad (2.1)$$

Donde λ es un número aleatorio entre 0 y 1. Esta interpolación creará una muestra sobre la línea entre x_{zi} y x_i .

ADASYN a diferencia de SMOTE, el número de muestras generadas para cada x_i es proporcional al número de muestras que no son de la misma clase que x_i en una vecindad determinada. Por lo tanto, se generarán más muestras en el área donde no exista una fuerte densidad de la clase minoritaria.

2.3.4 Paso 4: Transformación de los datos

Los datos poseen 31 dimensiones o atributos, la dimensión *Class* va a ser separada del conjunto de datos y la dimensión *Amount* será eliminada del preprocesamiento. Por lo tanto, se trabajará con 29 dimensiones, mientras que la dimensión *Class* será usada para clasificar los datos preprocesados y poder ser comparados con las predicciones.

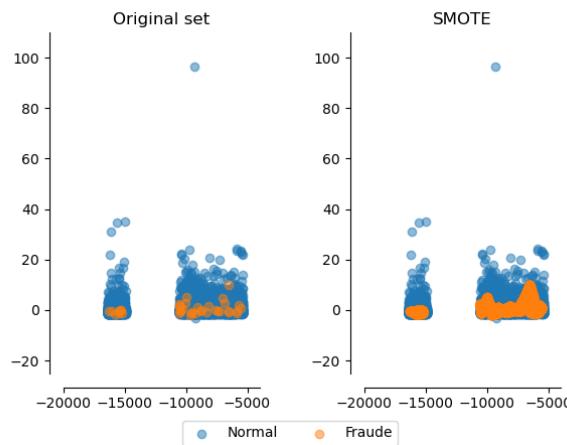


Figura 2.4: Aplicación de SMOTE

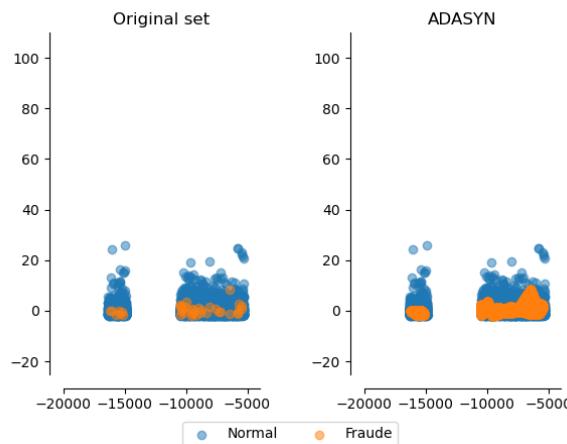


Figura 2.5: Aplicación de ADASYN

La división de los datos en los conjuntos de entrenamiento y prueba, se realiza con una proporción de 80% para el entrenamiento y 20% para las pruebas de forma estándar. No obstante, se realizarán pruebas para comprobar cuál es la mejor forma de dividir los grupos en cuanto a por ciento de datos para pruebas.

Al entrenar los modelos es necesario realizar la estandarización y escalado de variables numéricas. Al ser numéricos los predictores, la escala en que se miden y la magnitud de su varianza pueden influir en gran medida en el modelo. Si no se igualan de alguna forma los predictores pueden ocasionar que las variables objetivas no dominen el modelo y la variable respuesta no guarde relación con ellas. Para ello se usará la estrategia de normalización que consiste en transformar los datos de forma que todos los predictores estén aproximadamente en la misma escala. Sus dos variantes son:

- Normalización Z-score (StandardScaler): se divide cada predictor entre su desviación típica después de haber sido centrado.
- Estandarización max-min (MinMaxScaler): transforma los datos de forma que estén dentro del rango [0, 1].

2.4 Paso 5: Método de minería de datos

Existen tres métodos de minería de datos:

- Regresión: usa algoritmos de aprendizaje supervisado para predecir y modelar variables aleatorias continuas. Es usado para cambios continuos como pronósticos de precios de vivienda, tendencia de acciones o resultados de pruebas.
- Clasificación: usa algoritmos de aprendizaje supervisados para modelar o predecir variables aleatorias discretas. Se usan en varias tareas como filtrado de correo electrónico, fraude financiero y predicción de cambios de empleados.
- Agrupación: usa algoritmos de aprendizaje no supervisado, estos encuentran los grupos étnicos naturales de las muestras observadas basándose en la estructura interna de los datos. Es usado en la segmentación de clientes, agrupación de noticias, recomendación de artículos, etc.

La predicción es a veces referida como una minería de datos supervisada, entonces el método a usar es la clasificación debido al enfoque del proyecto que es la detección de fraude de tarjeta de crédito.

2.5 Paso 6 y 7: Elección e implementación de los algoritmos de minería de datos

A partir de la estrategia seleccionada, se selecciona en este paso los métodos para la búsqueda de patrones. Considerando las métricas seleccionadas para comparar los algoritmos, se decide usar las redes neuronales, en específico las pertenecientes a DL, para conocer cuáles son los algoritmos apropiados para obtener óptimos resultados.

Las redes neuronales o ANN son modelos creados al ordenar operaciones matemáticas siguiendo determinada estructura. La forma más común de representar su estructura es mediante el uso de capas (*layers*), formadas a su vez por neuronas (*units* o *neurons*). Cada neurona, realiza una operación sencilla y está conectada a las neuronas de la capa anterior y de la capa siguiente mediante pesos, cuya función es regular la información que se propaga de una neurona a la otra. La primera capa de la red es la capa de entrada (*Input Layer*), donde se reciben los datos sin clasificar, luego pasa por la capa intermedia o capa oculta (*hidden layer*) donde se reciben los datos de la capa de entrada, ponderados por los pesos. Finaliza en la última capa (*output layer*) donde combina los valores que salen de la capa intermedia para generar la predicción.

La neurona es la unidad funcional de los modelos de redes, donde ocurren simplemente dos operaciones: la suma ponderada de sus entradas y la aplicación de una función de activación. En la primera se multiplica cada valor de entrada x_i , por su peso asociado w_i y se suman junto con la parcialidad b , luego este valor pasa por una función, conocida como función de activación a , transformando el valor neto de entrada en un valor de salida S . Entonces la ecuación para obtener los datos de entrada en las capas intermedias se define como $E = \sum_{i=0}^n x_i w_i + b_i$, luego la salida de la neurona se convierte en $S = a(x_i w_i + b_i)$.

2.5. PASO 6 Y 7: ELECCIÓN E IMPLEMENTACIÓN DE LOS ALGORITMOS DE MINERÍA DE DATOS

Las funciones de activación controlan la información que se propaga de una capa a la siguiente (*forward propagation*). Entre las funciones a utilizar se encuentran las siguientes:

- *Rectified linear unit* (ReLU): aplica una transformación no lineal donde la neurona se activa solo si la entrada está por encima de 0, esto hace que se retengan los valores positivos y descarta los negativos dándoles una activación de 0. $\text{ReLU}(x) = \max(x, 0)$.
- Sigmoide: transforma los valores en el rango $(-\infty, +\infty)$ a valores en el rango $(0, 1)$. $\text{sigmoid}(x) = \frac{1}{1+\exp(-x)}$.
- Softmax: esta función calcula las probabilidades relativas, parecido a sigmoid. $\text{softmax}(x) = \frac{\exp(x_i)}{\sum_j \exp(z_j)}$.

La función de coste l o función de pérdida (*loss function* o *cost function*), es la encargada de cuantificar la distancia entre el valor real y el predicho por la red. En el caso del método de clasificación, se utiliza para dirigir el entrenamiento de los modelos, por lo que se elige el método *binary cross-entropy loss*. Como la clasificación es binaria, la variable de respuesta es 1 o 0, y la probabilidad $p = \text{Pr}(y = 1)$. La función para ello es:

$$L_{\log}(y, Pr) = -\log \text{Pr}(y | p) = -(y \log(p) + (1 - y) \log(1 - p)) \quad (2.2)$$

El modelo de red neuronal con una única capa, sólo es capaz de aprender patrones sencillos. Para romper con esta limitación se combinan múltiples capas ocultas, permitiendo a la red aprender relaciones mucho más complejas entre los predictores y la variable de respuesta. Combinando múltiples capas ocultas y funciones de activación no lineales, los modelos de redes pueden aprender prácticamente cualquier patrón.

En el proceso de entrenamiento de una red neuronal consiste en ajustar el valor de los pesos y parcialidad de tal forma que se obtenga el menor error posible. Debido a ello el modelo es capaz de identificar qué predictores tiene mayor influencia y de qué forma están relacionados entre ellos y con la variable respuesta. Para lograr disminuir el error se han combinado múltiples métodos matemáticos, entre ellos:

- Retro propagación (*backpropagation*): es el algoritmo que permite cuantificar la influencia que tiene cada peso y parcialidad de la red en sus predicciones. Hace uso de la regla de la cadena para calcular el gradiente, esto permite identificar que pesos de la red hay que modificar para mejorarla.
- Descenso de gradiente estocástico (SGD): consiste en dividir el conjunto de entrenamiento en lotes (*minibatch* o *batch*) y actualizar los parámetros de la red con cada uno. De esta forma en vez de esperar evaluar todas las observaciones para actualizar los parámetros, se pueden ir actualizando de forma progresiva.
- Adam: es una combinación de RMSprop¹ y SGD con impulso. Utiliza los gradientes cuadrados para escalar la tasa de aprendizaje como RMSprop y aprovecha el impulso mediante el uso de la media móvil del gradiente en lugar del gradiente como SGD con impulso [33]. Es uno de los optimizadores con mayor tendencia en la actualidad.

¹RMSprop: algoritmo que se utiliza para la optimización de lotes completos, intenta resolver la amplia variación de las magnitudes de los gradientes.

Es importante mencionar que una ronda completa de iteraciones sobre todos los *batchs* se llama época. El número de épocas con las que se entrena una red equivale al número de veces que la red ve cada ejemplo de entrenamiento. Es decir que la cantidad de épocas influye en los resultados del modelo predictor.

Las redes neuronales son capaces de generar modelos que aprenden relaciones muy complejas, sin embargo, sufren fácilmente el problema de sobreajuste (*overfitting*) lo que los incapacita al tratar de predecir nuevas observaciones. La forma de minimizar este problema y conseguir modelos útiles es configurando de forma adecuada sus hiperparámetros:

Número y tamaño de capas

La capa de entrada tiene tantas neuronas como predictores y la capa de salida tiene tantas neuronas como clases en problemas de clasificación. Cuantas más neuronas y capas tengan el modelo, mayor la complejidad de las relaciones que puede aprender el modelo. Sin embargo, dado que cada neurona está conectada por pesos al resto de neuronas de las capas adyacentes, el número de parámetros a aprender aumenta y con ello el tiempo de entrenamiento. Debido a ello los modelos tendrán a lo sumo 10 capas, para lograr compensar la efectividad de los predictores con el tiempo estimado de entrenamiento.

Ratio de aprendizaje

La ratio de aprendizaje (*learning rate*) establece cómo de rápido pueden cambiar los parámetros de un modelo a medida que se optimiza. Es uno de los más complicados de establecer, ya que depende mucho de los datos e interacciona con el resto de hiperparámetros. Si el *learning rate* es muy grande, el proceso de optimización puede ir saltando de una región a otra sin que el modelo sea capaz de aprender. Por el contrario, si es muy pequeño, el proceso de entrenamiento puede demorar demasiado y no llegar a completarse.

Se utiliza el *learning rate* definido como estándar de cada optimizador, en el caso de Adam es 0.001 y en el caso de SGD es 0.01.

Algoritmo de optimización

El SGD y Adam son los algoritmos de optimización a utilizar en el proyecto, a pesar de que SGD presente problemas a medida que los modelos aumentan de tamaño. Por lo tanto, SGD es utilizado para modelos que presenten pocas neuronas y capas en el modelo, mientras que Adam será utilizado para conjunto de datos grandes, aunque puede ser utilizado en cualquier tipo de modelos.

Regularización

Los métodos de regularización se utilizan con el objetivo de reducir el sobreajuste de los modelos. Un modelo con sobreajuste memoriza los datos de entrenamiento, pero es incapaz de predecir correctamente nuevas observaciones. Debido a que los modelos de redes neuronales pueden considerarse como modelos sobre parametrizados, se adopta el *dropout* como estrategia de regularización.

2.5. PASO 6 Y 7: ELECCIÓN E IMPLEMENTACIÓN DE LOS ALGORITMOS DE MINERÍA DE DATOS

Dropout consiste en desactivar aleatoriamente una serie de neuronas durante el proceso de entrenamiento. Durante cada iteración del entrenamiento se ponen a cero los pesos de una fracción aleatoria de neuronas por capa. El porcentaje de neuronas que suele desactivarse por capa (*dropout rate*) suele ser un valor entre 0.2 y 0.5. Por lo tanto, en el proyecto se les aplicarán a los algoritmos tres capas de *dropout* con capas intermedias, cada uno con un porcentaje de neuronas desactivadas diferentes. Para establecer el orden y porcentaje se aplicarán experimentos con diferentes combinaciones, hasta encontrar el óptimo.

Una vez definidas las bases, opciones y estrategias a aplicar para obtener óptimos resultados en los modelos, se procede a definir la estructura de los algoritmos seleccionados para el proyecto.

AutoEncoder (AE)

El AE consiste en tres partes, la capa de entrada donde se introducen los datos originales sin clasificar, la capa oculta para codificar la entrada y la capa de salida para decodificar. Usando *Backpropagation*, el algoritmo se auto entrena continuamente cambiando los valores de la salida objetivo por los valores de entrada. Esto fuerza a que la capa más pequeña del codificador use la reducción dimensional para eliminar ruido y reconstruir los datos de entrada, de esta forma aprenden como comprimir los datos de la capa de entrada y luego descomprimir en cualquier formato que mejor se aadecue a los datos originales. AE está estrechamente relacionado al análisis de componente principal (PCA), principalmente en el cuello de botella que se encuentra en la capa más pequeña de la red, donde corresponde con sus principales componentes.

Se puede expresar el proceso matemáticamente, para ello se divide la red en dos partes, el codificador θ y el decodificador ϕ , es necesario aclarar que la capa de entrada con los datos originales X no se incluyen en ninguna. Además, se define F como la representación de los datos en el cuello de botella o capa de menor dimensión.

$$\theta : X \rightarrow F \quad (2.3)$$

$$\phi : F \rightarrow X \quad (2.4)$$

$$\theta, \phi = \operatorname{argmin}_{\theta, \phi} \|X - (\theta \circ \phi)X\|^2 \quad (2.5)$$

Básicamente se intenta recrear los datos originales luego de aplicar una compresión no linear generalizada. En la parte codificadora puede ser representada por la función de activación de una red neuronal estándar, donde $z = \sigma(Wx + b)$ es la dimensión latente. De manera similar se puede representar la parte decodificadora como $x' = \sigma'(W'z + b')$.

Al algoritmo AE se le aplicará tres capas de *Dropout* no consecutivas y no necesariamente con el mismo porcentaje de eliminación para evitar el *overfitting*; y el optimizador Adam. Posee 9 capas para su funcionamiento, que consiste en una capa de datos de entrada, cuatro capas para el codificador y cuatro capas del decodificador. Para lograr un entendimiento de la reducción de las dimensiones se especificarán las dimensiones cada capa y su secuencia:

1. *Input Layer*: es la capa de entrada de datos, por lo tanto, se introducen los datos en bruto sin procesar, estos datos poseen 29 dimensiones.
2. *Dense1 Layer*: Reduce las dimensiones de los datos de entrada a 14 dimensiones.

3. *Dense2 Layer*: Reduce nuevamente las dimensiones a 7.
4. *Dropout1 Layer*: Elimina modelos de patrones según el por ciento especificado, de esta forma se evita el *overfitting*. *Dense3 Layer*: Reduce las dimensiones a 1. Hasta este punto se realiza la codificación. *Dropout2 Layer*: Elimina modelos de patrones nuevamente. A partir de este paso comienza la decodificación.
5. *Dense4 Layer*: Incrementa las dimensiones a 7 de forma tal que va clasificando los datos.
6. *Dropout3 Layer*: Elimina patrones nuevamente.
7. *Dense5 Layer*: Incrementa las dimensiones a 29, igual que las dimensiones de entrada.
8. *Dense6 Layer o Output Layer*: Comprime las dimensiones a 1 y concluye la clasificación.

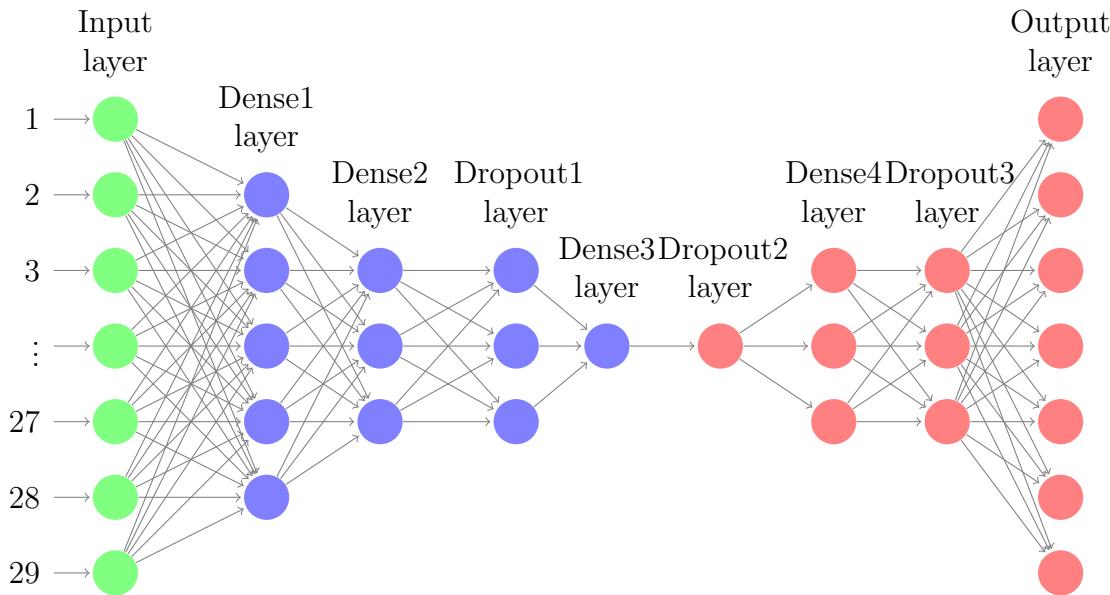


Figura 2.6: Modelo AE

Denoising AutoEncoder (DAE)

DAE es una versión estocástica de AE que reduce el riesgo del aprendizaje de la función identidad. En AE si existen más capas ocultas que entradas, existe el riesgo de que el algoritmo solo aprenda la función identidad durante el entrenamiento, es cuando la salida es igual a la entrada, entonces se hace inservible el algoritmo. DAE corrompe aleatoriamente los datos de entrada para reducir este riesgo y aprende a reconstruir los datos originales a partir de los datos corruptos.

Para la corrupción de los datos se aplicará un método que aplica ruido gaussiano aditivo centrado en cero, esto ayuda a mitigar el *overfitting* y es una opción común para el proceso de corrupción para valores reales de entrada. No se aplicará el *dropout* y se utilizará el optimizador SGD. En cuanto a las capas se comporta similar al AE: posee la capa de entrada de datos, cuatro capas para el codificador y dos capas para el decodificador. Secuencialmente se conforman de la siguiente forma:

1. *Input Layer*: La capa de entrada donde se introducen en la red los datos originales.

2.5. PASO 6 Y 7: ELECCIÓN E IMPLEMENTACIÓN DE LOS ALGORITMOS DE MINERÍA DE DATOS

2. *Gaussian Noise*: En esta capa se corrompen los datos y comienza el proceso de codificación.
3. *Dense1 Layer*: Se reducen las dimensiones a 14.
4. *Dense2 Layer*: Se reducen las dimensiones a 7.
5. *Dense3 Layer*: Se reducen las dimensiones a 1. Termina el proceso de codificación
6. *Dense4 Layer*: Se incrementan las dimensiones 7. Comienza la decodificación.
7. *Dense5 Layer*: Se incrementan las dimensiones a 29.
8. *Dense6 Layer o Output Layer*: Se comprimen los datos a una dimensión. Termina la decodificación.

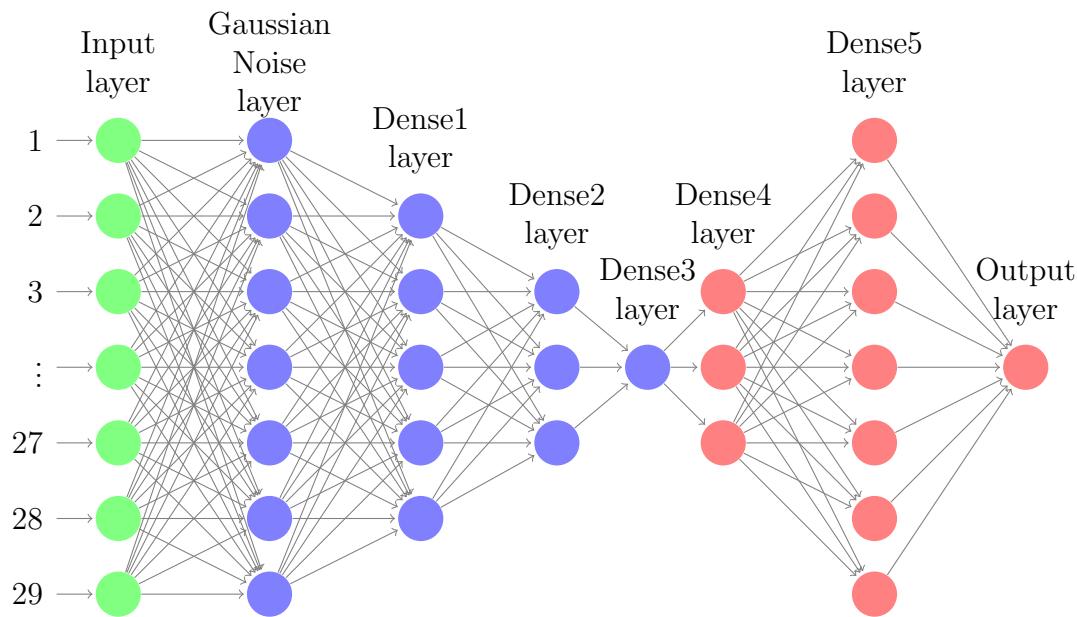


Figura 2.7: Modelo DAE

Convolutional Neural Network (CNN)

La CNN es una red multicapa que consta de capas convolucionales y de reducción alternada, y al final tiene capas de conexión total como una red perceptrón multicapa. Son capaces de detectar características simples y componer en características más complejas hasta detectar lo que se busca. Su principal ventaja es que cada parte de la red se le entrena para realizar una tarea, esto reduce significativamente el número de capas ocultas, por lo que el entrenamiento es más rápido.

Al aplicar CNN para la clasificación se realiza en dos fases: la primera fase es la de extracción de características, la cual está compuesta por neuronas convolucionales y de reducción de muestreo; y la segunda fase es la clasificación final a partir de las características extraídas.

La primera fase se compone de capas alternas de neuronas convolucionales y neuronas de reducción de muestreo. A medida que los datos avanzan por esta fase, van perdiendo dimensionalidad, siendo las neuronas de las capas finales poco sensibles a perturbaciones en los

datos de entrada, al mismo tiempo se activan por características más complejas. Las neuronas son remplazadas por procesadores en matriz que realizan una operación sobre los datos que pasan por ellas en vez de un único valor numérico. Matemáticamente la salida S_j de una neurona j , es una matriz que se calcula por medio de la combinación lineal de las salidas S_i de las neuronas de la capa anterior, cada una de ellas operadas con el núcleo de convolucional K_{ij} correspondiente a esa conexión. Esta cantidad es sumada a una parcialidad b_j y luego se pasa por una función de activación g . Prácticamente la ecuación queda de la siguiente manera:

$$S_j = g(b_j + \sum_i (K_{ij} \otimes Y_i)) \quad (2.6)$$

Las CNN anteriormente utilizaban el proceso de submuestreo para realizar la reducción de muestreo, pero con estudios recientes se han encontrado otras operaciones como *max-pooling*, que encuentra el valor máximo entre una ventana de muestra y pasa este valor como resumen de características sobre esta área. Esto permite que el tamaño de los datos se reduce por un factor igual al tamaño de la ventana de muestra sobre la cual se opera.

Al concluir con las extracciones de características, se procede a la fase de clasificación, gracias a que los datos han sido depurados hasta una serie de características únicas para los datos de entrada. Las neuronas en esta fase funcionan de manera similar a las de un perceptrón multicapa. La salida y_j de una neurona j se calcula a partir de las salidas y_i de las neuronas de la capa anterior cada una de ellas multiplicadas con el peso w_{ij} correspondiente. Esta cantidad es sumada a una parcialidad b_j y luego se pasa por una función de activación g . La ecuación queda de esta forma:

$$y_j = g(b_j + \sum_i w_{ij} y_i) \quad (2.7)$$

Para el proyecto se desarrollará una CNN con dos fases de extracción de características, en cada una se aplicará la convolución, un *BatchNormalization* y un *Dropout*. Luego una fase de clasificación donde se aplicará un *Flatten* para reducir los datos a una dimensión, una capa *Dense* para incrementar las dimensiones a 64 y por último un *Dropout*. Y, por último, una capa de salida; además se aplicará Adam como optimizador. Las capas en orden se describen a continuación:

1. *Input Layer*: Se introducen los datos con dimensión (*None*, 29, 1).
2. *Conv1D-1 Layer*: Comienza la primera fase de extracción de características, introduciendo una capa convolucional que transforma las dimensiones a (*None*, 28, 32).
3. *Batch-Normalization-1 Layer*: Se aplica el regularizador para normalizar los datos.
4. *Dropout-1 Layer*: Se aplica el regularizador para eliminar modelos para evitar el *overfitting*, y culmina la primera fase de extracción de características.
5. *Conv1D-2 Layer*: Se aplica una capa de convolución y comienza la segunda fase de extracción de características. La dimensión de los datos se transforma a (*None*, 27, 64).
6. *Batch-Normalization-2 Layer*: Se aplica el regularizador para normalizar los datos.
7. *Dropout-2 Layer*: Se aplica el regularizador para eliminar modelos para evitar el *overfitting* y termina la segunda fase de extracción.

2.5. PASO 6 Y 7: ELECCIÓN E IMPLEMENTACIÓN DE LOS ALGORITMOS DE MINERÍA DE DATOS

8. *Flatten Layer*: Se descomponen los datos a tener la dimensión (*None*, 1728) y comienza la fase de clasificación.
9. *Dense-1 Layer*: Se comprimen los datos hasta obtener las dimensiones (*None*, 64).
10. *Dropout-3 Layer*: Se eliminan modelos para evitar el *overfitting*.
11. *Dense-2 Layer o Output Layer*: Se comprimen los datos una dimensión obteniendo los datos clasificados y terminando la fase de clasificación.

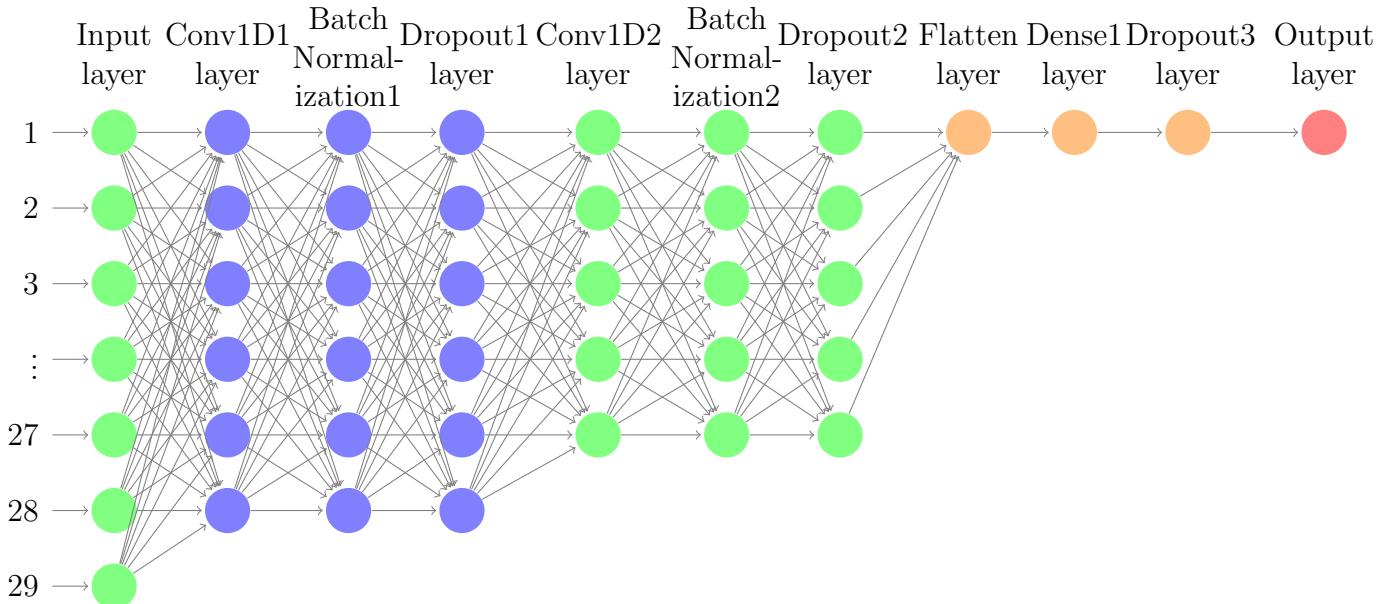


Figura 2.8: Modelo CNN

Backpropagation Neural Network (BPNN)

En las redes neuronales se introducen los datos por la capa de entrada y se propagan hacia adelante atravesando las capas ocultas hasta llegar a la capa de salida, obteniendo el resultado final. Este resultado se compara con los valores reales para calcular el error que se comete, y propagar este error hacia atrás a través de la red neuronal, permitiendo ajustar los pesos durante el proceso de entrenamiento. Uno de los métodos para realizar esta operación es *backpropagation*.

Backpropagation se activa luego de que la información llega a la capa de salida y se mide el error que se ha cometido en esa observación. En la propagación hacia atrás, se van actualizando los pesos de cada conexión neuronal, dependiendo de la responsabilidad del peso actualizado en el error cometido. Este proceso se repite para el conjunto de observaciones, al pasar todas las observaciones por la red neuronal, se completa una época.

Matemáticamente el nodo de una red neuronal se podría representar como a_i^k donde i es el lugar que representa el nodo dentro de una capa y k representa la capa en que se encuentra el nodo. El peso de las conexiones se representa como w_{ji}^k donde i indica el número de la neurona dentro de la capa final de la conexión, e j indica el número de la neurona de la capa inicial de la conexión. Entonces la ecuación para calcular *backpropagation* queda de la siguiente forma:

$$a_i^k = f(u_i^k + \sum_{j=1}^{n_k-1} a_j^{k-1} w_{ji}^{k-1}) \quad (2.8)$$

$$i = 1, \dots, n_k \quad (2.9)$$

El algoritmo BPNN que se utilizará en el proyecto hace uso del optimizador Adam y posee solamente 5 capas, incluyendo la de entrada y salida. A continuación, la secuencia de las capas:

1. *Input layer*: se introducen el conjunto de datos originales sin clasificar.
2. *Dense-1 layer*: se comprimen los datos a 14 dimensiones.
3. *Dense-2 layer*: se comprimen los datos a 2 dimensiones.
4. *Dense-3 layer*: se descomprimen los datos a 14 dimensiones.
5. *Dense-4 layer* o *Output layer*: se comprimen los datos a una dimensión y termina la clasificación de los datos.

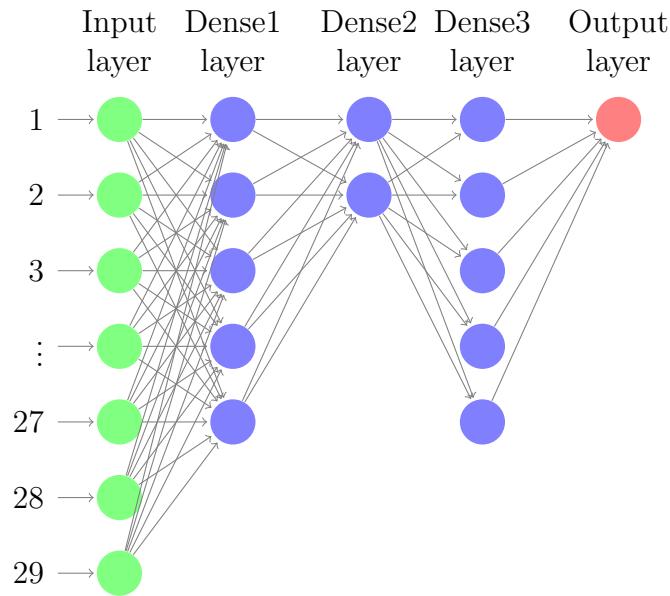


Figura 2.9: Modelo BPNN

Recurrent Neural Network (RNN)

RNN es un modelo supervisado de aprendizaje profundo donde la actual salida depende del valor actual y de las entradas anteriores usando *backpropagation* a través del tiempo t en que se realizan las transacciones. Matemáticamente, se define X_t como los datos de entrada en el instante de tiempo t , H_t como el paso de tiempo de la capa oculta, y y_t como la salida de los datos en el instante de tiempo. En el procesamiento de los datos desde la entrada a la salida intervienen los vectores de peso dependiendo de las capas, para mejor apreciación se muestran en el gráfico siguiente:

2.5. PASO 6 Y 7: ELECCIÓN E IMPLEMENTACIÓN DE LOS ALGORITMOS DE MINERÍA DE DATOS

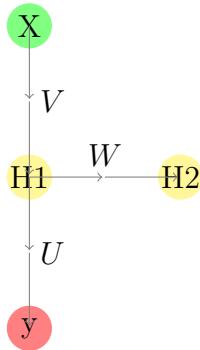


Figura 2.10: Funcionamiento RNN

V es el vector peso para la capa oculta, U es el vector peso para la capa de salida y W es el mismo vector peso para los diferentes instantes de tiempo. Teniendo en cuenta esto, se puede calcular la capa oculta y la salida teniendo en cuenta estas variables y la función de activación σ , para ello se usan las siguientes ecuaciones:

$$H_t = \sigma(U X_t + W H_{t-1}) \quad (2.10)$$

$$y_t = \text{Softmax}(V H_t) \quad (2.11)$$

RNN aplica el algoritmo *backward propagation* para actualizar los pesos, sus fundamentos se exponen en el siguiente algoritmo LSTM.

El algoritmo RNN a aplicar en el proyecto está compuesto con una capa de RNN para el tratamiento de la información teniendo en cuenta los instantes de tiempo. Se Aplicarán tres capas regularizadoras *Dropout* no consecutivas con el objetivo de evitar el *overfitting* y capas *Dense* para redimensionar los datos, además, se aplicará el optimizador Adam. En conclusión, el algoritmo está compuesto por 8 capas divididos en: 1 capa de salida, 1 capa de RNN, 5 capas de clasificación y 1 capa de salida. La arquitectura es la siguiente de forma secuencial:

1. *Input Layer*: Se introducen los datos originales con los pasos de tiempos.
2. *RNN layer*: Se aplica el proceso correspondiente al *backpropagation* con pasos de tiempo.
3. *Dropout-1 layer*: Se aplica el regularizador *Dropout* y comienza la clasificación.
4. *Dense-1 layer*: Se comprimen los datos a 14 dimensiones.
5. *Dropout-2 layer*: Se aplica el nuevamente el regularizador.
6. *Dense-2 layer*: Se vuelve a aplicar la compresión de los datos.
7. *Dropout-3 layer*: Se vuelve a aplicar el regularizador.
8. *Dense-3 layer o Output layer*: Se comprimen los datos a una dimensión terminando con la clasificación de los datos.

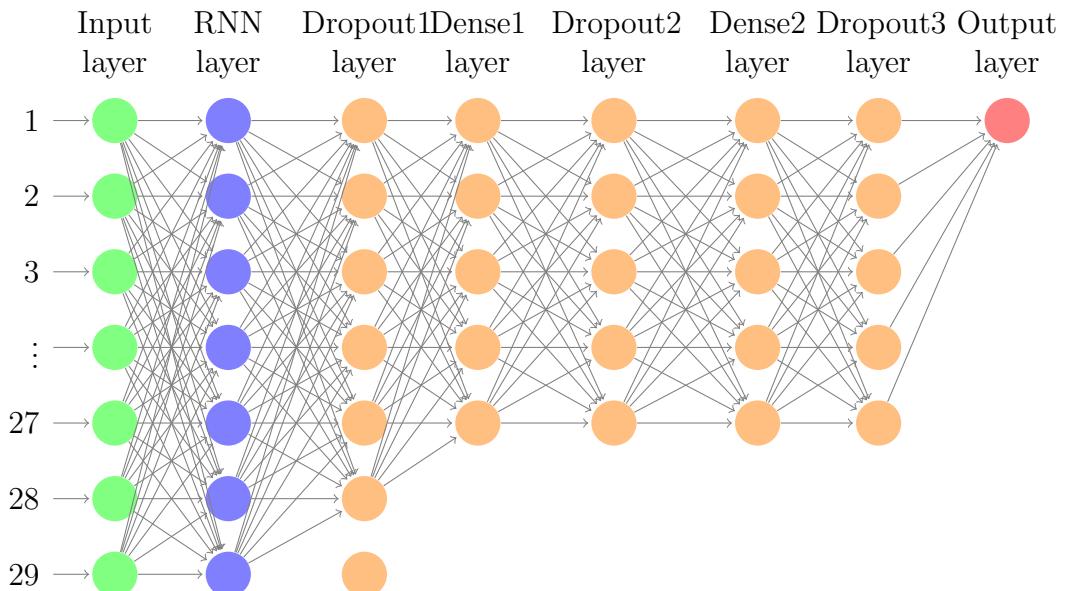


Figura 2.11: Modelo RNN

Long Short Term Memory (LSTM)

LSTM soluciona el problema de los gradientes que desaparecen y explotan de RNN al poseer una mayor memoria para almacenar los vectores de peso. La siguiente figura muestra la arquitectura del funcionamiento del algoritmo:

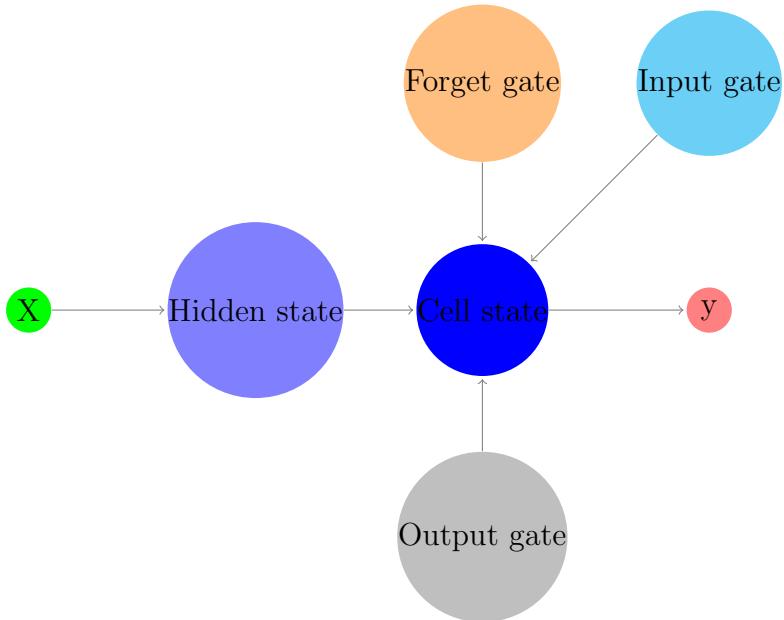


Figura 2.12: Funcionamiento de LSTM

En la figura se denota X como el conjunto de datos originales sin clasificar, este conjunto pasa al estado oculto (*Hidden state*) que es similar a la capa RNN donde se alimenta de las entradas anteriores. Los datos pasan a tres puertas: *Input gate*, *Forget gate* y *Output gate*, las cuales forman parte de la capa de memoria (*Cell state*) que contiene los contextos de las entradas. *Input gate*, decide si es necesario actualizar la memoria anterior de la red teniendo en cuenta a X . *Forget gate*, decide si es necesario eliminar la memoria anterior de la red teniendo en cuenta a X . *Output gate*, debe dar una salida y basada en la entrada actual, la salida de la

2.5. PASO 6 Y 7: ELECCIÓN E IMPLEMENTACIÓN DE LOS ALGORITMOS DE MINERÍA DE DATOS

capa anterior y la celda de memoria persistente.

Teniendo en cuenta que H' es la salida del vector anterior con sus vectores de peso de la capa oculta w' y de la capa de entrada u' . Estos vectores pesos se definen para cada una de las puertas, obteniendo w'_1 y u'_1 para *forget gate*, w'_2 , w'_3 , u'_2 y u'_3 para *input gate*, y w'_4 , u'_4 para *output gate*.

Forget gate

Esta puerta solo ajusta las dos entradas con pesos y se le aplica una función de activación para ajustar los valores de salida en $(0, 1)$. Si el valor de entrada a esta puerta es 0, al ser multiplicado con la celda de memoria, se podrán eliminar los valores con éxito, por ello se llama la puerta de olvido. Se formaliza su ecuación como:

$$F = \sigma(u'_1 X + w'_1 H') \quad (2.12)$$

Input gate

Esta puerta lanza una salida a través de la multiplicación de dos funciones, el resultado de las funciones de activación σ y tan. La responsabilidad de esta puerta es establecer el valor en el estado de la memoria, todos los valores se establecerán, es por ello que se aplica σ que establece el rango de los valores en $(0, 1)$, luego se multiplican por tan, que tiene un rango de valores entre $(-1, 1)$. Teniendo en cuenta este resultado, se establecerán en la memoria los valores que tienen valores que tienden a 1 y los valores que tienden a -1 serán eliminados. Para establecer las ecuaciones se define G como el estado candidato y a I como el estado de entrada.

$$I = \sigma(u'_2 X + w'_2 H') \quad (2.13)$$

$$G = \tan(u'_3 X + w'_3 H') \quad (2.14)$$

Memory state

Una vez obtenidos los valores de *forget gate* e *input gate* se pueden calcular los valores de la memoria actual C haciendo uso también de los valores anteriores de la memoria C' .

$$C = C' F + I G \quad (2.15)$$

Output gate

Esta puerta es básicamente la responsable de encontrar la salida final multiplicando las nuevas entradas y la memoria final. Se vuelve a multiplicar C por tan para que el rango de salida sea $(-1, 1)$ y sea mejor la eliminación de valores si es necesario.

$$O = \tan(u'_4 X + w'_4 H') \quad (2.16)$$

$$H' = O \tan(C) \quad (2.17)$$

$$Output = vH' \quad (2.18)$$

Hasta este momento se obtuvo la salida del *forward propagation*, luego se aplica la técnica del *backward propagation*. Las ecuaciones finales para obtener los nuevos pesos mediante la regla de la cadena para poder disminuir el error absoluto de las soluciones son:

$$E = \frac{(Output - Y)^2}{2} \quad (2.19)$$

$$w_{new} = w_{old} - \frac{\partial E}{\partial w} \quad (2.20)$$

$$v_{new} = v_{old} - \frac{\partial E}{\partial v} \quad (2.21)$$

$$u_{new} = u_{old} - \frac{\partial E}{\partial u} \quad (2.22)$$

Estas ecuaciones se aplican para los diferentes vectores de peso utilizados por las diferentes puertas de la memoria de valores.

El algoritmo LSTM a aplicar en el proyecto está compuesto con una capa LSTM para el tratamiento de la información teniendo en cuenta los instantes de tiempo. Se Aplicarán tres capas regularizadoras *Dropout* no consecutivas con el objetivo de evitar el *overfitting* y capas *Dense* para redimensionar los datos, además, se aplicará el optimizador Adam. En conclusión, el algoritmo está compuesto por 8 capas divididos en: 1 capa de salida, 1 capa de LSTM, 5 capas de clasificación y 1 capa de salida. La arquitectura es la siguiente de forma secuencial:

1. *LSTM layer*: Se aplica el proceso correspondiente al *backpropagation* con pasos de tiempo.
2. *Dropout-1 layer*: Se aplica el regularizador *Dropout* y comienza la clasificación.
3. *Dense-1 layer*: Se comprimen los datos a 14 dimensiones.
4. *Dropout-2 layer*: Se aplica nuevamente el regularizador.
5. *Dense-2 layer*: Se vuelve a aplicar la compresión de los datos.
6. *Dropout-3 layer*: Se vuelve a aplicar el regularizador.
7. *Dense-3 layer o Output layer*: Se comprimen los datos a una dimensión terminando con la clasificación de los datos.

Ya concluida la presentación y fundamentación del funcionamiento y estructura de los algoritmos de DL a utilizar en el proyecto, es posible proceder a la evaluación de los algoritmos. La evaluación y aplicación del conocimiento forman los pasos finales de la metodología aplicada para el desarrollo del proyecto. Para la evaluación de los algoritmos se realizarán 7 experimentos que serán detallados y representados sus resultados en el siguiente capítulo, y finalizando el mismo la aplicación del conocimiento adquirido.

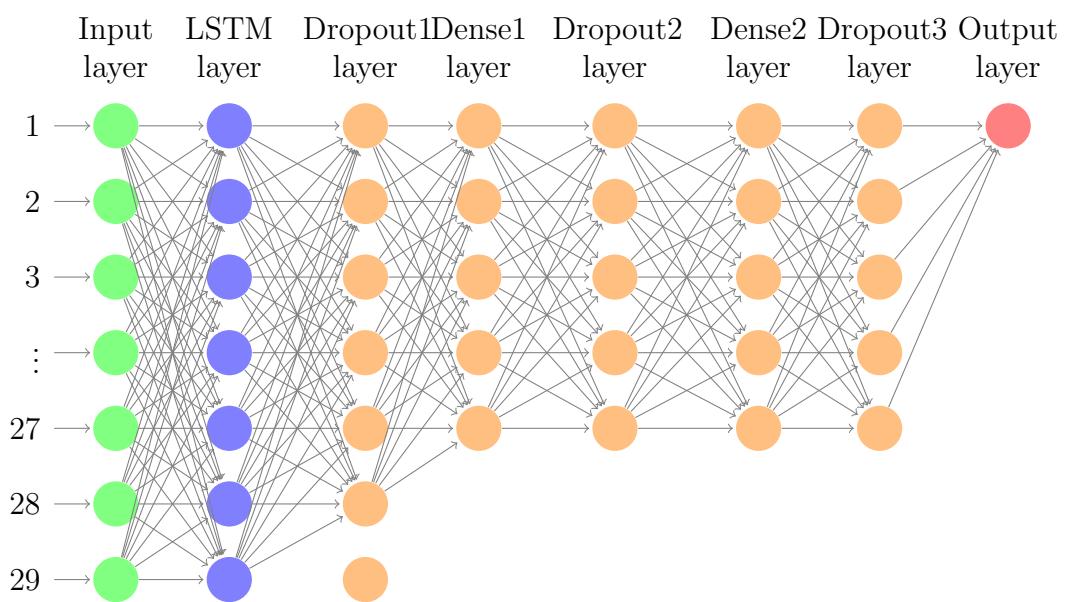


Figura 2.13: Modelo LSTM

Capítulo 3

Evaluación y aplicación de los modelos DL

Este capítulo abarca la evaluación de los modelos de DL seleccionados en los pasos anteriores de la metodología KDD. Para ello están definidas las métricas y algunos parámetros, no obstante, es necesario realizar unas primeras pruebas para terminar de ajustar los parámetros restantes. En las tablas obtenidas de los resultados se definen tres colores representativos, que va a permitir definir una superioridad o inferioridad de los algoritmos, estos colores son:

- **Verde**: representa el valor más alto en la métrica.
- **Amarillo**: representa el segundo mejor valor de la métrica.
- **Rojo**: representa el peor valor de la métrica.

Ahora se procede a realizar los experimentos definidos para evaluar los algoritmos propuestos. Estos experimentos conforman la parte 8 de la metodología, centrada en la evaluación de los modelos seleccionados.

3.1 Experimento 1: Comparación entre la selección de datos para las pruebas

Este primer experimento está basado en buscar la mejor proporción de datos para el entrenamiento y prueba, para ello se usará el algoritmo KNN de ML con los datos desbalanceados.

%Test	Model	Accuracy	AUC	Precision	Recall	F1
10%	KNN IMBALANCE	0.997	0.571	1	0.14285714	0.25
20%	KNN IMBALANCE	0.99775	0.55	1	0.1	0.181818182
30%	KNN IMBALANCE	0.996833333	0.525	1	0.05	0.09523809523809523
40%	KNN IMBALANCE	0.996375	0.5	0	0	0
50%	KNN IMBALANCE	0.9971	0.5	0	0	0

Tabla 3.2: Puntuación de tamaños de prueba en KNN desbalanceado.

3.2. EXPERIMENTO 2: COMPARACIÓN ENTRE LOS MODELOS DE ML POR ESTRATEGIAS PARA SOLUCIONAR AL DESBALANCE DE LA INFORMACIÓN

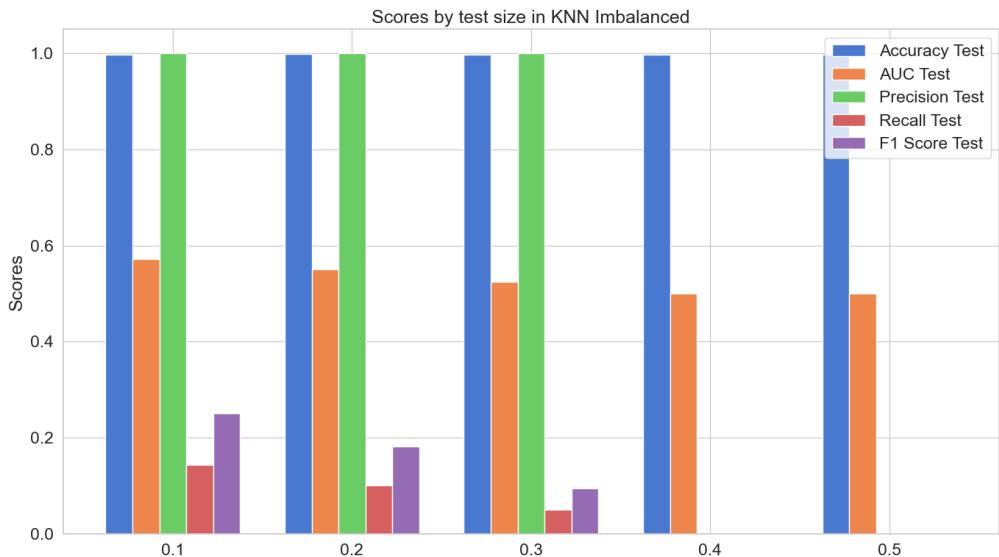


Figura 3.1: Puntuaciones por tamaños de prueba en KNN desbalanceado.

Observaciones:

1. Los peores resultados se obtienen cuando el por ciento de datos para la prueba es de 40% y 50% del conjunto de datos.
2. El mejor resultado lo obtiene cuando el por ciento de datos para las pruebas es de un 10% del conjunto de datos.
3. La mejor exactitud de predicción es cuando se realiza la prueba con un 20% del conjunto de datos.

Por lo tanto, se decide tomar un 20% de datos para realizar las pruebas, porque posee mayor exactitud y posee el segundo mejor resultado de *F1 Score*, ya que con un 10% se pierde exactitud en las predicciones a pesar de tener el mejor *F1 Score*.

3.2 Experimento 2: Comparación entre los modelos de ML por estrategias para solucionar al desbalance de la información

El objetivo de este experimento es seleccionar los mejores modelos ML por estrategias. Para ello se utilizarán los modelos DT, GNB, KNN, LR, MLP, RF y XGBOOST, y cada modelo se le aplicarán las estrategias con datos desbalanceados, UnderSampler, OverSampler, SMOTE y ADASYN. Una vez definido los modelos y estrategias, se procede a realizar el experimento y obtener los resultados.

Model	Accuracy	AUC	Precision	Recall	F1
RF IMABALANCED	0.99775	0.681567	0.666667	0.363636	0.470588
XGBOOST IMBALANCED	0.9975	0.681442	0.571429	0.363636	0.444444

Model	Accuracy	AUC	Precision	Recall	F1
KNN IMBALANCE	0.99775	0.590909	1	0.181818	0.307692
DT IMBALANCED	0.99425	0.679813	0.2	0.363636	0.258065
RF ADASYN	0.997	0.583083	0.5	0.166667	0.25
RF OVERSAMPLE	0.997	0.599499	0.333333	0.2	0.25
XGBOOST	0.9965	0.599248	0.25	0.2	0.222222
DT OVERSAMPLE	0.99575	0.598872	0.181818	0.2	0.190476
RF SMOTE	0.99675	0.535714	1	0.071429	0.133333
XGBOOST ADASYN	0.9925	0.580826	0.090909	0.166667	0.117647
NB IMBALANCED	0.98	0.717997	0.063291	0.454545	0.111111
XGBOOST SMOTE	0.99425	0.53446	0.090909	0.071429	0.08
LR ADASYN	0.986	0.577566	0.041667	0.166667	0.066667
NB UNDERSAMPLE	0.96425	0.710101	0.035211	0.454545	0.065359
LR SMOTE	0.98525	0.565533	0.040816	0.142857	0.063492
LR OVERSAMPLE	0.9845	0.593233	0.035714	0.2	0.060606
NB ADASYN	0.969	0.610582	0.025424	0.25	0.046154
DT SMOTE	0.98575	0.530195	0.022222	0.071429	0.033898
MLP ADASYN	0.95275	0.602432	0.016393	0.25	0.030769
DT ADASYN	0.9825	0.534269	0.016667	0.083333	0.027778
NB OVERSAMPLE	0.96275	0.582331	0.013986	0.2	0.026144
MLP SMOTE	0.95575	0.550731	0.011976	0.142857	0.022099
NB SMOTE	0.972	0.523296	0.01	0.071429	0.017544
KNN SMOTE	0.76225	0.595997	0.006322	0.428571	0.012461
RF UNDERSAMPLE	0.724	0.634973	0.00543	0.545455	0.010753
KNN ADASYN	0.806	0.570378	0.005181	0.333333	0.010204
DT UNDERSAMPLE	0.586	0.611112	0.004219	0.636364	0.008383
XGBOOST UNDERSAMPLE	0.32725	0.572039	0.003336	0.818182	0.006645
KNN UNDERSAMPLE	0.57375	0.514312	0.002934	0.454545	0.005831
MLP UNDERSAMPLE	0.03425	0.515793	0.002839	1	0.005663
LR UNDERSAMPLE	0.00275	0.5	0.00275	1	0.005485
MLP OVERSAMPLE	0.155	0.376942	0.001774	0.6	0.003538
MLP IMBALANCE	0.99675	0.499749	0	0	0
KNN OVERSAMPLE	0.9915	0.496992	0	0	0
LR IMBALANCED	0.99725	0.5	0	0	0

Tabla 3.3: Puntuaciones de las pruebas de los modelos ML por estrategias.

Basados en la tabla 3.3 y en los anexos 3.5, 3.6, 3.7, 3.8 y 3.9, se pueden obtener las siguientes observaciones:

- El modelo con mejor resultado es RF con datos desbalanceados.
- El segundo mejor resultado lo obtiene el modelo XGBOOST con datos desbalanceados.
- El peor resultado lo obtiene el modelo LR con datos desbalanceados.
- El mejor resultado en el modelo DT se obtiene con la estrategia OverSampler.
- El mejor resultado en el modelo KNN se obtiene con los datos desbalanceados.

3.3. EXPERIMENTO 3: COMPARACIÓN EN EL ORDEN DE LOS DROPOUT CON DATOS DESBALANCEADOS

- El mejor resultado en el modelo LR se obtiene con ADASYN.
- El mejor resultado en el modelo MPLClassifier se obtiene con ADASYN.
- El mejor resultado en el modelo NB se obtiene con datos desbalanceados.
- El mejor resultado en el modelo RF se obtiene con datos desbalanceados.
- El mejor resultado en el modelo XGBOOST se obtiene con datos desbalanceados.
- El mejor resultado con datos desbalanceados lo obtiene el modelo RF.
- El mejor resultado con UnderSampler lo obtiene el modelo NB.
- El mejor resultado con OverSampler lo obtiene el modelo RF.
- El mejor resultado con SMOTE lo obtiene el modelo RF.
- El mejor resultado con ADASYN lo obtiene el modelo RF.
- El modelo RF presenta superioridad con respecto a los demás modelos.

Con estas observaciones son suficientes para decidir los modelos que representarán a los modelos ML. Se escoge el modelo RF con datos desbalanceados, OverSampler, SMOTE y ADASYN, y el modelo NB con UnderSampler para próximos experimentos como los mejores modelos ML.

3.3 Experimento 3: Comparación en el orden de los dropout con datos desbalanceados

En este experimento se analizará el orden los *dropout* de los algoritmos CNN y RNN, donde el máximo posible recomendado por la librería es 0.5. Se realizarán las pruebas con datos desbalanceados y 1 *epoch* para los órdenes de *dropout* mostrados en la tabla.

Dropout	Model	Accuracy	AUC	Precision	Recall	F1
0.5-0.4-0.3	CNN IMBALANCE	0.999	0.714286	0.999001	0.999	0.9988
0.3-0.4-0.5	CNN IMBALANCE	0.999	0.714286	0.999001	0.999	0.9988
0.5-0.5-0.5	CNN IMBALANCE	0.999	0.714286	0.999001	0.999	0.9988
0.4-0.4-0.4	CNN IMBALANCE	0.99875	0.71416	0.998563	0.99875	0.99858
0.3-0.3-0.3	CNN IMBALANCE	0.99875	0.71416	0.998563	0.99875	0.99858
0.2-0.2-0.2	CNN IMBALANCE	0.99875	0.71416	0.998563	0.99875	0.99858
0.2-0.2-0.5	CNN IMBALANCE	0.99875	0.71416	0.998563	0.99875	0.99858
0.5-0.2-0.2	CNN IMBALANCE	0.99875	0.71416	0.998563	0.99875	0.99858
0.5-0.4-0.3	RNN IMBALANCE	0.99825	0.5	0.996503	0.99825	0.997376
0.3-0.4-0.5	RNN IMBALANCE	0.99825	0.5	0.996503	0.99825	0.997376
0.5-0.5-0.5	RNN IMBALANCE	0.99825	0.5	0.996503	0.99825	0.997376
0.4-0.4-0.4	RNN IMBALANCE	0.99825	0.5	0.996503	0.99825	0.997376
0.3-0.3-0.3	RNN IMBALANCE	0.99825	0.5	0.996503	0.99825	0.997376
0.2-0.2-0.2	RNN IMBALANCE	0.99825	0.5	0.996503	0.99825	0.997376

Dropout	Model	Accuracy	AUC	Precision	Recall	F1
0.2-0.2-0.5	RNN IMBALANCE	0.99825	0.5	0.996503	0.99825	0.997376
0.5-0.2-0.2	RNN IMBALANCE	0.99825	0.5	0.996503	0.99825	0.997376

Tabla 3.4: Puntuación en las pruebas del modelo CNN con datos desbalanceados para los órdenes de *dropout*.

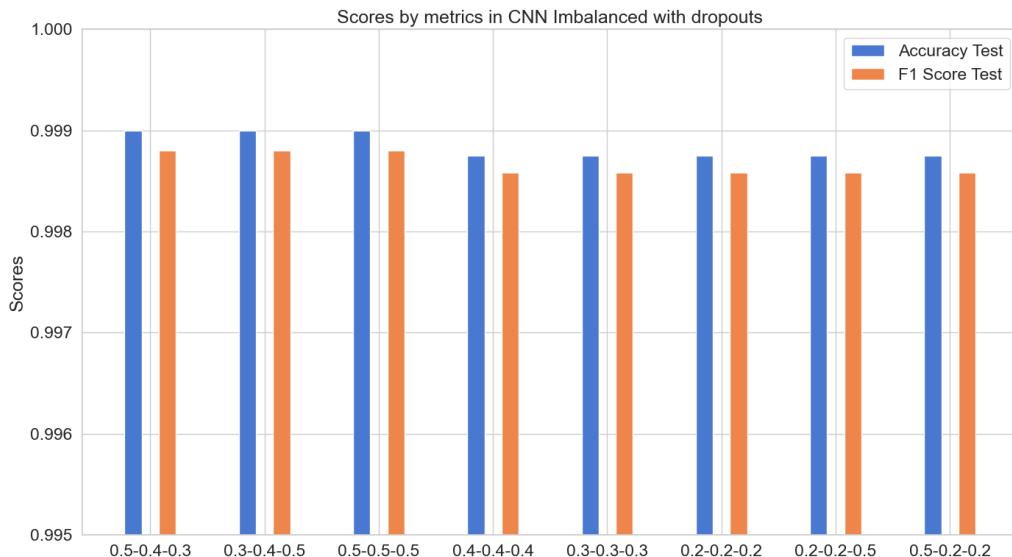


Figura 3.2: Puntuaciones en las pruebas del modelo CNN con datos desbalanceados por dropouts.

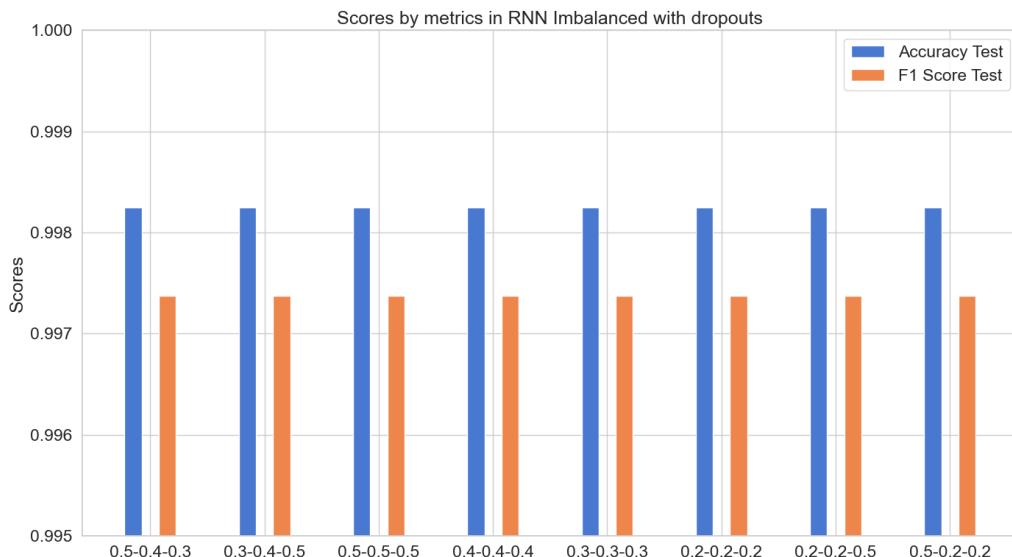


Figura 3.3: Puntuaciones en las pruebas del modelo RNN con datos desbalanceados por dropouts.

Basados en la tabla 3.4 y en las figuras 3.2 y 3.3, se obtienen las siguientes observaciones generales:

- Los mejores resultados se obtienen con CNN con los dropouts 0.5-0.4-0.3, 0.3-0.4-0.5 y 0.5-0.5-0.5.

3.4. EXPERIMENTO 4: COMPARACIÓN EN LOS MODELOS DL TENIENDO EN CUENTA LOS RESULTADOS DE LAS PRUEBAS.

- Los peores resultados se obtienen con RNN, siendo los resultados indistinguibles con cualquier orden de dropout.
- El peor resultado con CNN es con los restantes órdenes de dropouts, siendo sus resultados iguales.
- Los CNN dominan a RNN en cualquier orden de dropout.

Teniendo en cuenta esto, se decide usar de entre los 3 mejores dropouts, el orden 0.5-0.4-0.3 para los próximos experimentos

3.4 Experimento 4: Comparación en los modelos DL teniendo en cuenta los resultados de las pruebas.

Este experimento está orientado a seleccionar los mejores modelos DL teniendo en cuenta dos factores: la cantidad de *epochs* de entrenamiento y la estrategia aplicada para la solución al desbalance de la información. Para ello este experimento se dividirá en dos partes: primero, se analizarán todos los modelos por *epochs* y con la misma estrategia aplicada, segundo, se analizarán todas las estrategias y *epoch* por modelo. Luego de obtener las observaciones por cada una de las dos partes anteriores, se procederá a obtener las observaciones generales y junto a ello, los mejores modelos en cuanto a las pruebas.

Para el experimento se tiene en cuenta que: las cantidades de *epochs* son 1, 20, 50 y 100, las estrategias son datos desbalanceados, UnderSampler, OverSampler, SMOTE y ADASYN, y los modelos a analizar son CNN, AE, DAE, RNN, LSTM y BPNN. Una vez definidas las bases del experimento, se proceden a realizar el experimento y analizarlo por cada una de las partes.

3.4.1 Primera parte: Modelos por epochs con la misma estrategia.

Los resultados obtenidos se muestran a continuación, para ello se dividirá por estrategias:

Datos desbalanceados

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
1	CNN IMBALANCE	0.99675	0.56666667	0.99676057	0.99675	0.99551
50	CNN IMBALANCE	0.99675	0.56666667	0.99676057	0.99675	0.99551
20	CNN IMBALANCE	0.9965	0.5665412	0.99550976	0.9965	0.99533596
100	CNN IMBALANCE	0.99625	0.56641573	0.99488395	0.99625	0.99516706
1	BPNN IMBALANCE	0.99625	0.5	0.99626406	0.99625	0.99437852
20	BPNN IMBALANCE	0.99625	0.5	0.99626406	0.99625	0.99437852
50	BPNN IMBALANCE	0.99625	0.5	0.99626406	0.99625	0.99437852
100	BPNN IMBALANCE	0.99625	0.5	0.99626406	0.99625	0.99437852
1	AE IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
1	DAE IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
1	RNN IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
1	LSTM IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
20	AE IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
20	DAE IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
20	RNN IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
20	LSTM IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
50	AE IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
50	DAE IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
50	RNN IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
50	LSTM IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	AE IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	DAE IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	RNN IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	LSTM IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852

Tabla 3.5: Puntuaciones en las pruebas de los modelos DL con datos desbalanceados en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.5 se obtienen las siguientes observaciones:

- El mejor modelo es CNN con 1 y 50 *epochs* con los mejores resultados.
- El segundo mejor modelo es CNN con 20 *epochs* con los segundos mejores resultados a excepción de la precisión.
- Los peores modelos son AE, DAE, RNN y LSTM con los resultados idénticos y los peores.
- La superioridad de modelos sin tener en cuenta la cantidad de *epochs* es: CNN, BPNN y el resto en tercer lugar.
- Los resultados del modelo BPNN se comportan igual sin importar la cantidad de *epochs*.

UnderSampler

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
1	BPNN UNDERSAMPLE	0.99625	0.5	0.99626406	0.99625	0.99437852
20	BPNN UNDERSAMPLE	0.99625	0.5	0.99626406	0.99625	0.99437852
50	BPNN UNDERSAMPLE	0.99625	0.5	0.99626406	0.99625	0.99437852
100	BPNN UNDERSAMPLE	0.99625	0.5	0.99626406	0.99625	0.99437852
1	AE UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
1	DAE UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
1	RNN UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
1	LSTM UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
20	AE UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
20	DAE UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
20	RNN UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
20	LSTM UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852

3.4. EXPERIMENTO 4: COMPARACIÓN EN LOS MODELOS DL TENIENDO EN CUENTA LOS RES

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
50	AE UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
50	DAE UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
50	RNN UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
50	LSTM UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	AE UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	DAE UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	RNN UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	LSTM UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	CNN UNDERSAMPLE	0.76675	0.61727311	0.99368038	0.76675	0.86451737
50	CNN UNDERSAMPLE	0.53675	0.5350481	0.99301653	0.53675	0.69518094
20	CNN UNDERSAMPLE	0.121	0.45922208	0.98997066	0.121	0.21089089
1	CNN UNDERSAMPLE	0.0765	0.53651192	0.99626517	0.0765	0.13562832

Tabla 3.6: Puntuaciones en las pruebas de los modelos DL con UnderSampler en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.6 se obtienen las siguientes observaciones:

- El mejor modelo es BPNN en 1, 20, 50 y 100 con los mejores resultados a excepción de la precisión con los segundos mejores resultados y *AUC Score*.
- Los segundos mejores modelos son AE, DAE, RNN y LSTM en 1, 20, 50 y 100 *epochs* con resultados idénticos a BPNN a diferencia de la precisión donde obtiene resultados más bajos.
- El peor modelo es CNN con 1 *epoch* con los peores resultados a excepción de *AUC Score* con el segundo mejor resultado y la precisión con el mejor resultado.
- Los modelos AE, DAE, RNN y LSTM no presentan comportamientos diferentes en cuanto a los resultados sin importar el número de *epochs*.

OverSampler

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
1	BPNN OVERSAMPLE	0.99575	0.5	0.99576806	0.99575	0.99362953
20	BPNN OVERSAMPLE	0.99575	0.5	0.99576806	0.99575	0.99362953
50	BPNN OVERSAMPLE	0.99575	0.5	0.99576806	0.99575	0.99362953
100	BPNN OVERSAMPLE	0.99575	0.5	0.99576806	0.99575	0.99362953
1	AE OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
1	DAE OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
1	RNN OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
1	LSTM OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
20	AE OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
20	DAE OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
20	RNN OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
20	LSTM OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
50	AE OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
50	DAE OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
50	RNN OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
50	LSTM OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
100	AE OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
100	DAE OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
100	RNN OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
100	LSTM OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
100	CNN OVERSAMPLE	0.9875	0.70086101	0.99386401	0.9875	0.99040606
50	CNN OVERSAMPLE	0.9865	0.70035888	0.99381181	0.9865	0.98984607
20	CNN OVERSAMPLE	0.98475	0.69948014	0.99373693	0.98475	0.98887889
1	CNN OVERSAMPLE	0.93125	0.7019022	0.99346889	0.93125	0.9604627

Tabla 3.7: Puntuaciones en las pruebas de los modelos DL con OverSampler en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.7 se obtienen las siguientes observaciones:

- El mejor modelo es BPNN en 1, 20, 50 y 100 *epochs* con resultados idénticos y los mejores a excepción de *AUC Score* que obtiene el peor resultado.
- Los segundos mejores modelos son AE, DAE, RNN y LSTM con resultados idénticos a BPNN a excepción de la precisión donde obtienen los peores resultados.
- El peor modelo es CNN en 1 *epoch* donde obtiene los peores resultados a excepción de *AUC Score* con el mejor resultado y la precisión.
- Los resultados de los modelos AE, DAE, RNN y LSTM se comportan iguales sin importar la cantidad de *epochs*.

SMOTE

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
1	BPNN SMOTE	0.9955	0.5	0.99552025	0.9955	0.99325507
20	BPNN SMOTE	0.9955	0.5	0.99552025	0.9955	0.99325507
50	BPNN SMOTE	0.9955	0.5	0.99552025	0.9955	0.99325507
100	BPNN SMOTE	0.9955	0.5	0.99552025	0.9955	0.99325507
1	AE SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
1	DAE SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
1	RNN SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
1	LSTM SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
20	AE SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
20	DAE SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
20	RNN SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
20	LSTM SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
50	AE SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507

3.4. EXPERIMENTO 4: COMPARACIÓN EN LOS MODELOS DL TENIENDO EN CUENTA LOS RES

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
50	DAE SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
50	RNN SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
50	LSTM SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
100	AE SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
100	DAE SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
100	RNN SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
100	LSTM SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
50	CNN SMOTE	0.99225	0.60897651	0.9928545	0.99225	0.99254662
100	CNN SMOTE	0.99125	0.60847425	0.99271384	0.99125	0.99195836
20	CNN SMOTE	0.98975	0.60772085	0.99256918	0.98975	0.99110131
1	CNN SMOTE	0.39575	0.55824823	0.99236381	0.39575	0.56255824

Tabla 3.8: Puntuaciones en las pruebas de los modelos DL con SMOTE en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.8 se obtienen las siguientes observaciones:

- El mejor modelo es BPNN en 1, 20, 50 y 100 *epochs*, con los mejores resultados a excepción del *AUC Score* que obtiene el peor resultado.
- Los segundos mejores modelos son AE, DAE, RNN y LSTM con 1, 20, 50 y 100 *epochs* con resultados idénticos a BPNN a excepción de la precisión donde obtienen los peores resultados.
- El peor modelo es CNN en 1 *epoch* con los peores resultados a excepción de *AUC Score* y la precisión.
- Los modelos AE, DAE, RNN y LSTM tienen resultados iguales sin importar el número de *epochs*.

ADASYN

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
1	BPNN ADASYN	0.99525	0.5	0.99527256	0.99525	0.99288065
20	BPNN ADASYN	0.99525	0.5	0.99527256	0.99525	0.99288065
50	BPNN ADASYN	0.99525	0.5	0.99527256	0.99525	0.99288065
100	BPNN ADASYN	0.99525	0.5	0.99527256	0.99525	0.99288065
1	AE ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
1	DAE ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
1	RNN ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
1	LSTM ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
20	AE ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
20	DAE ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
20	RNN ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
20	LSTM ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
50	AE ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
50	DAE ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
50	RNN ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
50	LSTM ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
100	AE ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
100	DAE ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
100	RNN ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
100	LSTM ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
100	CNN ADASYN	0.99125	0.62894142	0.9926573	0.99125	0.99192671
50	CNN ADASYN	0.99	0.62831344	0.99250555	0.99	0.99119245
20	CNN ADASYN	0.97925	0.62291278	0.62291278	0.97925	0.98531359
1	CNN ADASYN	0.38025	0.53150491	0.99133065	0.38025	0.54630481

Tabla 3.9: Puntuaciones en las pruebas de los modelos DL con ADASYN en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.9 se obtienen las siguientes observaciones:

- El mejor modelo es BPNN en 1, 20, 50 y 100 *epochs*, con los mejores resultados a excepción del *AUC Score* que obtiene el peor resultado.
- Los segundos mejores modelos son AE, DAE, RNN y LSTM con 1, 20, 50 y 100 *epochs* con resultados idénticos a BPNN a excepción de la precisión donde obtienen los peores resultados.
- El peor modelo es CNN en 1 *epoch* con los peores resultados a excepción de *AUC Score* y la precisión.
- Los modelos AE, DAE, RNN y LSTM tienen resultados iguales sin importar el número de *epochs*.

3.4.2 Análisis de la primera parte

Basados en las comparaciones realizadas en la primera parte se puede llegar a la conclusión de que:

- Para datos desbalanceados el mejor modelo es CNN, mientras que los demás modelos tienen resultados idénticos a pesar de que BPNN tiene la precisión mejor que el resto.
- Para UnderSampler, OverSampler, SMOTE y ADASYN, el mejor modelo es BPNN, aunque posea resultados similares a AE, DAE, RNN y LSTM, este posee mayor precisión en los resultados.
- Por cada estrategia aplicada, los modelos AE, DAE, RNN y LSTM obtienen resultados similares sin importar la cantidad de *epochs*.

Teniendo en cuenta esto, se decide utilizar para próximos experimentos el modelo CNN con datos desbalanceados y el modelo BPNN con UnderSampler, OverSampler, SMOTE y ADASYN.

3.4. EXPERIMENTO 4: COMPARACIÓN EN LOS MODELOS DL TENIENDO EN CUENTA LOS RESULTADOS

3.4.3 Segunda parte: Estrategias y epochs por modelos

Con los resultados obtenidos se pueden comparar las estrategias de los modelos por cada *epoch* de forma tal de que se pueda obtener la mejor estrategia y *epoch* por modelo.

CNN

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
50	CNN IMBALANCE	0.99675	0.56666667	0.99676057	0.99675	0.99551
1	CNN IMBALANCE	0.99675	0.56666667	0.99676057	0.99675	0.99551
20	CNN IMBALANCE	0.9965	0.5665412	0.99550976	0.9965	0.99533596
100	CNN IMBALANCE	0.99625	0.56641573	0.99488395	0.99625	0.99516706
50	CNN SMOTE	0.99225	0.60897651	0.9928545	0.99225	0.99254662
100	CNN SMOTE	0.99125	0.60847425	0.99271384	0.99125	0.99195836
100	CNN ADASYN	0.99125	0.62894142	0.9926573	0.99125	0.99192671
50	CNN ADASYN	0.99	0.62831344	0.99250555	0.99	0.99119245
20	CNN SMOTE	0.98975	0.60772085	0.99256918	0.98975	0.99110131
100	CNN OVERSAMPLE	0.9875	0.70086101	0.99386401	0.9875	0.99040606
50	CNN OVERSAMPLE	0.9865	0.70035888	0.99381181	0.9865	0.98984607
20	CNN OVERSAMPLE	0.98475	0.69948014	0.99373693	0.98475	0.98887889
20	CNN ADASYN	0.97925	0.62291278	0.99202191	0.97925	0.98531359
1	CNN OVERSAMPLE	0.93125	0.7019022	0.99346889	0.93125	0.9604627
100	CNN UNDERSAMPLE	0.76675	0.61727311	0.99368038	0.76675	0.86451737
50	CNN UNDERSAMPLE	0.53675	0.5350481	0.99301653	0.53675	0.69518094
1	CNN SMOTE	0.39575	0.55824823	0.99236381	0.39575	0.56255824
1	CNN ADASYN	0.38025	0.53150491	0.99133065	0.38025	0.54630481
20	CNN UNDERSAMPLE	0.121	0.45922208	0.98997066	0.121	0.21089089
1	CNN UNDERSAMPLE	0.0765	0.53651192	0.99626517	0.0765	0.13562832

Tabla 3.10: Puntuaciones en las pruebas del modelo CNN por estrategias en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.10 se obtienen las siguientes observaciones:

- La mejor estrategia es con datos desbalanceados en 1 y 50 *epochs* con resultados idénticos.
- La segunda mejor estrategia es con datos desbalanceados en 20 *epochs*.
- La peor estrategia es UnderSampler en 1 *epoch*.
- El orden de las tres mejores estrategias sin tener en cuenta la cantidad de *epochs* es: datos desbalanceados, SMOTE y ADASYN.
- El mejor resultado del modelo CNN con datos desbalanceados se obtiene en 1 y 50 *epochs*.
- El mejor resultado del modelo CNN con UnderSampler, OverSampler y ADASYN se obtiene en 100 *epochs*.
- El mejor resultado del modelo CNN con SMOTE se obtiene en 50 *epochs*.

AE

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
20	AE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
1	AE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
50	AE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
100	AE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
100	AE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
50	AE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
1	AE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
20	AE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
100	AE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
1	AE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
50	AE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
20	AE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
100	AE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
20	AE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
50	AE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
1	AE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
100	AE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
1	AE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
50	AE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
20	AE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881

Tabla 3.11: Puntuaciones en las pruebas del modelo AE por estrategias en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.11 se obtienen las siguientes observaciones:

- Las mejores estrategias son con datos desbalanceados y UnderSampler sin importar la cantidad de *epochs*.
- La segunda mejor estrategia es con OverSampler sin importar la cantidad de *epochs*.
- La peor estrategia es con ADASYN sin importar la cantidad de *epochs*.
- La cantidad de *epochs* son irrelevantes en cuanto a los resultados obtenidos por estrategias.

Teniendo en cuenta los resultados se pueden obtener las mismas observaciones en los modelos DAE, RNN, LSTM y BPNN, aunque los resultados difieren únicamente en la precisión en el modelo BPNN donde es mayor. Los resultados para 20, 50 y 100 *epochs* siguen el mismo comportamiento y se pueden obtener los mismos resultados.

Se puede apreciar que los resultados son indiferentes al número de *epochs*, por lo tanto, para el modelo AE se puede utilizar cualquier cantidad de *epochs*, ya que se obtienen los mismos resultados. De forma similar ocurre con los modelos DAE, RNN, LSTM y BPNN, obteniendo los mismos resultados y observaciones a excepción de BPNN donde los resultados de precisión son mejores.

3.4. EXPERIMENTO 4: COMPARACIÓN EN LOS MODELOS DL TENIENDO EN CUENTA LOS RESULTADOS

3.4.4 Análisis de la segunda parte

Las comparaciones anteriores y los resultados expuestos en el anexo 3.27 permiten la obtención de las siguientes observaciones en general:

- La mejor estrategia para el modelo CNN es con datos desbalanceados en 1 y 50 *epochs* con los mejores resultados.
- Las mejores estrategias para los modelos AE, DAE, RNN, LSTM y BPNN son con datos desbalanceados y UnderSampler sin importar la cantidad de *epochs*.
- La peor estrategia para el modelo CNN es con UnderSampler en 1 *epoch*.
- La peor estrategia en los modelos AE; DAE, RNN, LSTM y BPNN es con ADASYN sin importar la cantidad de *epochs*.
- Los mejores resultados del modelo CNN con datos desbalanceados se obtienen en 1 y 50 *epochs*, mientras que con UnderSampler, OverSampler y SMOTE se obtienen en 100 *epochs* y con SMOTE en 50 *epochs*.
- Para los modelos a excepción de CNN, no se aprecian diferencias en los resultados sin importar la cantidad de *epochs*, es decir, los resultados se mantienen constantes sin importar el número de *epochs*.

Teniendo en cuenta estas observaciones, se decide utilizar el modelo CNN con datos desbalanceados en 1 y 50 *epochs*, con OverSampler y ADASYN en 100 *epochs*, con SMOTE en 50 *epochs*, y los modelos AE; DAE; RNN, LSTM y BPNN con cualquier estrategia y número de *epochs* para próximos experimentos.

3.4.5 Análisis general del experimento 4

Para una mayor comprensión de los resultados y observaciones obtenidas, se obtiene la siguiente matriz de los mejores modelos por estrategias y la cantidad de *epochs* donde obtienen mejores resultados:

Estrategias					
	Imbalanced	UnderSampler	OverSampler	SMOTE	ADASYN
CNN	1 y 50	-	-	-	-
AE	-	-	-	-	-
DAE	-	-	-	-	-
RNN	-	-	-	-	-
LSTM	-	-	-	-	-
BPNN	-	100	100	100	100

Tabla 3.12: Mejores modelos DL por estrategia.

Teniendo en cuenta la matriz anterior, los mejores modelos son CNN con datos desbalanceados en 1 y 50 *epochs*, y el modelo BPNN con UnderSampler, OverSampler, SMOTE y ADASYN en 100 *epochs*.

3.5 Experimento 5: Comparación de los modelos DL teniendo en cuenta los resultados del entrenamiento

Para encontrar de otra posible diferencia entre los modelos, se realiza este experimento que se realiza de conjunto con el experimento 4. En este caso el objetivo es, al igual que en el anterior, seleccionar los mejores modelo DL teniendo en cuenta los mismos factores, pero la gran diferencia es que se centra en el entrenamiento sin tener en cuenta las pruebas. Por ello, el experimento se divide de igual manera en dos partes y concluirá con la obtención de los mejores modelos en cuanto al entrenamiento.

Para el experimento se tiene en cuenta las cantidades 1, 20, 50 y 100 de *epochs*, las estrategias datos desbalanceados, UnderSampler, OverSampler, SMOTE y ADASYN, y los modelos a comparar son CNN, AE, DAE, RNN, LSTM y BPNN. Ahora se procede a obtener los resultados de los entrenamientos y sus comparaciones en cada una de las partes.

3.5.1 Primera parte: Modelos por epochs con la misma estrategia

El resultado que se espera de esta parte es el mejor modelo por estrategia en cada uno de los *epochs*. Los resultados y observaciones obtenidas se muestran a continuación.

Datos desbalanceados

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	CNN IMBALANCE	0.998749971	0.17832166	0.17016315	0.17132866
50	CNN IMBALANCE	0.998187482	0.146853134	0.12820512	0.13403264
20	CNN IMBALANCE	0.997562528	0.1048951	0.08974358	0.09440558
100	BPNN IMBALANCE	0.998687506	0.057999995	0.057	0.05733333
50	BPNN IMBALANCE	0.998437524	0.045999996	0.046	0.046
20	BPNN IMBALANCE	0.99849999	0.045999996	0.044	0.04466666
1	CNN IMBALANCE	0.98724997	0.040209785	0.04079254	0.04009324
50	LSTM IMBALANCE	0.9979375	0.031999998	0.032	0.032
100	LSTM IMBALANCE	0.99781251	0.029999997	0.027	0.028
100	RNN IMBALANCE	0.997437477	0.021999998	0.02	0.02066666
20	LSTM IMBALANCE	0.997562528	0.019999998	0.02	0.02
20	RNN IMBALANCE	0.997500002	0.017999997	0.017	0.01733333
1	RNN IMBALANCE	0.968625009	0.009999999	0.01	0.01
50	RNN IMBALANCE	0.99712503	0.007999999	0.007	0.00733333
1	LSTM IMBALANCE	0.99150002	0.004	0.004	0.004
100	AE IMBALANCE	0.996937513	0	0	0
100	DAE IMBALANCE	0.996937513	0	0	0
50	AE IMBALANCE	0.996937513	0	0	0
50	DAE IMBALANCE	0.996937513	0	0	0
20	AE IMBALANCE	0.996937513	0	0	0
20	DAE IMBALANCE	0.996937513	0	0	0
1	DAE IMBALANCE	0.994937479	0	0	0
1	BPNN IMBALANCE	0.991937518	0	0	0

3.5. EXPERIMENTO 5: COMPARACIÓN DE LOS MODELOS DL TENIENDO EN CUENTA LOS RES

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
1	AE IMBALANCE	0.98862499	0	0	0

Tabla 3.13: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.13 y los anexos 3.10, 3.11, 3.12 y 3.13, se pueden obtener las siguientes observaciones:

- El mejor modelo es CNN.
- El segundo mejor modelo es BPNN.
- Los peores modelos son AE y DAE.
- Los modelos CNN y BPNN mejoran sus resultados a medida que aumenta la cantidad de *epochs*.
- Los modelos AE y DAE no mejoran en ninguna cantidad de *epochs*, obteniendo pésimos resultados.

UnderSampler

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	CNN UNDERSAMPLE	0.908163249	0.976190448	0.83673471	0.90109885
100	BPNN UNDERSAMPLE	0.816326559	0.94155848	0.76789212	0.84010977
50	CNN UNDERSAMPLE	0.846938789	0.886363626	0.79591835	0.83870965
20	CNN UNDERSAMPLE	0.795918345	0.808510661	0.77551019	0.79166663
100	AE UNDERSAMPLE	0.5	0.6171875	1	0.73930478
50	LSTM UNDERSAMPLE	0.632653058	0.774999976	0.58002269	0.6529811
50	BPNN UNDERSAMPLE	0.744897962	1	0.48214287	0.64673036
20	AE UNDERSAMPLE	0.5	0.623205423	0.80570173	0.64404374
20	LSTM UNDERSAMPLE	0.571428597	0.667714715	0.58994222	0.6053344
1	AE UNDERSAMPLE	0.53061223	0.632142842	0.70472753	0.5985378
100	LSTM UNDERSAMPLE	0.622448981	0.75	0.48333335	0.57905978
1	CNN UNDERSAMPLE	0.520408154	0.516666651	0.63265306	0.5688073
100	RNN UNDERSAMPLE	0.765306115	0.586779475	0.54345655	0.56374556
20	DAE UNDERSAMPLE	0.5	0.3828125	0.75	0.50680268
50	AE UNDERSAMPLE	0.5	0.3828125	0.75	0.50511503
50	RNN UNDERSAMPLE	0.663265288	0.52136755	0.4696545	0.49259257
20	RNN UNDERSAMPLE	0.653061211	0.525990665	0.40697944	0.45634919
20	BPNN UNDERSAMPLE	0.602040827	0.446018875	0.4627451	0.44985989
1	RNN UNDERSAMPLE	0.602040827	0.448525131	0.430668	0.43900132
1	LSTM UNDERSAMPLE	0.520408154	0.397058845	0.3821016	0.38839284
1	BPNN UNDERSAMPLE	0.540816307	0.4375	0.21666667	0.28929761
1	DAE UNDERSAMPLE	0.561224461	0.15625	0.25	0.19230768
100	DAE UNDERSAMPLE	0.5	0	0	0

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
50	DAE UNDERSAMPLE	0.5	0	0	0

Tabla 3.14: Puntuaciones del entrenamiento de los modelos DL con UnderSampler en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.14 y los anexos 3.14, 3.15, 3.16 y 3.17, se obtienen las siguientes observaciones:

- El mejor modelo es CNN.
- El segundo mejor modelo es BPNN.
- El peor modelo es DAE.
- Los modelos DAE con epochs superiores a los 20, empeoran sus resultados de entrenamiento.
- Los modelos CNN y BPNN mejoran sus resultados a medida que aumenta la cantidad de epochs.

OverSampler

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	BPNN OVERSAMPLE	0.996364295	0.993416727	0.99922538	0.99617994
100	CNN OVERSAMPLE	0.993919611	0.99008745	0.99774992	0.99383694
100	LSTM OVERSAMPLE	0.993449509	0.99035269	0.99682218	0.99335557
50	BPNN OVERSAMPLE	0.993042052	0.98695153	0.99906373	0.99273348
50	CNN OVERSAMPLE	0.991412282	0.986423314	0.99655753	0.99138087
50	LSTM OVERSAMPLE	0.990942121	0.985821307	0.99627328	0.99069422
20	CNN OVERSAMPLE	0.990283966	0.985048652	0.9956938	0.99025965
100	RNN OVERSAMPLE	0.984266281	0.978752553	0.98808312	0.98288071
20	LSTM OVERSAMPLE	0.983200669	0.97532773	0.9914127	0.98272848
20	BPNN OVERSAMPLE	0.982417107	0.991747379	0.97327888	0.98179853
50	RNN OVERSAMPLE	0.982103705	0.977232516	0.98553199	0.98072463
20	RNN OVERSAMPLE	0.979220212	0.973606467	0.9855867	0.97885162
1	BPNN OVERSAMPLE	0.83090955	0.83434689	0.83071715	0.8243857
1	CNN OVERSAMPLE	0.826960444	0.825432777	0.82687044	0.82379562
1	RNN OVERSAMPLE	0.686234593	0.686507225	0.681256	0.67567873
1	LSTM OVERSAMPLE	0.590108454	0.666488409	0.3947688	0.47518793
20	AE OVERSAMPLE	0.500156701	0.313696891	0.62725449	0.41537589
50	DAE OVERSAMPLE	0.49316743	0.286667079	0.58016032	0.38089424
100	AE OVERSAMPLE	0.497712016	0.23843208	0.47645834	0.31496578
1	AE OVERSAMPLE	0.494609177	0.235250413	0.4743332	0.31219319
100	DAE OVERSAMPLE	0.498620957	0.230783135	0.46292585	0.30592763
1	DAE OVERSAMPLE	0.49880901	0.212330908	0.4258517	0.28140163
50	AE OVERSAMPLE	0.495612115	0.192697898	0.38977957	0.2556991
20	DAE OVERSAMPLE	0.500438809	0.149486467	0.2975952	0.19771725

3.5. EXPERIMENTO 5: COMPARACIÓN DE LOS MODELOS DL TENIENDO EN CUENTA LOS RES

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
-------	-------	----------	-----------------	--------------	----------

Tabla 3.15: Puntuaciones del entrenamiento de los modelos DL con OverSampler en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.15 y los anexos 3.18, 3.19, 3.20 y 3.21, se pueden obtener las siguientes observaciones:

- El mejor modelo es BPNN.
- El segundo mejor modelo es CNN.
- El peor modelo es DAE.
- Los modelos CNN, BPNN, RNN y LSTM mejoran sus resultados mientras mayor sea la cantidad de *epochs*.

SMOTE

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	BPNN SMOTE	0.998997092	0.998539507	0.99923921	0.99884498
50	BPNN SMOTE	0.998558342	0.998017848	0.99916291	0.99854267
100	LSTM SMOTE	0.997524142	0.996127367	0.99886996	0.99741119
100	CNN SMOTE	0.995518386	0.994039357	0.99706334	0.99551123
50	CNN SMOTE	0.993575275	0.992109418	0.99509072	0.99353939
20	CNN SMOTE	0.99069196	0.988459527	0.99304038	0.99065036
50	LSTM SMOTE	0.990880013	0.986309826	0.99564129	0.99063724
100	RNN SMOTE	0.984831393	0.981499672	0.98868603	0.98457217
50	RNN SMOTE	0.984392643	0.982494891	0.98650694	0.98391545
20	RNN SMOTE	0.973047495	0.966816664	0.97994155	0.97239929
20	LSTM SMOTE	0.963958859	0.947134852	0.9833923	0.96373844
20	BPNN SMOTE	0.938291311	0.953579187	0.92108405	0.93480515
1	CNN SMOTE	0.85335964	0.859778166	0.84736043	0.85067493
1	BPNN SMOTE	0.800081491	0.81560123	0.79757857	0.79251993
1	RNN SMOTE	0.719788134	0.720076799	0.72189444	0.71296543
100	DAE SMOTE	0.495800436	0.417240739	0.83867735	0.55287749
1	LSTM SMOTE	0.627648234	0.69548589	0.48815134	0.55267763
50	AE SMOTE	0.501065552	0.31172353	0.58127224	0.396272
100	AE SMOTE	0.498746395	0.275425851	0.55210423	0.36485434
50	DAE SMOTE	0.497868866	0.272701651	0.54709417	0.36141822
1	AE SMOTE	0.501190901	0.330650508	0.50685281	0.35451099
20	DAE SMOTE	0.494045377	0.1691508	0.34468937	0.22527717
20	AE SMOTE	0.500407398	0.217240199	0.25200582	0.1867536
1	DAE SMOTE	0.50162971	0.109249748	0.21643287	0.14415723

Tabla 3.16: Puntuaciones del entrenamiento de los modelos DL con SMOTE en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.16 y los anexos 3.22, 3.23, 3.24 y 3.25, se pueden obtener las siguientes observaciones:

- El mejor modelo es BPNN.
- El segundo mejor modelo es LSTM.
- El peor modelo es DAE.
- Los modelos CNN, BPNN, RNN, LSTM y DAE mejoran a medida que aumenta la cantidad de *epochs*.

ADASYN

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	BPNN ADASYN	0.997366965	0.997244418	0.99752074	0.99728757
20	BPNN ADASYN	0.996583283	0.996061802	0.99714017	0.99647945
100	LSTM ADASYN	0.996144414	0.99491775	0.99749511	0.99607491
50	BPNN ADASYN	0.995141387	0.992640615	0.99756128	0.9949252
100	CNN ADASYN	0.993354678	0.991527438	0.9952265	0.99330389
50	CNN ADASYN	0.992288888	0.989902735	0.99474478	0.99225354
50	LSTM ADASYN	0.991285801	0.988178194	0.99448264	0.99104667
20	CNN ADASYN	0.988527358	0.985117078	0.99214935	0.98851031
100	RNN ADASYN	0.98570621	0.98420471	0.98729259	0.9852491
50	RNN ADASYN	0.980973005	0.97678107	0.98526347	0.98037964
20	RNN ADASYN	0.976459146	0.971780539	0.98173946	0.9758895
20	LSTM ADASYN	0.970378041	0.961096585	0.98171395	0.97024137
1	CNN ADASYN	0.865149498	0.865566611	0.86654222	0.86335516
1	BPNN ADASYN	0.771581709	0.790596604	0.76459557	0.76627654
1	RNN ADASYN	0.709234536	0.711095929	0.70515919	0.69940358
1	LSTM ADASYN	0.621904612	0.670764625	0.49858537	0.55402362
100	AE ADASYN	0.501786709	0.353379697	0.70511532	0.4674027
50	AE ADASYN	0.497366935	0.333625883	0.67001003	0.44187346
50	DAE ADASYN	0.495674253	0.325257033	0.65496492	0.4315089
20	DAE ADASYN	0.496489257	0.290057659	0.58375126	0.38466439
1	AE ADASYN	0.499686539	0.439912975	0.32745457	0.2894938
1	DAE ADASYN	0.495580226	0.112566531	0.22968906	0.14997599
100	DAE ADASYN	0.497931153	0.103723668	0.20962888	0.13783602
20	AE ADASYN	0.499435782	0.05381979	0.10832497	0.07135723

Tabla 3.17: Puntuaciones del entrenamiento de los modelos DL con ADASYN en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.17 y los anexos 3.26, 3.27, 3.28 y 3.29, se pueden obtener las siguientes observaciones:

- El mejor modelo es BPNN.
- El segundo mejor modelo es LSTM.

3.5. EXPERIMENTO 5: COMPARACIÓN DE LOS MODELOS DL TENIENDO EN CUENTA LOS RES

- El peor modelo es DAE.
- Los modelos CNN, BPNN, RNN, LSTM y DAE mejoran a medida que aumenta la cantidad de *epochs*.

3.5.2 Análisis de la primera parte

Por los resultados obtenidos y las observaciones que se han podido apreciar, se pueden obtener las siguientes conclusiones:

- El mejor modelo en el entrenamiento con datos desbalanceados y UnderSampler es CNN, seguido de BPNN como el segundo mejor modelo en ambas estrategias.
- El mejor modelo en el entrenamiento con OverSampler, SMOTE y ADASYN es BPNN.
- Los modelos BPNN, CNN, RNN y LSTM mejoran sus resultados a medida que aumenta la cantidad de *epochs* en cualquier estrategia aplicada.
- Los peores modelos en todas las estrategias son AE y DAE.

Basado en las observaciones de la primera parte, se decide tomar el modelo CNN con datos desbalanceados y UnderSampler, y el modelo BPNN con OverSampler, SMOTE y ADASYN para próximos experimentos.

3.5.3 Segunda parte: Estrategias y epochs por modelos

Se cambia la forma de comparar los resultados, de forma tal que se obtenga la mejor estrategia y *epoch* por modelo.

CNN

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	CNN SMOTE	0.995518386	0.994039357	0.99706334	0.99551123
100	CNN OVERSAMPLE	0.993919611	0.99008745	0.99774992	0.99383694
50	CNN SMOTE	0.993575275	0.992109418	0.99509072	0.99353939
100	CNN ADASYN	0.993354678	0.991527438	0.9952265	0.99330389
50	CNN ADASYN	0.992288888	0.989902735	0.99474478	0.99225354
50	CNN OVERSAMPLE	0.991412282	0.986423314	0.99655753	0.99138087
20	CNN SMOTE	0.99069196	0.988459527	0.99304038	0.99065036
20	CNN OVERSAMPLE	0.990283966	0.985048652	0.9956938	0.99025965
20	CNN ADASYN	0.988527358	0.985117078	0.99214935	0.98851031
100	CNN UNDERSAMPLE	0.908163249	0.976190448	0.83673471	0.90109885
1	CNN ADASYN	0.865149498	0.865566611	0.86654222	0.86335516
1	CNN SMOTE	0.85335964	0.859778166	0.84736043	0.85067493
50	CNN UNDERSAMPLE	0.846938789	0.886363626	0.79591835	0.83870965
1	CNN OVERSAMPLE	0.826960444	0.825432777	0.82687044	0.82379562

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
20	CNN UNDERSAMPLE	0.795918345	0.808510661	0.77551019	0.79166663
1	CNN UNDERSAMPLE	0.520408154	0.516666651	0.63265306	0.5688073
100	CNN IMBALANCE	0.998749971	0.17832166	0.17016315	0.17132866
50	CNN IMBALANCE	0.998187482	0.146853134	0.12820512	0.13403264
20	CNN IMBALANCE	0.997562528	0.1048951	0.08974358	0.09440558
1	CNN IMBALANCE	0.98724997	0.040209785	0.04079254	0.04009324

Tabla 3.18: Puntuaciones del entrenamiento del modelo CNN por estrategias en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.18 y los anexos 3.30, 3.31, 3.32 y 3.33 se pueden obtener las siguientes observaciones:

- La mejor estrategia es SMOTE en 100 epochs.
- La segunda mejor estrategia es OverSampler en 100 epochs.
- La peor estrategia es con datos desbalanceados con 1 epochs.
- Los resultados del entrenamiento para el modelo CNN en todas las estrategias, mejoran a medida que aumenta la cantidad de epochs.

AE

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	AE UNDERSAMPLE	0.5	0.6171875	1	0.73930478
20	AE UNDERSAMPLE	0.5	0.623205423	0.80570173	0.64404374
1	AE UNDERSAMPLE	0.53061223	0.632142842	0.70472753	0.5985378
50	AE UNDERSAMPLE	0.5	0.3828125	0.75	0.50511503
100	AE ADASYN	0.501786709	0.353379697	0.70511532	0.4674027
50	AE ADASYN	0.497366935	0.333625883	0.67001003	0.44187346
20	AE OVERSAMPLE	0.500156701	0.313696891	0.62725449	0.41537589
50	AE SMOTE	0.501065552	0.31172353	0.58127224	0.396272
100	AE SMOTE	0.498746395	0.275425851	0.55210423	0.36485434
1	AE SMOTE	0.501190901	0.330650508	0.50685281	0.35451099
100	AE OVERSAMPLE	0.497712016	0.23843208	0.47645834	0.31496578
1	AE OVERSAMPLE	0.494609177	0.235250413	0.4743332	0.31219319
1	AE ADASYN	0.499686539	0.439912975	0.32745457	0.2894938
50	AE OVERSAMPLE	0.495612115	0.192697898	0.38977957	0.2556991
20	AE SMOTE	0.500407398	0.217240199	0.25200582	0.1867536
20	AE ADASYN	0.499435782	0.05381979	0.10832497	0.07135723
100	AE IMBALANCE	0.996937513	0	0	0
20	AE IMBALANCE	0.996937513	0	0	0
50	AE IMBALANCE	0.996937513	0	0	0
1	AE IMBALANCE	0.98862499	0	0	0

3.5. EXPERIMENTO 5: COMPARACIÓN DE LOS MODELOS DL TENIENDO EN CUENTA LOS RES

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
-------	-------	----------	-----------------	--------------	----------

Tabla 3.19: Puntuaciones del entrenamiento del modelo

AE por estrategias en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.19 y los anexos 3.34, 3.35, 3.36 y 3.37, se obtienen las siguientes observaciones:

- La mejor estrategia es UnderSampler en 100 *epochs*.
- La segunda mejor estrategia es ADASYN en 100 *epochs*.
- La peor estrategia es con datos desbalanceados en 1 *epoch*.
- Los entrenamientos del modelo AE para las estrategias UnderSampler, OverSampler, SMOTE y ADASYN son inestables.

DAE

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	DAE SMOTE	0.495800436	0.417240739	0.83867735	0.55287749
20	DAE UNDERSAMPLE	0.5	0.3828125	0.75	0.50680268
50	DAE ADASYN	0.495674253	0.325257033	0.65496492	0.4315089
20	DAE ADASYN	0.496489257	0.290057659	0.58375126	0.38466439
50	DAE OVERSAMPLE	0.49316743	0.286667079	0.58016032	0.38089424
50	DAE SMOTE	0.497868866	0.272701651	0.54709417	0.36141822
100	DAE OVERSAMPLE	0.498620957	0.230783135	0.46292585	0.30592763
1	DAE OVERSAMPLE	0.49880901	0.212330908	0.4258517	0.28140163
20	DAE SMOTE	0.494045377	0.1691508	0.34468937	0.22527717
20	DAE OVERSAMPLE	0.500438809	0.149486467	0.2975952	0.19771725
1	DAE UNDERSAMPLE	0.561224461	0.15625	0.25	0.19230768
1	DAE ADASYN	0.495580226	0.112566531	0.22968906	0.14997599
1	DAE SMOTE	0.50162971	0.109249748	0.21643287	0.14415723
100	DAE ADASYN	0.497931153	0.103723668	0.20962888	0.13783602
20	DAE IMBALANCE	0.996937513	0	0	0
50	DAE IMBALANCE	0.996937513	0	0	0
100	DAE IMBALANCE	0.996937513	0	0	0
1	DAE IMBALANCE	0.994937479	0	0	0
50	DAE UNDERSAMPLE	0.5	0	0	0
100	DAE UNDERSAMPLE	0.5	0	0	0

Tabla 3.20: Puntuaciones del entrenamiento del modelo

DAE por estrategias en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.20 y los anexos 3.38, 3.39, 3.40 y 3.41, se obtienen las siguientes observaciones:

- La mejor estrategia es SMOTE en 100 *epochs*.

- La segunda mejor estrategia es UnderSampler en 20 *epochs*.
- Las peores estrategias son UnderSampler en 50 y 100 *epochs*, y con datos desbalanceados.
- Los entrenamientos del modelo DAE son inestables para todas las estrategias aplicadas.

RNN

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	RNN ADASYN	0.98570621	0.98420471	0.98729259	0.9852491
100	RNN SMOTE	0.984831393	0.981499672	0.98868603	0.98457217
50	RNN SMOTE	0.984392643	0.982494891	0.98650694	0.98391545
100	RNN OVERSAMPLE	0.984266281	0.978752553	0.98808312	0.98288071
50	RNN OVERSAMPLE	0.982103705	0.977232516	0.98553199	0.98072463
50	RNN ADASYN	0.980973005	0.97678107	0.98526347	0.98037964
20	RNN OVERSAMPLE	0.979220212	0.973606467	0.9855867	0.97885162
20	RNN ADASYN	0.976459146	0.971780539	0.98173946	0.9758895
20	RNN SMOTE	0.973047495	0.966816664	0.97994155	0.97239929
1	RNN SMOTE	0.719788134	0.720076799	0.72189444	0.71296543
1	RNN ADASYN	0.709234536	0.711095929	0.70515919	0.69940358
1	RNN OVERSAMPLE	0.686234593	0.686507225	0.681256	0.67567873
100	RNN UNDERSAMPLE	0.765306115	0.586779475	0.54345655	0.56374556
50	RNN UNDERSAMPLE	0.663265288	0.52136755	0.4696545	0.49259257
20	RNN UNDERSAMPLE	0.653061211	0.525990665	0.40697944	0.45634919
1	RNN UNDERSAMPLE	0.602040827	0.448525131	0.430668	0.43900132
100	RNN IMBALANCE	0.997437477	0.021999998	0.02	0.02066666
20	RNN IMBALANCE	0.997500002	0.017999997	0.017	0.01733333
1	RNN IMBALANCE	0.968625009	0.009999999	0.01	0.01
50	RNN IMBALANCE	0.99712503	0.007999999	0.007	0.00733333

Tabla 3.21: Puntuaciones del entrenamiento del modelo RNN por estrategias en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.21 y los anexos 3.42, 3.43, 3.44 y 3.45, se obtienen las siguientes observaciones:

- La mejor estrategia es ADASYN en 100 *epochs*.
- La segunda mejor estrategia es SMOTE en 100 *epochs*.
- La peor estrategia es con datos desbalanceados en 50 *epochs*.
- El modelo RNN con las estrategias UnderSampler, OverSampler, SMOTE y ADASYN obtienen mejores resultados en el entrenamiento a medida que aumenta la cantidad de *epochs*, además presenta estabilidad en los entrenamientos.

LSTM

3.5. EXPERIMENTO 5: COMPARACIÓN DE LOS MODELOS DL TENIENDO EN CUENTA LOS RES

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	LSTM SMOTE	0.997524142	0.996127367	0.99886996	0.99741119
100	LSTM ADASYN	0.996144414	0.99491775	0.99749511	0.99607491
100	LSTM OVERSAMPLE	0.993449509	0.99035269	0.99682218	0.99335557
50	LSTM ADASYN	0.991285801	0.988178194	0.99448264	0.99104667
50	LSTM OVERSAMPLE	0.990942121	0.985821307	0.99627328	0.99069422
50	LSTM SMOTE	0.990880013	0.986309826	0.99564129	0.99063724
20	LSTM OVERSAMPLE	0.983200669	0.97532773	0.9914127	0.98272848
20	LSTM ADASYN	0.970378041	0.961096585	0.98171395	0.97024137
20	LSTM SMOTE	0.963958859	0.947134852	0.9833923	0.96373844
50	LSTM UNDERSAMPLE	0.632653058	0.774999976	0.58002269	0.6529811
20	LSTM UNDERSAMPLE	0.571428597	0.667714715	0.58994222	0.6053344
100	LSTM UNDERSAMPLE	0.622448981	0.75	0.48333335	0.57905978
1	LSTM ADASYN	0.621904612	0.670764625	0.49858537	0.55402362
1	LSTM SMOTE	0.627648234	0.69548589	0.48815134	0.55267763
1	LSTM OVERSAMPLE	0.590108454	0.666488409	0.3947688	0.47518793
1	LSTM UNDERSAMPLE	0.520408154	0.397058845	0.3821016	0.38839284
50	LSTM IMBALANCE	0.9979375	0.031999998	0.032	0.032
100	LSTM IMBALANCE	0.99781251	0.029999997	0.027	0.028
20	LSTM IMBALANCE	0.997562528	0.019999998	0.02	0.02
1	LSTM IMBALANCE	0.99150002	0.004	0.004	0.004

Tabla 3.22: Puntuaciones del entrenamiento del modelo LSTM por estrategias en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.22 y los anexos 3.46, 3.47, 3.48 y 3.49, se obtienen las siguientes observaciones:

- La mejor estrategia es SMOTE en 100 epochs.
- La segunda mejor estrategia es ADASYN en 100 epochs.
- La peor estrategia es con datos desbalanceados en 1 epochs.
- El modelo LSTM con las estrategias SMOTE, ADASYN y OverSampler obtienen mejores resultados a medida que aumenta la cantidad de epochs, además presentan entrenamientos estables.

BPNN

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
100	BPNN SMOTE	0.998997092	0.998539507	0.99923921	0.99884498
50	BPNN SMOTE	0.998558342	0.998017848	0.99916291	0.99854267
100	BPNN ADASYN	0.997366965	0.997244418	0.99752074	0.99728757
20	BPNN ADASYN	0.996583283	0.996061802	0.99714017	0.99647945
100	BPNN OVERSAMPLE	0.996364295	0.993416727	0.99922538	0.99617994
50	BPNN ADASYN	0.995141387	0.992640615	0.99756128	0.9949252
50	BPNN OVERSAMPLE	0.993042052	0.98695153	0.99906373	0.99273348

Epoch	Model	Accuracy	Precision Train	Recall Train	F1 Train
20	BPNN OVERSAMPLE	0.982417107	0.991747379	0.97327888	0.98179853
20	BPNN SMOTE	0.938291311	0.953579187	0.92108405	0.93480515
100	BPNN UNDERSAMPLE	0.816326559	0.94155848	0.76789212	0.84010977
1	BPNN OVERSAMPLE	0.83090955	0.83434689	0.83071715	0.8243857
1	BPNN SMOTE	0.800081491	0.81560123	0.79757857	0.79251993
1	BPNN ADASYN	0.771581709	0.790596604	0.76459557	0.76627654
50	BPNN UNDERSAMPLE	0.744897962	1	0.48214287	0.64673036
20	BPNN UNDERSAMPLE	0.602040827	0.446018875	0.4627451	0.44985989
1	BPNN UNDERSAMPLE	0.540816307	0.4375	0.21666667	0.28929761
100	BPNN IMBALANCE	0.998687506	0.057999995	0.057	0.05733333
50	BPNN IMBALANCE	0.998437524	0.045999996	0.046	0.046
20	BPNN IMBALANCE	0.99849999	0.045999996	0.044	0.04466666
1	BPNN IMBALANCE	0.991937518	0	0	0

Tabla 3.23: Puntuaciones del entrenamiento del modelo BPNN por estrategias en 1, 20, 50 y 100 epochs.

Basados en la tabla 3.23 y los anexos 3.50, 3.51, 3.52 y 3.53, se obtienen las siguientes observaciones:

- La mejor estrategia es SMOTE en 100 *epochs*.
- La segunda mejor estrategia es ADASYN en 100 *epochs*.
- La peor estrategia es con datos desbalanceados en 1 *epoch*.
- El modelo BPNN con las estrategias UnderSampler, OverSampler, SMOTE y con datos desbalanceados, mejoran sus resultados de entrenamiento a medida que aumenta la cantidad de *epochs*, siendo los más estables.

3.5.4 Análisis de la segunda parte

Una vez concluido el análisis de los modelos DL por sus estrategias y *epochs*, están preparadas las condiciones para escoger los mejores modelos. Basados en las observaciones de esta segunda parte se muestra la siguiente matriz con la cantidad de *epochs* y la estrategia de los modelos, donde se obtienen los mejores resultados:

Estrategias					
	Imbalanced	UnderSampler	OverSampler	SMOTE	ADASYN
CNN	100	100	100	100	100
AE	NO	100	20	50	100
DAE	NO	20	50	100	50
RNN	100	100	100	100	100
LSTM	50	50	100	100	100
BPNN	100	100	100	100	100

Tabla 3.24: Matriz de las mejores estrategias y *epochs* por modelos.

Basándose en la tabla 3.24, se decide utilizar el modelo CNN, DAE, LSTM y BPNN con SMOTE, el modelo AE con UnderSampler y el modelo RNN con ADASYN, todos con 100 *epochs*, para próximos experimentos por sus resultados en los entrenamientos.

3.5.5 Análisis general del experimento 5

Realizando la unión de las observaciones de las tres partes se obtienen los mejores modelos en entrenamiento, por lo tanto, se toman las siguientes observaciones:

- El modelo CNN obtiene los mejores resultados de entrenamiento con datos desbalanceados y UnderSampler en 100 *epochs*.
- El modelo BPNN obtiene los mejores resultados de entrenamiento con OverSampler, SMOTE y ADASYN en 100 *epochs*.
- El mejor resultado de CNN es con SMOTE en 50 *epochs*.
- El mejor resultado de AE es con UnderSampler en 100 *epochs*.
- El mejor resultado de DAE es con SMOTE en 100 *epochs*.
- El mejor resultado de RNN es con ADASYN en 100 *epochs*.
- El mejor resultado de LSTM es con SMOTE en 100 *epochs*.
- El mejor resultado de BPNN es con SMOTE en 100 *epochs*.

Teniendo en cuenta esto, se decide tomar el modelo CNN con datos desbalanceados y UnderSampler, el modelo BPNN con OverSampler, SMOTE y ADASYN, con 100 *epochs* ambos modelos para próximos experimentos como los mejores en entrenamiento.

3.6 Análisis de los resultados de los experimentos 4 y 5

Es necesario encontrar un término medio entre los resultados del experimento 4 y 5, ya que lo más factible sería que los modelos seleccionados sean los mejores tanto en pruebas como en entrenamiento, pero cabe destacar que la prioridad la tiene los resultados de las pruebas.

Como resultado del experimento 4 se obtiene a BPNN como el mejor modelo en UnderSampler, OverSampler, SMOTE y ADASYN en 100 *epochs*, coincidiendo con los resultados del entrenamiento en el experimento 5, a excepción de UnderSampler, donde el mejor resultado lo obtiene CNN. En cuanto a los datos desbalanceados, en el experimento 4, CNN obtiene los mejores resultados en 1 y 50 *epochs*, y en el experimento 5, lo obtiene CNN en 100 *epochs*.

Teniendo en cuenta los resultados del experimento 4 tienen mayor prioridad sobre el experimento 5, se toman las siguientes decisiones:

- En cuanto a la estrategia UnderSampler, se toma el modelo BPNN con 100 *epochs* por sus resultados en las pruebas y obtener mejores resultados de entrenamiento para esa cantidad de *epochs*.

- Se toma el modelo BPNN con 100 *epochs* en las estrategias OverSampler, SMOTE y ADASYN por coincidir con los mejores resultados tanto en pruebas como entrenamiento.
- Para la estrategia con datos desbalanceados se toma el modelo CNN con 50 *epochs* por sus resultados en las pruebas, aunque los mejores resultados de entrenamiento se con 100 *epochs*, los resultados de entrenamientos mejoran a medida que aumenta la cantidad de *epochs*. Por lo tanto, no afecta en gran medida los resultados.

Una vez definidos los mejores modelos DL por estrategias, se puede proceder a la realización del próximo experimento.

3.7 Experimento 6: Comparación entre los mejores modelos ML y DL, en cuanto a los resultados de las pruebas

El objetivo de este experimento es demostrar la superioridad de los modelos DL sobre los modelos ML. Para llevar a cabo este experimento se tomarán los mejores modelos ML resultantes del experimento 2 y los mejores modelos DL resultantes del experimento 4 y 5.

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
100	BPNN ADASYN	0.998	0.5	0.998004	0.998	0.997001
100	BPNN OVERSAMPLE	0.99775	0.5	0.997755	0.99775	0.996626
50	CNN IMBALANCE	0.997	0.633208	0.996507	0.997	0.996252
100	BPNN SMOTE	0.99725	0.5	0.997258	0.99725	0.995877
100	BPNN UNDERSAMPLE	0.99625	0.5	0.996264	0.99625	0.994379
0	RF OVERSAMPLE	0.99875	0.722222	1	0.444444	0.615385
0	RF IMABALANCED	0.997	0.6	1	0.2	0.333333
0	RF SMOTE	0.99725	0.545329	0.5	0.090909	0.153846
0	NB UNDERSAMPLE	0.96	0.647846	0.032258	0.333333	0.058824
0	RF ADASYN	0.99775	0.499875	0	0	0

Tabla 3.25: Puntuaciones de las pruebas de los mejores modelos ML y DL.

Basados en la tabla 3.25 se pueden obtener las siguientes observaciones:

- Los modelos DL son superiores a los modelos ML en resultados.
- El modelo BPNN con ADASYN en 100 *epochs* es superior a los demás modelos DL y ML.
- El segundo mejor modelo es BPNN con OverSampler en 100 *epochs* con respecto a los demás modelos DL y ML.
- El mejor modelo ML es RF con OverSampler.
- El segundo mejor modelo ML es RF con datos desbalanceados.
- El peor modelo es RF con ADASYN con respecto a los modelos ML y DL.

3.8. EXPERIMENTO 7: COMPARACIÓN DE TODOS LOS MODELOS ML Y DL EN EL RESULTADO

- El peor modelo DL es BPNN con UnderSampler en 100 *epochs*.
- La estrategia ADASYN es la mejor en los modelos DL, sin embargo, es la peor en los modelos ML.
- La estrategia OverSampler es la mejor en los modelos ML, sin embargo, es el segundo mejor en los modelos DL.

Teniendo en cuenta las observaciones obtenidas, se llega a la conclusión de que los modelos DL son superiores a los modelos ML, en específico, el modelo BPNN con ADASYN en 100 *epochs*.

3.8 Experimento 7: Comparación de todos los modelos ML y DL en el resultado de las pruebas

Este es el experimento final donde se realizan las pruebas de los modelos ML junto a los modelos DL usando todas las estrategias utilizadas y en caso de los modelos DL se realizarán cada uno en 1, 20, 50 y 100 *epochs* de entrenamiento. De esta manera se podrá confirmar la superioridad de los modelos DL sobre los modelos ML. No obstante, se seleccionarán los tres mejores modelos teniendo en cuenta las puntuaciones de las métricas.

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
20	CNN IMBALANCE	0.9975	0.642857143	0.997506256	0.9975	0.9968071
50	CNN IMBALANCE	0.99725	0.642731704	0.996805632	0.99725	0.9966002
1	CNN IMBALANCE	0.99725	0.607142857	0.997257568	0.99725	0.9963621
100	CNN IMBALANCE	0.9965	0.642355387	0.995753758	0.9965	0.9960240
50	BPNN SMOTE	0.9965	0.5	0.99651225	0.9965	0.9947530
100	BPNN SMOTE	0.9965	0.5	0.99651225	0.9965	0.9947530
20	BPNN SMOTE	0.9965	0.5	0.99651225	0.9965	0.9947530
100	BPNN UNDERSAMPLE	0.9965	0.5	0.99651225	0.9965	0.9947530
1	BPNN SMOTE	0.9965	0.5	0.99651225	0.9965	0.9947530
50	BPNN UNDERSAMPLE	0.9965	0.5	0.99651225	0.9965	0.9947530
20	BPNN UNDERSAMPLE	0.9965	0.5	0.99651225	0.9965	0.9947530
50	BPNN IMBALANCE	0.9965	0.5	0.99651225	0.9965	0.9947530
100	BPNN IMBALANCE	0.9965	0.5	0.99651225	0.9965	0.9947530
20	BPNN IMBALANCE	0.9965	0.5	0.99651225	0.9965	0.9947530
1	BPNN IMBALANCE	0.9965	0.5	0.99651225	0.9965	0.9947530
1	BPNN UNDERSAMPLE	0.9965	0.5	0.99651225	0.9965	0.9947530
100	LSTM SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	LSTM SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	RNN SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	RNN SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	RNN SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	LSTM SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	RNN UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	LSTM UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	DAE UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
100	DAE UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	RNN SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	LSTM UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	RNN UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	LSTM UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	RNN UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	RNN UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	AE SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	LSTM SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	LSTM UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	DAE SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	AE UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	AE SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	DAE UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	AE SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	DAE SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	DAE UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	AE SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	DAE SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	DAE SMOTE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	LSTM IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	LSTM IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	RNN IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	RNN IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	LSTM IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	RNN IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	RNN IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	LSTM IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	DAE IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	DAE IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	DAE IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	AE IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	AE IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	AE IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	DAE IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
1	AE IMBALANCE	0.9965	0.5	0.99301225	0.9965	0.9947530
20	AE UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	AE UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
100	AE UNDERSAMPLE	0.9965	0.5	0.99301225	0.9965	0.9947530
50	BPNN OVERSAMPLE	0.996	0.5	0.996016	0.996	0.9940040
100	BPNN OVERSAMPLE	0.996	0.5	0.996016	0.996	0.9940040
100	BPNN ADASYN	0.996	0.5	0.996016	0.996	0.9940040
50	BPNN ADASYN	0.996	0.5	0.996016	0.996	0.9940040
20	BPNN ADASYN	0.996	0.5	0.996016	0.996	0.9940040
20	BPNN OVERSAMPLE	0.996	0.5	0.996016	0.996	0.9940040
1	BPNN OVERSAMPLE	0.996	0.5	0.996016	0.996	0.9940040
1	BPNN ADASYN	0.996	0.5	0.996016	0.996	0.9940040
100	LSTM OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040

3.8. EXPERIMENTO 7: COMPARACIÓN DE TODOS LOS MODELOS ML Y DL EN EL RESULTADO

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
100	LSTM ADASYN	0.996	0.5	0.992016	0.996	0.9940040
50	LSTM OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
50	LSTM ADASYN	0.996	0.5	0.992016	0.996	0.9940040
100	RNN OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
50	RNN OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
20	LSTM OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
100	RNN ADASYN	0.996	0.5	0.992016	0.996	0.9940040
20	RNN OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
50	RNN ADASYN	0.996	0.5	0.992016	0.996	0.9940040
20	LSTM ADASYN	0.996	0.5	0.992016	0.996	0.9940040
20	RNN ADASYN	0.996	0.5	0.992016	0.996	0.9940040
1	RNN OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
1	RNN ADASYN	0.996	0.5	0.992016	0.996	0.9940040
100	AE OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
50	DAE OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
1	LSTM ADASYN	0.996	0.5	0.992016	0.996	0.9940040
1	LSTM OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
100	DAE ADASYN	0.996	0.5	0.992016	0.996	0.9940040
1	AE ADASYN	0.996	0.5	0.992016	0.996	0.9940040
100	AE ADASYN	0.996	0.5	0.992016	0.996	0.9940040
1	DAE ADASYN	0.996	0.5	0.992016	0.996	0.9940040
20	DAE ADASYN	0.996	0.5	0.992016	0.996	0.9940040
50	AE ADASYN	0.996	0.5	0.992016	0.996	0.9940040
1	AE OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
20	AE OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
20	DAE OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
50	DAE ADASYN	0.996	0.5	0.992016	0.996	0.9940040
20	AE ADASYN	0.996	0.5	0.992016	0.996	0.9940040
1	DAE OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
100	DAE OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
50	AE OVERSAMPLE	0.996	0.5	0.992016	0.996	0.9940040
50	CNN SMOTE	0.99225	0.640222923	0.994553082	0.99225	0.9933375
100	CNN OVERSAMPLE	0.99275	0.622866466	0.993758135	0.99275	0.9932375
100	CNN ADASYN	0.9915	0.653363454	0.993955978	0.9915	0.9926526
50	CNN ADASYN	0.99125	0.653237952	0.993930652	0.99125	0.9925067
50	CNN OVERSAMPLE	0.9915	0.591114458	0.993243461	0.9915	0.9923457
100	CNN SMOTE	0.99	0.639093972	0.994399158	0.99	0.9920707
20	CNN OVERSAMPLE	0.98875	0.589733936	0.993077283	0.98875	0.9908321
20	CNN SMOTE	0.98175	0.634954484	0.994175266	0.98175	0.9876596
20	CNN ADASYN	0.981	0.616967871	0.99319562	0.981	0.9868185
1	CNN OVERSAMPLE	0.89525	0.573920683	0.992708754	0.89525	0.9409681
50	CNN UNDERSAMPLE	0.725	0.54171744	0.993427641	0.725	0.8374386
1	CNN SMOTE	0.675	0.552218479	0.993565772	0.675	0.8028343
20	CNN UNDERSAMPLE	0.54825	0.524209734	0.993332756	0.54825	0.7051083
1	CNN ADASYN	0.4885	0.587600402	0.993464974	0.4885	0.6525132
100	CNN UNDERSAMPLE	0.40025	0.4855338671	0.99277245	0.40025	0.5684831
0	RF IMABALANCED	0.9975	0.642857143	1	0.28571	0.4444444

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
0	RF SMOTE	0.997	0.642606265	0.666666667	0.28571	0.4
0	XGBOOST IMBALANCED	0.99675	0.642480826	0.571428571	0.28571	0.3809523
0	RF ADASYN	0.99625	0.624623494	0.571428571	0.25	0.3478260
0	DT IMBALANCED	0.99325	0.640724679	0.19047619	0.28571	0.2285714
0	RF OVERSAMPLE	0.99625	0.562374498	0.666666667	0.125	0.2105263
0	XGBOOST ADASYN	0.9925	0.622740964	0.181818182	0.25	0.2105263
0	XGBOOST OVERSAMPLE	0.995	0.561746988	0.25	0.125	0.1666666
0	XGBOOST SMOTE	0.9895	0.638843094	0.111111111	0.28571	0.16
0	KNN IMBALANCE	0.99675	0.535714286	1	0.07143	0.1333333
0	DT SMOTE	0.9835	0.671421403	0.080645161	0.35714	0.1315789
0	DT ADASYN	0.98375	0.618348394	0.070175439	0.25	0.1095890
0	LR ADASYN	0.97525	0.676330321	0.063157895	0.375	0.1081081
0	LR OVERSAMPLE	0.9875	0.589106426	0.075	0.1875	0.1071428
0	DT OVERSAMPLE	0.995	0.53062249	0.166666667	0.0625	0.0909090
0	NB IMBALANCED	0.978	0.633072898	0.048780488	0.28571	0.0833333
0	LR SMOTE	0.977	0.632571142	0.046511628	0.28571	0.08
0	NB OVERSAMPLE	0.96925	0.642193775	0.042735043	0.3125	0.0751879
0	NB ADASYN	0.96475	0.639934739	0.037037037	0.3125	0.0662251
0	NB SMOTE	0.966	0.627051824	0.030769231	0.28571	0.0555555
0	KNN OVERSAMPLE	0.99125	0.52873996	0.047619048	0.0625	0.0540540
0	NB UNDERSAMPLE	0.948	0.65360906	0.024509804	0.35714	0.0458718
0	MLP SMOTE	0.931	0.645079206	0.018382353	0.35714	0.0349650
1	CNN UNDERSAMPLE	0.0115	0.504014049	0.996512349	0.0115	0.0158971
0	MLP ADASYN	0.6615	0.643323293	0.00736377	0.625	0.0145560
0	KNN SMOTE	0.769	0.599383557	0.006507592	0.42857	0.0128205
0	KNN ADASYN	0.76075	0.568649598	0.006295908	0.375	0.0123839
0	MLP OVERSAMPLE	0.5715	0.59814257	0.005820722	0.625	0.0115340
0	RF UNDERSAMPLE	0.68625	0.593452082	0.005577689	0.5	0.0110323
0	KNN UNDERSAMPLE	0.49775	0.570048742	0.004470939	0.64286	0.0088801
0	XGBOOST UNDERSAMPLE	0.4255	0.569385707	0.004340278	0.71429	0.0086281
0	MLP UNDERSAMPLE	0.07775	0.537255394	0.003780718	1	0.0075329
0	DT UNDERSAMPLE	0.6015	0.515339402	0.003768844	0.42857	0.0074719
0	LR IMBALANCED	0.9965	0.5	0	0	0
0	MLP IMBALANCE	0.9965	0.5	0	0	0
0	LR UNDERSAMPLE	0.9965	0.5	0	0	0

Tabla 3.26: Puntuaciones de las pruebas de los modelos ML y DL.

Basados en la tabla 3.26 se obtienen las siguientes observaciones:

- Los modelos DL son superiores en resultados a excepción de CNN con UnderSampler en 1 epoch, donde obtiene resultados inferiores junto a los modelos ML.
- El mejor modelo con respecto a todos los modelos es CNN con datos desbalanceados en 20 epochs.

- El segundo mejor modelo con respecto a todos los modelos es CNN con datos desbalanceados en 50 *epochs*.
- El peor modelo con respecto a los modelos DL es CNN con UnderSampler en 1 *epoch*.
- El peor modelo con respecto a todos los modelos es LR con UnderSampler.
- El mejor modelo con respecto a los modelos ML es RF con datos desbalanceados.
- El segundo mejor modelo con respecto a los modelos ML es RF con SMOTE.
- La estrategia con mejor resultado con respecto a todos los modelos es con datos desbalanceados con el modelo CNN y sus peores resultados con respecto a los modelos ML es MLPC, y con respecto a los modelos DL, se obtienen en los modelos AE, DAE, RNN y LSTM con resultados parejos.
- Las estrategias con los segundos mejores resultados con respecto a todos los modelos es SMOTE y UnderSampler con sus mejores resultados con el modelo BPNN y sus peores resultados con respecto a los modelos ML, KNN y LR respectivamente, y con respecto a los modelos DL, CNN en 1 *epoch*.
- Las estrategias con los cuartos mejores resultados con respecto a todos los modelos es OverSampler y ADASYN con sus mejores resultados con el modelo BPNN y sus peores resultados con respecto a los modelos ML y en general, MLPC y KNN respectivamente, y con respecto a los modelos DL, CNN en 1 *epoch*.
- La estrategia con mejor resultado con respecto a los modelos ML es con datos desbalanceados con su mejor resultado en el modelo RF y su peor resultado en el modelo MLPC.
- La segunda mejor estrategia con respecto a los modelos ML es SMOTE con su mejor resultado en el modelo RF y su peor resultado en el modelo KNN.
- La tercera mejor estrategia con respecto a los modelos ML es ADASYN con su mejor resultado en el modelo RF y su peor resultado en el modelo KNN.
- La cuarta mejor estrategia con respecto a los modelos ML es OverSampler con su mejor resultado en el modelo RF y su peor resultado en el modelo MLP.
- La peor estrategia con respecto a los modelos ML es UnderSampler con su mejor resultado en el modelo NB y su peor resultado en el modelo LR.

Con estas observaciones se concluyen los experimentos para la evaluación de los modelos para la obtención de los mejores. Estos modelos representan patrones para detectar los fraudes de tarjeta de crédito, los cuales pueden ser utilizados en una aplicación para dicho propósito.

3.9 Resultados de la evaluación

Teniendo en cuenta las observaciones obtenidas de los experimentos, se llega a la conclusión de que los modelos DL son ampliamente superiores a los modelos ML a excepción del modelo CNN con UnderSampler en 1 *epoch*. Esto se ve reflejado en el experimento 7 donde se ve claramente el ranking de los resultados, no obstante, en los experimentos anteriores se muestran que los resultados de las métricas son altas. Además, los modelos CNN y BPNN son los mejores en

resultados, incluso el modelo CNN obtiene los mejores resultados con los datos desbalanceados, lo cual demuestra que aprende mejor con los datos desbalanceados.

A pesar de que los modelos CNN y BPNN sean los que mejor resultados obtienen, los demás modelos DL también obtienen buenos resultados con una diferencia de milésimas en *F1 Score*. A pesar de ser iguales aparentemente por sus resultados, la gran diferencia se halla en su entrenamiento que pueden ser visto en el anexo 3.28, donde los modelos ML no tienen sus resultados del entrenamiento. Para lograr alcanzar estos resultados finales fueron necesarios la regulación de parámetros, para ello se realizaron los experimentos 1 y 3.

Como resultado del experimento 1 se obtuvo que la mejor forma de particionar los datos era con 10% de datos para el entrenamiento. No obstante, como se pueden apreciar en los resultados, a medida que disminuye el porcentaje de selección de los datos para el entrenamiento, aumentan los resultados de *F1 Score* y también lo hace la exactitud, pero en el caso de la exactitud, llega a un pico en el 20%, y a partir de ahí empieza a disminuir. Es por ello que se selecciona el 20%, que es el pico de la exactitud, a pesar de que 10% clasifique mejor las detecciones, es menor la exactitud con que detecta el fraude, lo cual no es muy conveniente si se intenta detectar la mayor cantidad posible de fraude.

El experimento 3 fue necesario para la obtención del mejor orden de *dropout* para evitar el *overfitting* de los modelos, de forma tal que no se eliminaran muchos patrones o se repitieran algunos de manera excesiva. Como resultado se obtuvieron tres órdenes como los mejores en el modelo CNN, no obstante, sin importar el orden de los *dropout*, el modelo RNN no establece diferencias en los resultados, por lo cual, pudo ser elegido cualquiera de los tres mejores órdenes obtenidos por CNN.

Luego de ser parametrizado el orden de los *dropouts* y el particionamiento de los datos, se pudo proceder a la selección de los mejores modelos por estrategias, tanto en modelos ML como DL. Es por ello que se realizó el experimento 2 para seleccionar los mejores modelos ML, y los experimentos 4 y 5 para los modelos DL.

En el experimento 2 se pudo evidenciar la superioridad que posee el modelo RF dentro de los modelos ML, y obtuvo los mejores resultados con los datos balanceados. En los modelos ML se pudo evidenciar que los resultados de *F1 Score* en las pruebas no superan el 0.5, lo cual, a pesar de obtener buenos resultados en la exactitud, afecta en gran parte la capacidad de detección satisfactoria de los fraudes y ocasiona mala clasificación, pudiendo clasificar como transacción normal los fraudes. Por lo tanto, se seleccionó los mejores modelos por estrategias para luego ser comparados con los mejores modelos DL y lograr establecer un orden de superioridad entre ambos tipos de modelos.

El experimento 4 se realizó con el objetivo de obtener los mejores modelos por estrategias con respecto a los resultados de las pruebas. Esto se realizó en dos partes, ya que se incluye la variable *epoch*, donde en la primera se obtuvo los mejores modelos por *epochs* con la misma estrategia y en la segunda, las mejores estrategias y *epochs* por modelos. Basado en los resultados de ambas partes se obtuvieron los mejores modelos por estrategias con los *epochs* donde obtienen los mejores resultados. A pesar de esto, hubo la necesidad de realizar un experimento 5 de igual forma, pero con la diferencia de que se realizó la comparación con respecto a los resultados de los entrenamientos, ya que muchos modelos obtuvieron sus resultados similares en las pruebas, sin poder ser diferenciados. Finalmente, basado en los resultados de ambos experimentos se obtienen los mejores modelos DL por estrategias teniendo en cuenta tanto los resultados de las pruebas como los de entrenamiento.

Una vez obtenidos los mejores modelos ML y DL por estrategias se realizó el experimento 6 para la comparación de ambos tipos de modelos. Los resultados obtenidos arrojaron una abrumadora superioridad de los modelos DL sobre los modelos ML con una amplia diferencia en la métrica *F1 Score*. Esto permitió demostrar que los modelos DL clasifican y detectan el fraude con mayor efectividad que los modelos ML. En este experimento quedó el modelo BPNN con ADASYN en 100 *epoch* como el mejor modelo por sus resultados.

Por último, se realizó el experimento 7, donde se realizó la comparación de todos los modelos, tanto ML como DL. En este caso el modelo CNN con datos desbalanceados fue el mejor en resultados en 20 *epochs*, evidenciando que, para este modelo, las estrategias para solucionar el desbalance de la información afectan en sus resultados siendo, a pesar de obtener buenos resultados, inferiores a los demás modelos DL. Por este motivo, el modelo CNN se vuelve el candidato perfecto para detectar el fraude bancario en tiempo real por su gran efectividad en el trabajo con datos desbalanceados.

3.10 Parte 9: Aplicación del conocimiento adquirido

Para aplicar el conocimiento adquirido se decide desarrollar una librería Python, que no es más que un conjunto de funcionalidades empaquetadas que permiten a los usuarios realizar nuevas tareas. Para el desarrollo de esta librería se realizó un modelo de ingeniería de software orientada a la reutilización, ya que es una microarquitectura orientada a componentes. Este enfoque de desarrollo se basa en la existencia de un número significativo de componentes reutilizables, se enfoca en la integración de estos componentes en un sistema, en vez de desarrollarlos desde cero. Este modelo reduce grandemente la cantidad de software a desarrollar, lo que conlleva a la disminución de costos y riesgos, además de una entrega más rápida del software.

La librería propuesta tiene el nombre de CCFD (*Credit Card Fraud Detection*) y su estructura se muestra en la figura 3.4. En ella se puede observar los distintos componentes que forman parte de la librería, para el entendimiento de cada uno se dividirán en tres partes: primero las soluciones al desbalance, segundo los modelos ML y DL, y por último la integración de los componentes.

3.10.1 Componentes de las soluciones al desbalance

Las estrategias que se aplican en la solución son las utilizadas anteriormente en los experimentos realizados. Estas estrategias están implementadas en la librería *imblearn*, siendo un componente reutilizable que va a ser integrado con la librería *sklearn* para dividir los datos en conjuntos de entrenamiento y prueba. Otro componente es *pandas*, muy necesario para la lectura del conjunto de datos, en este caso de un documento *Excel*. Cada componente posee una clase con las siguientes funcionalidades:

$$\text{red_data}(\textit{self}, \textit{csv}, \textit{test_size} = 0.2) \quad (3.1)$$

$$\text{strategy}(\textit{self}) \quad (3.2)$$

$$\text{show_test}(\textit{self}) \quad (3.3)$$

La primera funcionalidad va orientada a la lectura del conjunto de datos y su división en

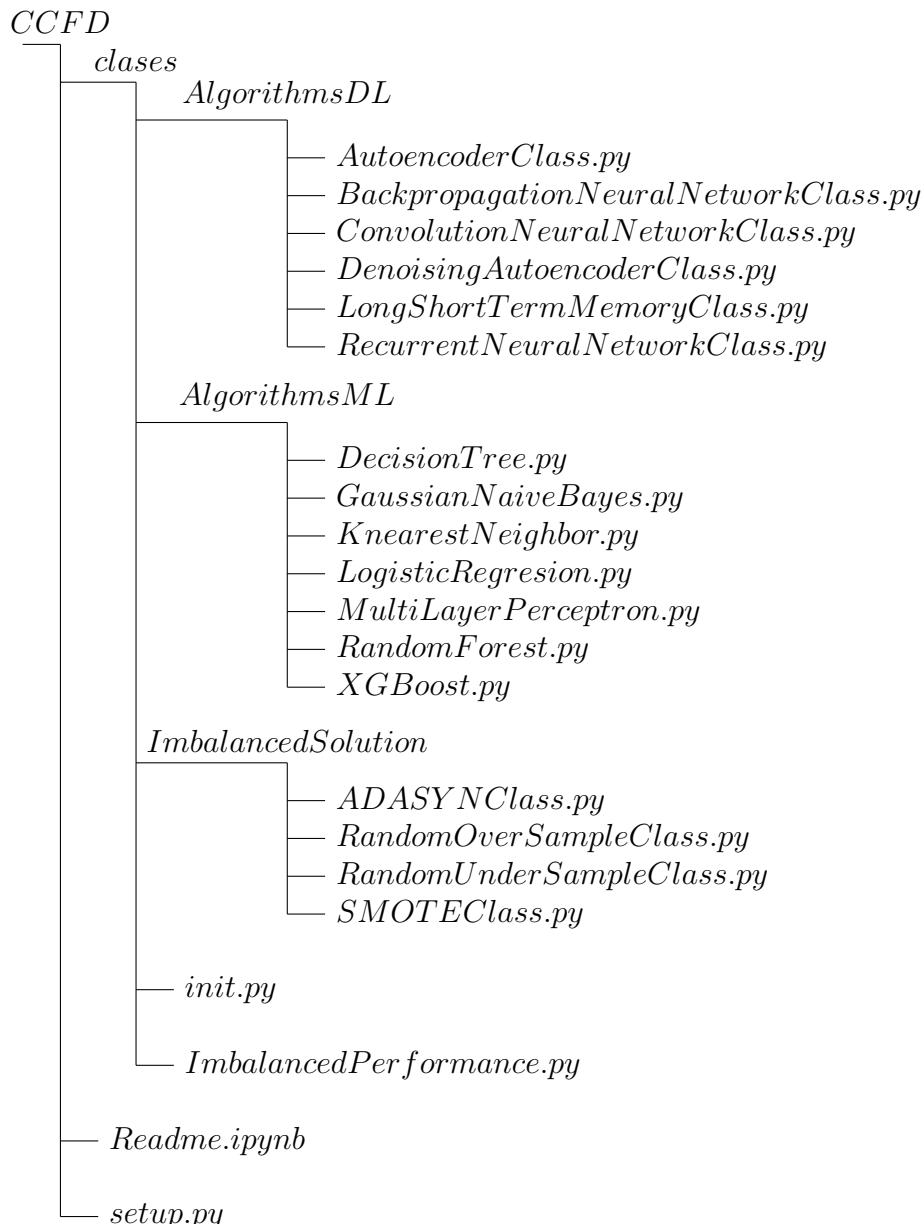


Figura 3.4: Arquitectura de la librería CCFD

los conjuntos de entrenamiento y prueba. Para ello recibe *csv* como la dirección del archivo que almacena el conjunto de datos, y *test_size* como la proporción deseada para el conjunto de prueba con respecto al conjunto general. En la segunda funcionalidad se usa el nombre *strategy* como genérica para referirse a cada una de las estrategias dependiendo del componente. Luego de ser leídos y particionados los datos, esta funcionalidad aplica la estrategia al conjunto de entrenamiento. Y la última funcionalidad, simplemente imprime por consola los *shape* de los conjuntos de entrenamiento y prueba, tanto a los que tienen aplicados la estrategia como los originales.

Estos componentes permiten la obtención de los conjuntos de datos de entrenamiento y prueba con estrategias aplicadas, los cuales serán utilizados en los modelos ML y DL.

3.10.2 Componentes de los modelos ML y DL

Los modelos ML seleccionados ya están implementados en las librerías *sklearn* y *xgboost*, por lo que es un componente que será reutilizados. Se crea un componente por cada modelo ML utilizado en los experimentos anteriores, donde cada uno tendrá las siguientes funcionalidades:

$$\text{Model}(\text{ImbalancedPerformanceClass}) \quad (3.4)$$

$$\text{Modelstrategy}(\text{ImbalancedPerformanceClass}) \quad (3.5)$$

Se usa *Model* como genérico para referenciar a los modelos y *strategy* para referenciar a cada estrategia aplicada. *ImbalancedPerformanceClass* hace referencia al componente que se explicará después, del cual extrae los conjuntos de entrenamiento y prueba de la estrategia seleccionada. En el caso de la primera funcionalidad, se devuelve una lista con un modelo por cada estrategia, siendo un total de 5, y la segunda funcionalidad, devuelve una lista con un único modelo dependiendo de la estrategia especificada. Cada modelo posee la siguiente estructura (*name*, *model*, *X_train*, *y_train*, *X_test*, *y_test*), donde *name* especifica el nombre del modelo con la estrategia, *model* es el modelo, *X_train* y *y_train* es el conjunto de entrenamiento con, *y*, *X_test* y *y_test* es el conjunto de prueba.

Con respecto a los modelos DL, no existe ninguna librería que tenga implementada las arquitecturas de los modelos seleccionados en el capítulo anterior. Es por ello que los componentes de cada modelo DL utiliza las mismas funcionalidades que los componentes de los modelos ML, a diferencia que en los elementos de las listas no existe *model*.

3.10.3 Integración de los componentes

Para la integración de los componentes anteriores se desarrolla el componente *ImbalancedPerformance.py*, donde mediante la clase *ImbalancedPerformanceClass* se utilizan los componentes de soluciones al desbalance, de los modelos ML y DL, además de implementar los modelos DL. Las funcionalidades que posee este componente se pueden resumir en las siguientes:

$$\text{solve_imbalanced}(\textit{self}, \textit{csv}, \textit{test_size} = 0.2) \quad (3.6)$$

$$\text{performanceML}(\textit{self}, \textit{model}) \quad (3.7)$$

$$\text{performanceDL}(\textit{self}, \textit{model}, \textit{dropout} = [0.5, 0.4, 0.3]) \quad (3.8)$$

$$\text{show_comparison}(\textit{self}) \quad (3.9)$$

$$\text{show_comparison_*}(\textit{self}) \quad (3.10)$$

$$(3.11)$$

La primera funcionalidad se encarga de obtener todos los conjuntos de entrenamiento y prueba por cada estrategia, y almacenarlas para su próxima utilización. Los parámetros son los mismos que la funcionalidad *red_data(...)* de los componentes de soluciones al desbalance. La segunda funcionalidad se utiliza para realizar el entrenamiento y las pruebas, además de los resultados de las métricas de los modelos pasados por parámetro en la variable *model*, los cuales son almacenados.

En la tercera funcionalidad se utiliza DL como genérico para referenciar a los modelos DL. A diferencia de la segunda funcionalidad, este posee el parámetro *dropout*, utilizados en algunos modelos para definir el valor que tendrán los *dropouts* en esos modelos, además de que se implementan los modelos. En la cuarta y quinta funcionalidad se crean *Data Frames* a partir de los resultados obtenidos y almacenados de la segunda y tercera funcionalidad. El * hace referencia a variantes de la cuarta funcionalidad, donde se abarcan o difieren en algunos atributos. Esta funcionalidad devuelve dichos *Data Frames*.

De esta forma queda desarrollada generalmente la librería *CCFD*, permitiendo a los usuarios realizar experimentos y pruebas a distintos conjuntos de datos.

Conclusiones generales

Los resultados de este trabajo han sido satisfactorios y cumplen con los objetivos propuestos para solucionar la problemática establecida. Estos junto a sus correspondientes respaldos investigativos, demuestran que los modelos DL poseen un gran resultado en la detección de anomalías, y a su vez, en la detección de fraude en tarjetas de crédito. Son además, la mejor opción cuando se trata de trabajar con *Big Data*, ya que los modelos ML pierden efectividad cuando aumenta el cúmulo de datos que procesa en la detección, creando un sobre-entrenamiento del modelo. El sobre-entrenamiento también puede afectar a los modelos DL, pero en este caso es solucionable con el correcto ajuste de los parámetros, lo que es una de las otras grandes ventajas de estos modelos. También son personalizables, esto se debe a que se le pueden cambiar la cantidad y el tipo de capas que se utilizan para la clasificación, incluso la cantidad de neuronas que posee cada capa. Mediante la experimentación se han podido obtener grandes resultados, los cuales evidencian una gran superioridad de los modelos DL con respecto a los modelos ML. A continuación los principales resultados que se obtuvieron:

- El modelo CNN obtiene mejores resultados con los datos desbalanceados, siendo el más efectivo en la detección de fraude de tarjeta de crédito cuando existe desbalance de la información.
- El modelo BPNN obtiene los mejores resultados en cuanto a las estrategias aplicadas para solucionar el desbalance de la información, a pesar de obtener resultados idénticos con los modelos AE, DAE, RNN y LSTM, este posee mayor precisión.
- En los modelos AE, DAE, RNN y LSTM, se aprecian igualdad en los resultados en cuanto a las pruebas, sin embargo su gran diferencia se puede observar en el entrenamiento de los mismos.
- En cuanto a la división del conjunto de datos, se aprecia que a medida que aumenta el porcentaje de datos para el entrenamiento, mejoran los resultados de *F1 Score*, pero a su vez disminuye la precisión con que se detectan los fraudes. Es por ello es importante encontrar un porcentaje donde ambas métricas se encuentren equilibradas.
- El *overfitting* afecta también a los modelos DL por lo que es necesario aplicar el *dropout regularization*, de esta forma se reajusta el modelo.
- *Random Forest* resultó como el mejor modelo dentro de los modelos ML, obteniendo los mejores resultados en las métricas de estudio.

Los resultados obtenidos en la experimentación definieron las bases para el desarrollo de una librería Python, la cual nos permite ajustar parámetros de los modelos. Esta librería es un proyecto en desarrollo, actualmente solo es ajustable el orden de los *dropouts* y la cantidad de *epochs*. Actualmente la librería permite el uso de los modelos ML y DL utilizados en la experimentación.

Bibliografía

- [1] ECONOMIPEDIA. *Funciones de los bancos - Definición, qué es y concepto.* Accedido 20 de mayo de 2021. <https://economipedia.com/definiciones/funciones-de-los-bancos.html>.
- [2] ILYAS, SADAF, SULTAN ZIA, ZAIB UN NISA, UMAIR MUNEER BUTT, Y SUKUMAR LETCHMUNAN. *Predicting the Future Transaction from Large an Imbalanced Banking Dataset.* IJACSA, 2020.
- [3] TECHNOLOGY PARTNER FOR INNOVATIVE COMPANIES. *Machine Learning in Banking - Opportunities, Risks, Use Cases,* 4 de febrero de 2021. <https://spd.group/machine-learning/machine-learning-in-banking/>.
- [4] TRIVEDI, NARESH, SARITA SIMAIYA, DR KUMAR, Y SANJEEV SHARMA. *An Efficient Credit Card Fraud Detection Model Based on Machine Learning Methods.* MATTER: International Journal of Science and Technology, 1 de enero de 2020.
- [5] LIMA, SERVIO. *Deep Learning for Fraud Detection in the Banking Industry.* Human IST Institute, diciembre de 2018.
- [6] BOUCHER, ÉGLANTINE. *Outlier Detection Methods Applied to Financial Fraud,* 2020.
- [7] BROWNLEE, JASON. *A Gentle Introduction to Imbalanced Classification.* Machine Learning Mastery (blog), 22 de diciembre de 2019. <https://machinelearningmastery.com/what-is-imbalanced-classification/>.
- [8] *Undersampling Algorithms for Imbalanced Classification.* Accedido 5 de julio de 2021. <https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/>.
- [9] BROWNLEE, JASON. *Random Oversampling and Undersampling for Imbalanced Classification.* Machine Learning Mastery (blog), 14 de enero de 2020. <https://machinelearningmastery.com/random-oversampling-and-undersampling-for-imbalanced-classification/>.
- [10] BROWNLEE, JASON. *SMOTE for Imbalanced Classification with Python,* 17 de enero de 2021. <https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/>.
- [11] RUI NIAN. *Fixing Imbalanced Datasets: An Introduction to ADASYN (with code!).* Medium. Accedido 5 de julio de 2021. <https://medium.com/@ruinian/an-introduction-to-adasyn-with-code-1383a5ece7aa>.
- [12] SUN, BO, Y HAIYAN CHEN. *A survey of k Nearest Neighbor Algorithms for Solving the Class Imbalanced Problem,* 17 de enero de 2021.

- [13] *Big Data*. Investopedia. Accedido 27 de mayo de 2021. <https://www.investopedia.com/terms/b/big-data.asp>.
- [14] THUDUMU, SRIKANTH, PHILIP BRANCH, JIONG JIN, Y JUGDUTT (JACK) SINGH. *A Comprehensive Survey of Anomaly Detection Techniques for High Dimensional Big Data*. Journal of Big Data 7, n.o 1 (diciembre de 2020): 42. <https://doi.org/10.1186/s40537-020-00320-x>.
- [15] HUANG, JIAN, JUNYI CHAI, Y STELLA CHO. *Deep learning in finance and banking: A literature review and clasification*, 2020. <https://doi.org/10.1186/s11782-020-00082-6>.
- [16] NGUYEN, THANH THI, HAMMAD TAHIR, MOHAMED ABDELRAZEK, Y ALI BABAR. *Deep Learning Methods for Credit Card Fraud Detection*, s. f., 8.
- [17] MOHAMMED V, RABAT, MOROCCO, IBTISSAM BENCHAJI, SAMIRA DOUZI, Y BOUABID EL OUAHIDI. *Credit Card Fraud Detection Model Based on LSTM Recurrent Neural Networks*. Faculty of Sciences IPSS, University, Journal of Advances in Information Technology 12, n.o 2 (2021): 113-18. <https://doi.org/10.12720/jait.12.2.113-118>.
- [18] NENGCHENG, CHEN, XIONG CHANG, DU WENYING, WANG CHAO, LIN XIN, Y CHEN ZEJIANG. *An Improved Genetic Algorithm Couplin a Back-Propagation Neural Network Model (IGA-BPNN) for Water-Level Predictions*, 30 de junio de 2019.
- [19] *Denoising Autoencoders with Keras, TensorFlow, and Deep Learning*, PyImageSearch, 24 de febrero de 2020. <https://www.pyimagesearch.com/2020/02/24/denoising-autoencoders-with-keras-tensorflow-and-deep-learning/>.
- [20] BROWNLEE, JASON. *Dropout Regularization in Deep Learning Models With Keras*. Machine Learning Mastery (blog), 19 de junio de 2016. <https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/>.
- [21] *Clasificación con datos desbalanceados — Aprende Machine Learning*. Accedido 7 de septiembre de 2021. <https://www.aprendemachinelearning.com/clasificacion-con-datos-desbalanceados/>.
- [22] *Keras: the Python deep learning API*. Accedido 3 de julio de 2021. <https://keras.io/>.
- [23] *Introduction to NumPy*. Accedido 3 de julio de 2021. [https://www.w3schools.com/python\(numpy\)_numpy_intro.asp](https://www.w3schools.com/python(numpy)_numpy_intro.asp).
- [24] *pandas PyPI*. Accedido 3 de julio de 2021. <https://pypi.org/project/pandas/>.
- [25] *scikit-learn: machine learning in Python - scikit-learn 0.24.2 documentation*. Accedido 3 de julio de 2021. <https://scikit-learn.org/stable/>.
- [26] *Matplotlib: Python plotting - Matplotlib 3.4.2 documentation*. Accedido 3 de julio de 2021. <https://matplotlib.org/>.
- [27] *imbalanced-learn documentation - Version 0.8.0*. Accedido 5 de julio de 2021. <https://imbalanced-learn.org/stable/>.
- [28] *PyCharm: the Python IDE for Professional Developers by JetBrains*. Accedido 5 de julio de 2021. <https://www.jetbrains.com/pycharm/>.
- [29] INTELLIGENT INFORMATION TECHNOLOGIES. *10 ventajas de la minería de datos*, 4 de agosto de 2016. <https://itelligent.es/es/10-ventajas-la-mineria-web/>.

- [30] *Sesion5_Metodologias.pdf*. Accedido 3 de julio de 2021. https://disi.unal.edu.co/~eleonguz/cursos/md/presentaciones/Sesion5_Metodologias.pdf.
- [31] MAIMON, ODED, Y LIOR ROKACH, eds. Data Mining and Knowledge Discovery Handbook. Boston, MA: Springer US, 2010. <https://doi.org/10.1007/978-0-387-09823-4>.
- [32] *Redes neuronales con Python*. Accedido 13 de septiembre de 2021. <https://www.cienciadedatos.net/documentos/py35-redes-neuronales-python.html>.
- [33] BUSHAEV, VITALY. *Adam - Latest Trends in Deep Learning Optimization*. Medium, 24 de octubre de 2018. <https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c>.

Anexos

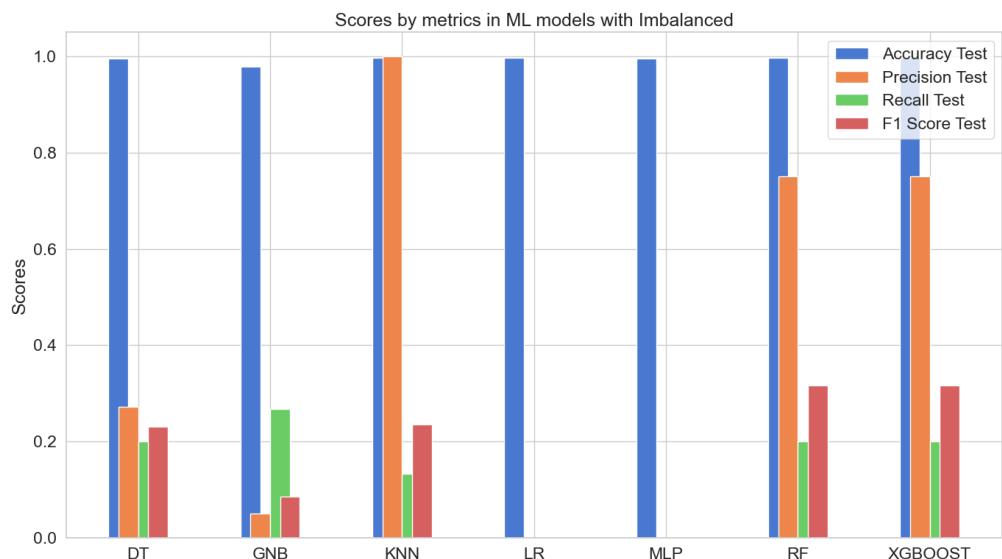


Figura 3.5: Puntuaciones de las pruebas de los modelos ML con datos desbalanceados.

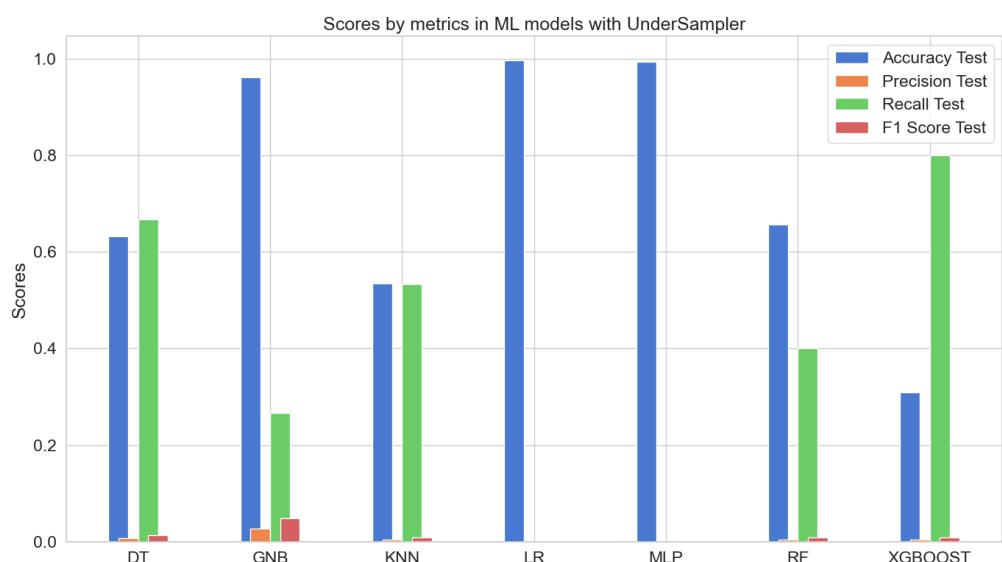


Figura 3.6: Puntuaciones de las pruebas de los modelos ML con UnderSampler.

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
1	CNN IMBALANCE	0.99675	0.566667	0.996761	0.99675	0.99551

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
50	CNN IMBALANCE	0.99675	0.566667	0.996761	0.99675	0.99551
20	CNN IMBALANCE	0.9965	0.566541	0.99551	0.9965	0.995336
100	CNN IMBALANCE	0.99625	0.566416	0.994884	0.99625	0.995167
1	DAE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
50	DAE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
20	AE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
20	DAE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
20	RNN UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
20	LSTM UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
20	BPNN UNDERSAMPLE	0.99625	0.5	0.996264	0.99625	0.994379
1	AE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
1	RNN IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
50	DAE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
50	RNN IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
50	LSTM IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
50	BPNN IMBALANCE	0.99625	0.5	0.996264	0.99625	0.994379
50	AE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
50	RNN UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
20	LSTM IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
50	LSTM UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
50	BPNN UNDERSAMPLE	0.99625	0.5	0.996264	0.99625	0.994379
100	AE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
100	DAE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
100	RNN IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
100	LSTM IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
100	BPNN IMBALANCE	0.99625	0.5	0.996264	0.99625	0.994379
100	AE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
100	DAE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
100	RNN UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
100	LSTM UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
100	BPNN UNDERSAMPLE	0.99625	0.5	0.996264	0.99625	0.994379
20	BPNN IMBALANCE	0.99625	0.5	0.996264	0.99625	0.994379
50	AE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
20	RNN IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
1	LSTM IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
20	DAE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
1	BPNN UNDERSAMPLE	0.99625	0.5	0.996264	0.99625	0.994379
1	LSTM UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
1	DAE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
1	AE UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
1	BPNN IMBALANCE	0.99625	0.5	0.996264	0.99625	0.994379
1	RNN UNDERSAMPLE	0.99625	0.5	0.992514	0.99625	0.994379
20	AE IMBALANCE	0.99625	0.5	0.992514	0.99625	0.994379
100	AE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
100	DAE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
100	RNN OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
1	BPNN OVERSAMPLE	0.99575	0.5	0.995768	0.99575	0.99363

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
100	LSTM OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
1	AE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
1	DAE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
100	BPNN OVERSAMPLE	0.99575	0.5	0.995768	0.99575	0.99363
50	LSTM OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
50	RNN OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
50	DAE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
50	AE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
1	RNN OVERSAMPLE	0.99575	0.5	0.995768	0.99575	0.99363
1	LSTM OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
50	BPNN OVERSAMPLE	0.99575	0.5	0.995768	0.99575	0.99363
20	LSTM OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
20	RNN OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
20	DAE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
20	AE OVERSAMPLE	0.99575	0.5	0.991518	0.99575	0.99363
20	BPNN OVERSAMPLE	0.99575	0.5	0.995768	0.99575	0.99363
100	AE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
20	AE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
20	DAE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
50	BPNN SMOTE	0.9955	0.5	0.99552	0.9955	0.993255
50	LSTM SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
50	RNN SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
50	DAE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
50	AE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
20	RNN SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
20	BPNN SMOTE	0.9955	0.5	0.99552	0.9955	0.993255
20	LSTM SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
100	DAE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
100	LSTM SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
100	BPNN SMOTE	0.9955	0.5	0.99552	0.9955	0.993255
1	BPNN SMOTE	0.9955	0.5	0.99552	0.9955	0.993255
1	LSTM SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
1	AE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
1	RNN SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
100	RNN SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
1	DAE SMOTE	0.9955	0.5	0.99102	0.9955	0.993255
1	DAE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
100	LSTM ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
1	LSTM ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
1	RNN ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
100	RNN ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
100	DAE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
100	AE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
1	BPNN ADASYN	0.99525	0.5	0.995273	0.99525	0.992881
20	BPNN ADASYN	0.99525	0.5	0.995273	0.99525	0.992881
50	BPNN ADASYN	0.99525	0.5	0.995273	0.99525	0.992881
100	BPNN ADASYN	0.99525	0.5	0.995273	0.99525	0.992881

Epoch	Model	Accuracy	AUC	Precision	Recall	F1
20	LSTM ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
1	AE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
50	LSTM ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
50	RNN ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
20	RNN ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
20	DAE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
50	DAE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
50	AE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
20	AE ADASYN	0.99525	0.5	0.990523	0.99525	0.992881
50	CNN SMOTE	0.99225	0.608977	0.992855	0.99225	0.992547
100	CNN SMOTE	0.99125	0.608474	0.992714	0.99125	0.991958
100	CNN ADASYN	0.99125	0.628941	0.992657	0.99125	0.991927
50	CNN ADASYN	0.99	0.628313	0.992506	0.99	0.991192
20	CNN SMOTE	0.98975	0.607721	0.992569	0.98975	0.991101
100	CNN OVERSAMPLE	0.9875	0.700861	0.993864	0.9875	0.990406
50	CNN OVERSAMPLE	0.9865	0.700359	0.993812	0.9865	0.989846
20	CNN OVERSAMPLE	0.98475	0.69948	0.993737	0.98475	0.988879
20	CNN ADASYN	0.97925	0.622913	0.992022	0.97925	0.985314
1	CNN OVERSAMPLE	0.93125	0.701902	0.993469	0.93125	0.960463
100	CNN UNDERSAMPLE	0.76675	0.617273	0.99368	0.76675	0.864517
50	CNN UNDERSAMPLE	0.53675	0.535048	0.993017	0.53675	0.695181
1	CNN SMOTE	0.39575	0.558248	0.992364	0.39575	0.562558
1	CNN ADASYN	0.38025	0.531505	0.991331	0.38025	0.546305
20	CNN UNDERSAMPLE	0.121	0.459222	0.989971	0.121	0.210891
1	CNN UNDERSAMPLE	0.0765	0.536512	0.996265	0.0765	0.135628

Tabla 3.27: Puntuaciones en las pruebas de los modelos DL por estrategias y epochs.

Epoch	Model	Accuracy Train	Precision Train	Recall Train	F1 Train
20	CNN IMBALANCE	0.997437477	0.083916076	0.08041957	0.08158508
50	CNN IMBALANCE	0.998374999	0.181818172	0.16783215	0.17249417
1	CNN IMBALANCE	0.971125007	0.037425533	0.05244755	0.03988132
100	CNN IMBALANCE	0.998562515	0.181818172	0.18181817	0.17948715
50	BPNN SMOTE	0.998307228	0.998056114	0.99864942	0.99829829
100	BPNN SMOTE	0.997523487	0.996707678	0.99837095	0.99745846
20	BPNN SMOTE	0.996896565	0.995942354	0.99777555	0.99675047
100	BPNN UNDERSAMPLE	0.949999988	0.986111104	0.93910253	0.96012986
1	BPNN SMOTE	0.855799377	0.867141366	0.79329419	0.81046265
50	BPNN UNDERSAMPLE	0.800000012	0.706439376	0.50834119	0.58997691
20	BPNN UNDERSAMPLE	0.689999998	0.928571463	0.41259468	0.55927348
50	BPNN IMBALANCE	0.998437524	0.051999997	0.052	0.052
100	BPNN IMBALANCE	0.998437524	0.049999997	0.04899999	0.04933333
20	BPNN IMBALANCE	0.99781251	0.033999994	0.03183333	0.0324
1	BPNN IMBALANCE	0.940437496	0	0	0
1	BPNN UNDERSAMPLE	0.5	0	0	0

Epoch	Model	Accuracy Train	Precision Train	Recall Train	F1 Train
100	LSTM SMOTE	0.994576812	0.993465602	0.99562114	0.99436241
50	LSTM SMOTE	0.990501583	0.987064838	0.99409676	0.9902218
100	RNN SMOTE	0.984608173	0.981494069	0.98768878	0.9840225
50	RNN SMOTE	0.97865206	0.974082828	0.98366153	0.97813684
20	RNN SMOTE	0.97667712	0.971696854	0.98222548	0.97604197
20	LSTM SMOTE	0.974043906	0.965691805	0.98348832	0.97362816
100	RNN UNDERSAMPLE	0.769999981	0.845588207	0.7712912	0.80352724
100	LSTM UNDERSAMPLE	0.680000007	0.829166651	0.64908314	0.71301317
20	DAE UNDERSAMPLE	0.5	0.5546875	1	0.70659566
100	DAE UNDERSAMPLE	0.5	0.5546875	1	0.70659566
1	RNN SMOTE	0.705736697	0.707795322	0.70846796	0.70029038
20	LSTM UNDERSAMPLE	0.660000026	0.683493614	0.69580197	0.6749922
20	RNN UNDERSAMPLE	0.589999974	0.622781396	0.68014705	0.64144778
50	LSTM UNDERSAMPLE	0.660000026	0.64967531	0.67938793	0.63414246
50	RNN UNDERSAMPLE	0.589999974	0.576943278	0.55650157	0.56057417
1	RNN UNDERSAMPLE	0.540000021	0.646139681	0.50062847	0.54794741
50	AE SMOTE	0.497680247	0.401092619	0.80377835	0.53134143
1	LSTM SMOTE	0.611347973	0.680008471	0.46954575	0.52970928
1	LSTM UNDERSAMPLE	0.49000001	0.453070164	0.52777779	0.48501226
1	DAE SMOTE	0.498307198	0.318110585	0.63791376	0.42133522
1	AE UNDERSAMPLE	0.460000008	0.2890625	0.5	0.35873014
100	AE SMOTE	0.500658333	0.290432364	0.46473235	0.34405035
50	DAE UNDERSAMPLE	0.439999998	0.2265625	0.5	0.31111109
1	AE SMOTE	0.495485902	0.245143235	0.45525572	0.30565432
20	DAE SMOTE	0.498463959	0.198380858	0.39819458	0.26273957
1	DAE UNDERSAMPLE	0.419999987	0.1640625	0.49999997	0.24444443
20	AE SMOTE	0.494451404	0.153185934	0.31193581	0.20388429
100	DAE SMOTE	0.498369902	0.13457337	0.27081245	0.17848259
50	DAE SMOTE	0.500376165	0.111038476	0.22166499	0.14697345
100	LSTM IMBALANCE	0.997687519	0.027999999	0.026	0.02666667
50	LSTM IMBALANCE	0.997500002	0.021999998	0.022	0.022
50	RNN IMBALANCE	0.997562528	0.021999998	0.021	0.02133333
100	RNN IMBALANCE	0.997250021	0.017999997	0.017	0.01733333
20	LSTM IMBALANCE	0.997187495	0.015999999	0.016	0.016
20	RNN IMBALANCE	0.997250021	0.011999999	0.011	0.01133333
1	RNN IMBALANCE	0.966437519	0.000417647	0.006	0.00077991
1	LSTM IMBALANCE	0.99356252	0.000133333	0.002	0.00025
50	DAE IMBALANCE	0.996874988	0	0	0
1	DAE IMBALANCE	0.996874988	0	0	0
20	DAE IMBALANCE	0.996874988	0	0	0
20	AE IMBALANCE	0.996874988	0	0	0
50	AE IMBALANCE	0.996874988	0	0	0
100	AE IMBALANCE	0.996874988	0	0	0
100	DAE IMBALANCE	0.996874988	0	0	0
1	AE IMBALANCE	0.961937487	0	0	0
20	AE UNDERSAMPLE	0.5	0	0	0
50	AE UNDERSAMPLE	0.5	0	0	0

Epoch	Model	Accuracy Train	Precision Train	Recall Train	F1 Train
100	AE UNDERSAMPLE	0.5	0	0	0
50	BPNN OVERSAMPLE	0.998746216	0.998128951	0.99945122	0.99874675
100	BPNN OVERSAMPLE	0.997868598	0.996402919	0.99924409	0.99773329
100	BPNN ADASYN	0.997837484	0.997854769	0.997711	0.99769962
50	BPNN ADASYN	0.997367322	0.996994555	0.9978323	0.99731481
20	BPNN ADASYN	0.994170547	0.990253687	0.99605656	0.99295169
20	BPNN OVERSAMPLE	0.977965117	0.997234166	0.9583956	0.97664672
1	BPNN OVERSAMPLE	0.848921776	0.847519338	0.79990423	0.80967408
1	BPNN ADASYN	0.853605807	0.853892446	0.7777186	0.7936157
100	LSTM OVERSAMPLE	0.997649193	0.99662441	0.99878359	0.99761117
100	LSTM ADASYN	0.996395767	0.995962143	0.99701226	0.99636859
50	LSTM OVERSAMPLE	0.993229687	0.989385605	0.99745667	0.99317294
50	LSTM ADASYN	0.99200803	0.989960134	0.99422622	0.99179375
100	RNN OVERSAMPLE	0.991035581	0.988565743	0.99359387	0.9907304
50	RNN OVERSAMPLE	0.987995207	0.986146092	0.98979628	0.98748618
20	LSTM OVERSAMPLE	0.986522079	0.97854352	0.9948802	0.98612052
100	RNN ADASYN	0.981947541	0.978451014	0.98589307	0.98150223
20	RNN OVERSAMPLE	0.980880141	0.975126803	0.9871127	0.98038369
50	RNN ADASYN	0.978782058	0.976029992	0.98226827	0.97843897
20	LSTM ADASYN	0.97126025	0.960935235	0.98374456	0.97121501
20	RNN ADASYN	0.970476687	0.964555323	0.97599018	0.96923673
1	RNN OVERSAMPLE	0.686841786	0.692915559	0.66565263	0.67027813
1	RNN ADASYN	0.672328949	0.674672723	0.66558403	0.66144168
100	AE OVERSAMPLE	0.505328476	0.477624118	0.94979382	0.63110417
50	DAE OVERSAMPLE	0.498338759	0.424950272	0.85155469	0.56324184
1	LSTM ADASYN	0.606763422	0.641127884	0.50184727	0.54669023
1	LSTM OVERSAMPLE	0.627883673	0.723023474	0.451469	0.53486472
100	DAE ADASYN	0.498166561	0.352663666	0.70741481	0.46731645
1	AE ADASYN	0.498511285	0.347468525	0.68203992	0.45665944
100	AE ADASYN	0.498918742	0.341819048	0.68436873	0.45252213
1	DAE ADASYN	0.495314509	0.299286067	0.6032064	0.39746156
20	DAE ADASYN	0.49948287	0.286990643	0.5741483	0.37982234
50	AE ADASYN	0.499012768	0.281594425	0.56412828	0.3731164
1	AE OVERSAMPLE	0.500313461	0.267389953	0.52516413	0.35001945
20	AE OVERSAMPLE	0.497837275	0.259196967	0.5205617	0.34358042
20	DAE OVERSAMPLE	0.498683542	0.255109072	0.51153463	0.33792847
50	DAE ADASYN	0.497978508	0.251168996	0.504008	0.33288077
20	AE ADASYN	0.498417288	0.20095399	0.39409533	0.26037276
1	DAE OVERSAMPLE	0.498182058	0.183143184	0.36810431	0.24292144
100	DAE OVERSAMPLE	0.502068698	0.15198721	0.30190572	0.20082469
50	AE OVERSAMPLE	0.504262805	0.157284349	0.05726714	0.04862355
50	CNN SMOTE	0.992194355	0.99048835	0.99384516	0.99208343
100	CNN OVERSAMPLE	0.99730444	0.996242702	0.99834925	0.99726689
100	CNN ADASYN	0.994139194	0.992498457	0.99586558	0.99411225
50	CNN ADASYN	0.991976678	0.991332591	0.99265498	0.99191338
50	CNN OVERSAMPLE	0.995267034	0.99372077	0.99681431	0.99522328
100	CNN SMOTE	0.993134797	0.99086076	0.99567538	0.9931922

Epoch	Model	Accuracy Train	Precision Train	Recall Train	F1 Train
20	CNN OVERSAMPLE	0.992477417	0.990783691	0.99415046	0.99240476
20	CNN SMOTE	0.988056421	0.984934807	0.99140549	0.98805213
20	CNN ADASYN	0.988685846	0.98493588	0.99253625	0.98860687
1	CNN OVERSAMPLE	0.837606549	0.840575874	0.83115494	0.83343488
50	CNN UNDERSAMPLE	0.870000005	0.862745106	0.88	0.87128705
1	CNN SMOTE	0.862601876	0.861973703	0.8608681	0.85896647
20	CNN UNDERSAMPLE	0.829999983	0.811320782	0.86000001	0.8349514
1	CNN ADASYN	0.865985513	0.869674802	0.86216491	0.86288518
100	CNN UNDERSAMPLE	0.939999998	0.958333313	0.92000002	0.93877548
0	RF IMABALANCED	0	0	0	0
0	RF SMOTE	0	0	0	0
0	XGBOOST IMBALANCED	0	0	0	0
0	RF ADASYN	0	0	0	0
0	DT IMBALANCED	0	0	0	0
0	RF OVERSAMPLE	0	0	0	0
0	XGBOOST ADASYN	0	0	0	0
0	XGBOOST OVERSAMPLE	0	0	0	0
0	XGBOOST SMOTE	0	0	0	0
0	KNN IMBALANCE	0	0	0	0
0	DT SMOTE	0	0	0	0
0	DT ADASYN	0	0	0	0
0	LR ADASYN	0	0	0	0
0	LR OVERSAMPLE	0	0	0	0
0	DT OVERSAMPLE	0	0	0	0
0	NB IMBALANCED	0	0	0	0
0	LR SMOTE	0	0	0	0
0	NB OVERSAMPLE	0	0	0	0
0	NB ADASYN	0	0	0	0
0	NB SMOTE	0	0	0	0
0	KNN OVERSAMPLE	0	0	0	0
0	NB UNDERSAMPLE	0	0	0	0
0	MLP SMOTE	0	0	0	0
1	CNN UNDERSAMPLE	0.50999999	0.506493509	0.77999997	0.61417317
0	MLP ADASYN	0	0	0	0
0	KNN SMOTE	0	0	0	0
0	KNN ADASYN	0	0	0	0
0	MLP OVERSAMPLE	0	0	0	0
0	RF UNDERSAMPLE	0	0	0	0
0	KNN UNDERSAMPLE	0	0	0	0
0	XGBOOST UNDERSAMPLE	0	0	0	0
0	MLP UNDERSAMPLE	0	0	0	0
0	DT UNDERSAMPLE	0	0	0	0
0	LR IMBALANCED	0	0	0	0
0	MLP IMBALANCE	0	0	0	0
0	LR UNDERSAMPLE	0	0	0	0

Epoch	Model	Accuracy Train	Precision Train	Recall Train	F1 Train
-------	-------	----------------	-----------------	--------------	----------

Tabla 3.28: Puntuaciones del entrenamiento de los modelos DL y ML.

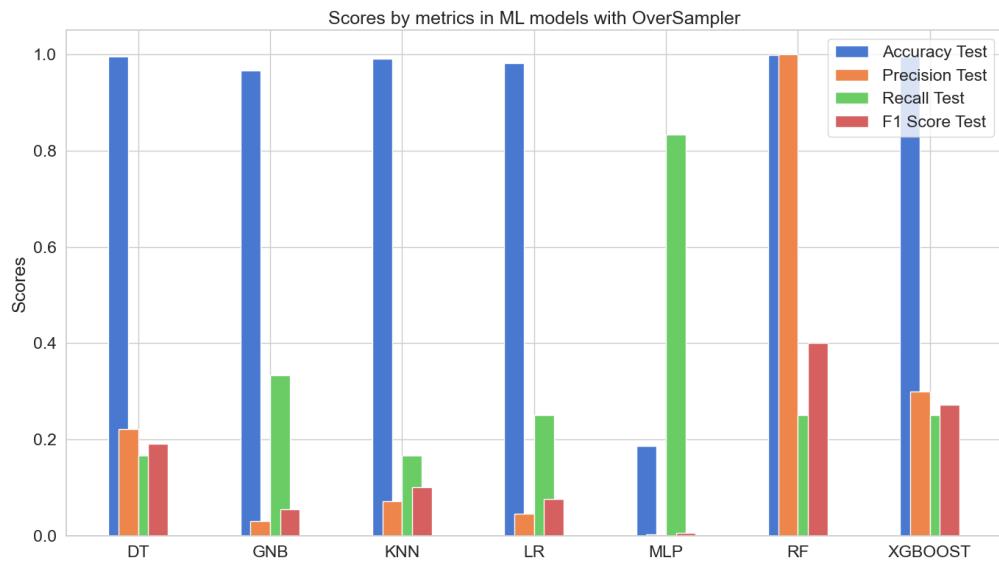


Figura 3.7: Puntuaciones de las pruebas de los modelos ML con OverSampler.

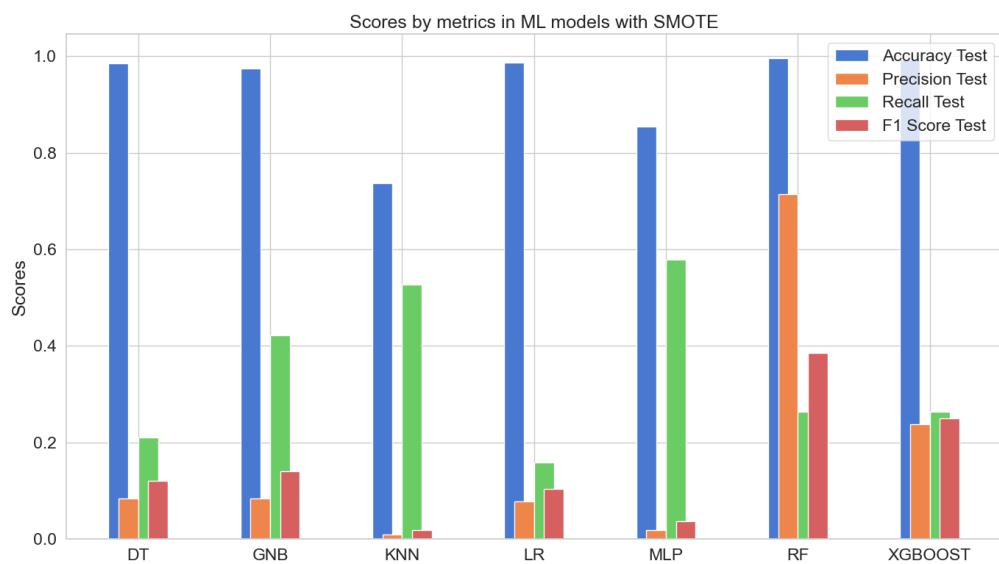


Figura 3.8: Puntuaciones de las pruebas de los modelos ML con SMOTE.

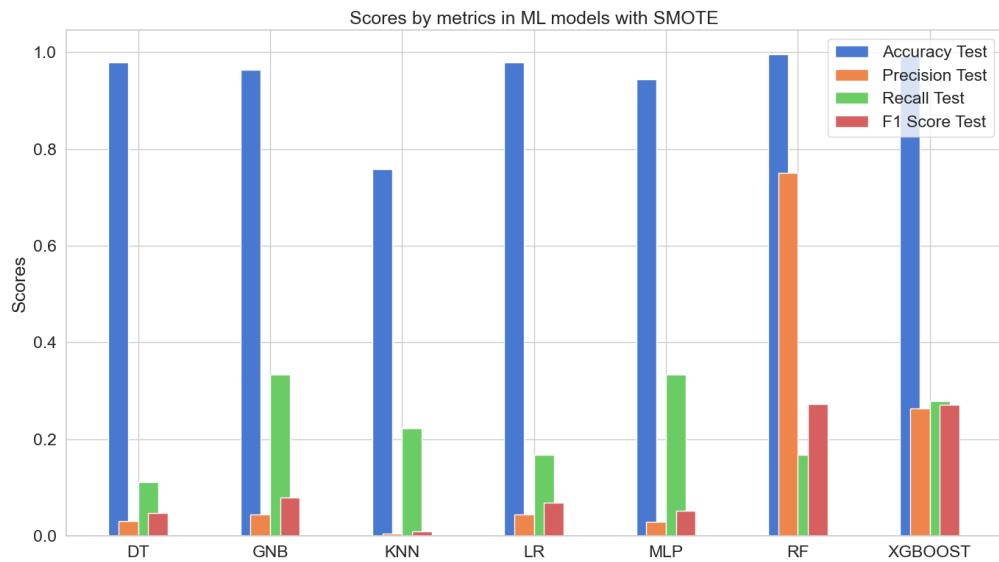


Figura 3.9: Puntuaciones de las pruebas de los modelos ML con ADASYN.

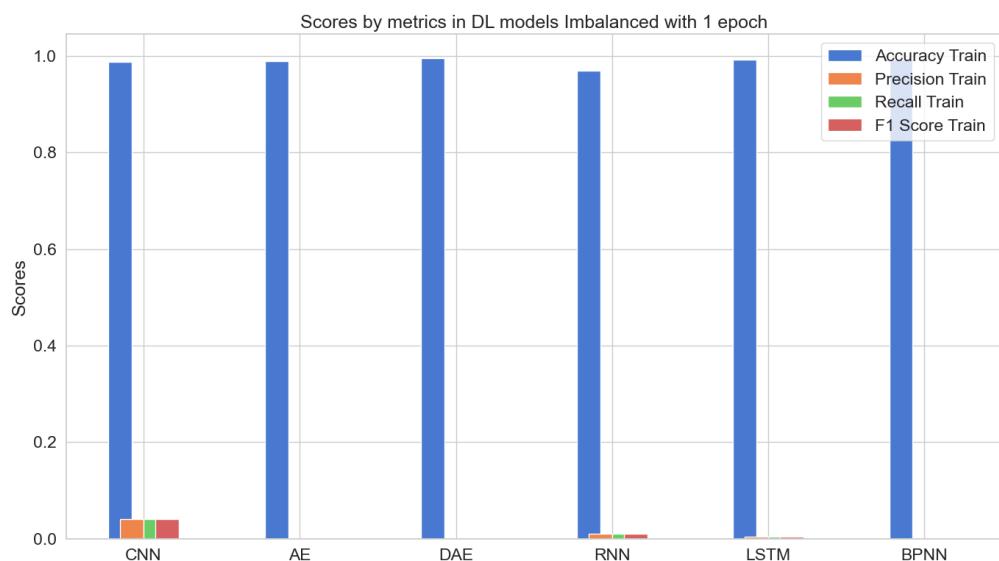


Figura 3.10: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 1 epoch.

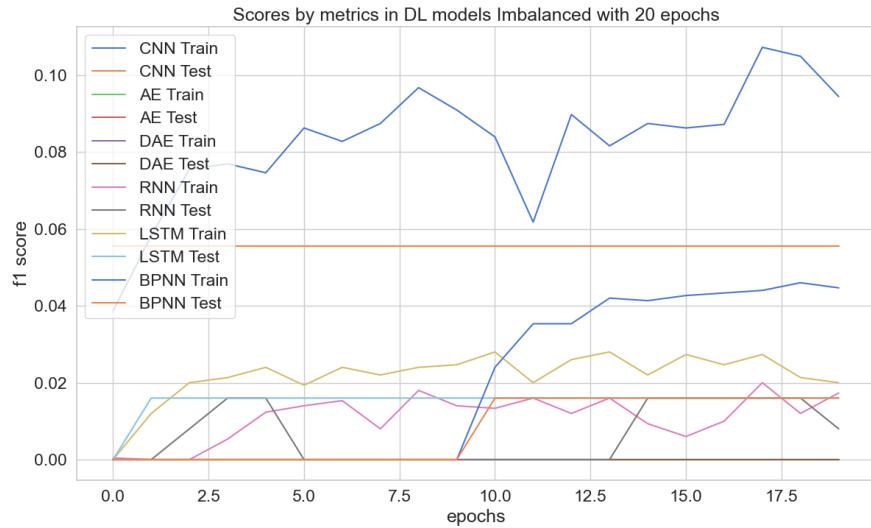


Figura 3.11: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 20 epochs.

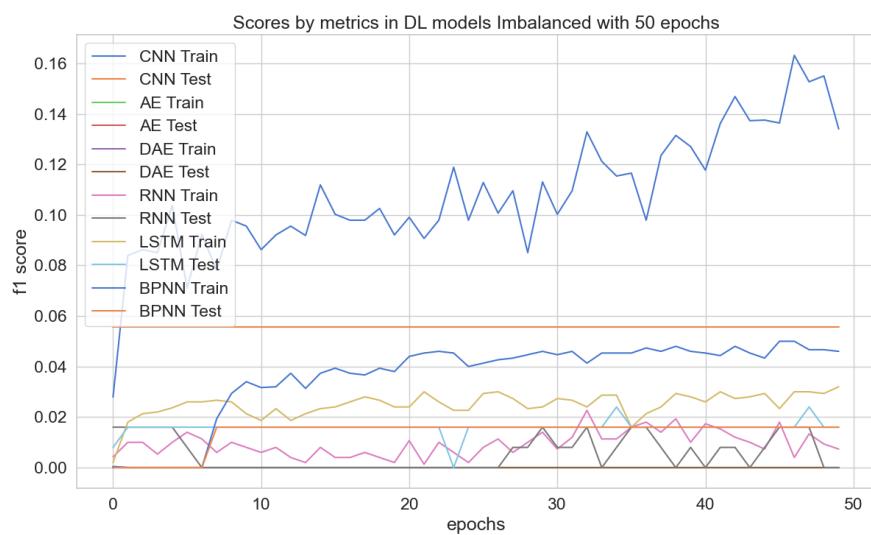


Figura 3.12: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 50 epochs.

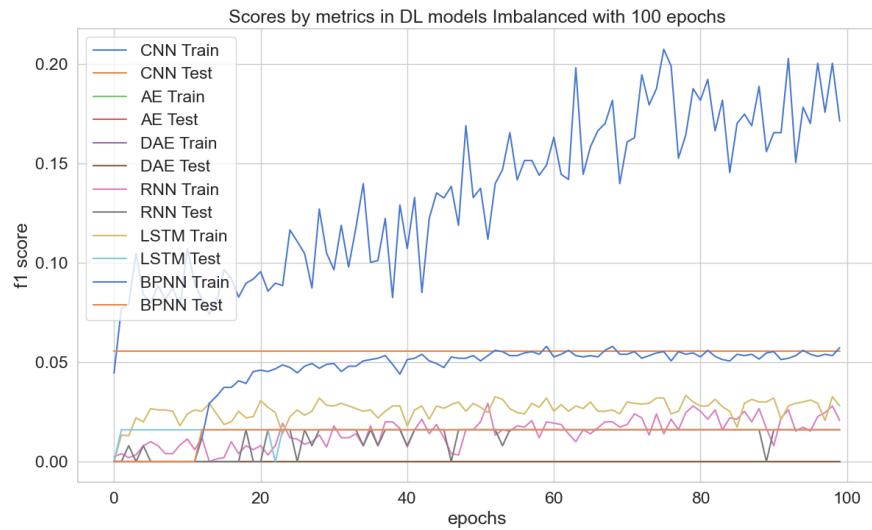


Figura 3.13: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 100 epochs.

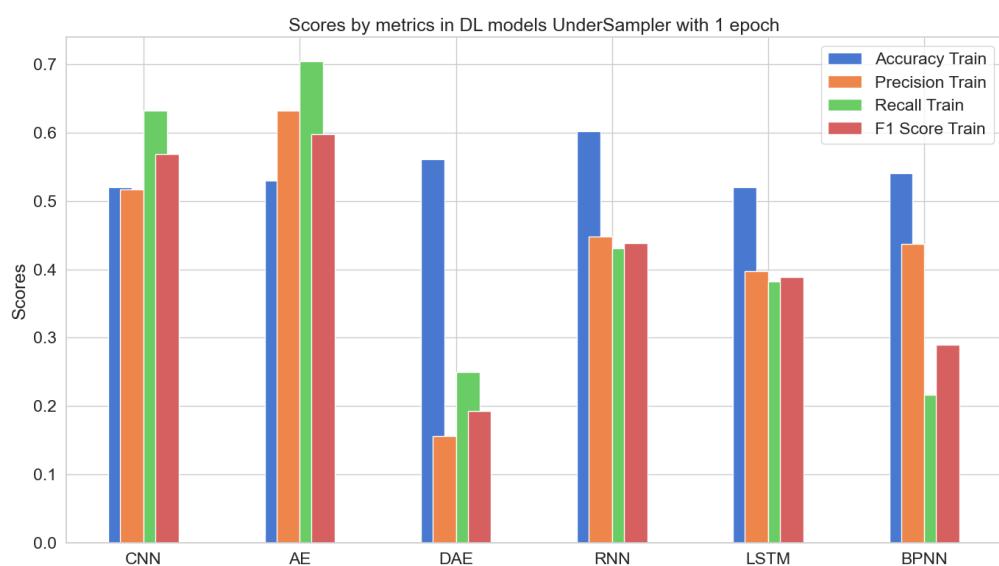


Figura 3.14: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 1 epoch.

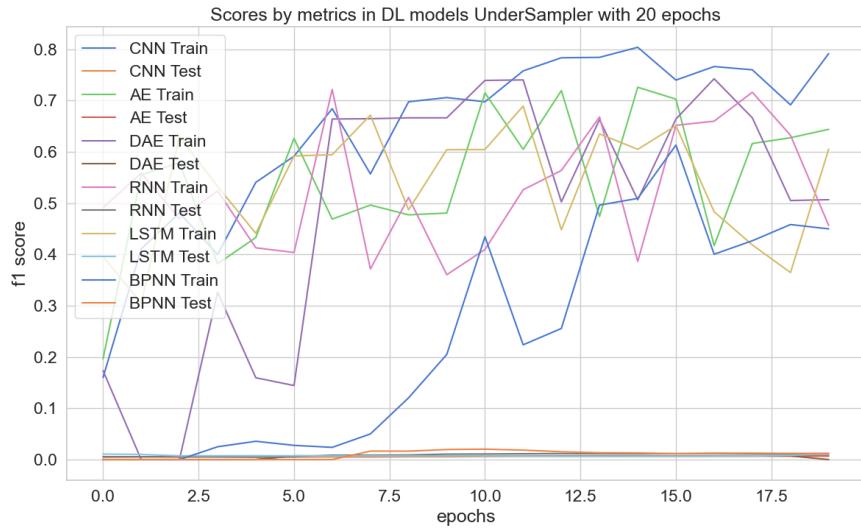


Figura 3.15: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 20 epochs.

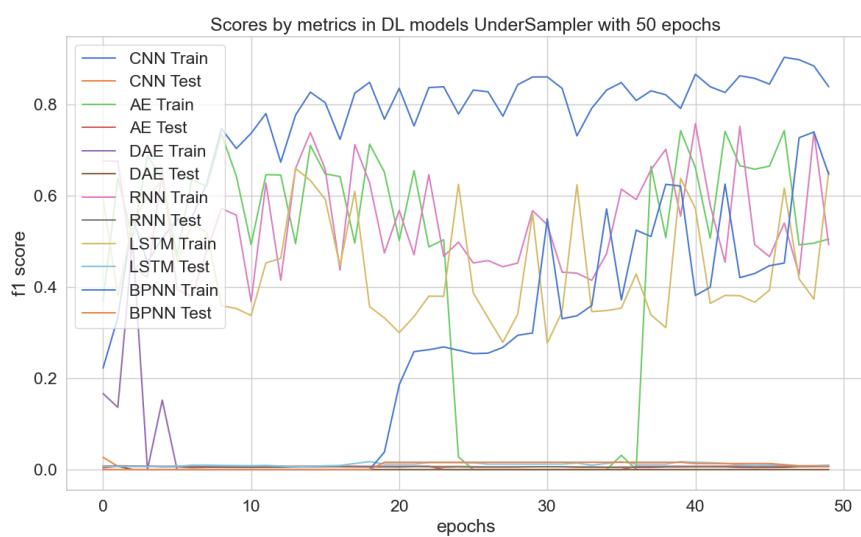


Figura 3.16: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 50 epochs.

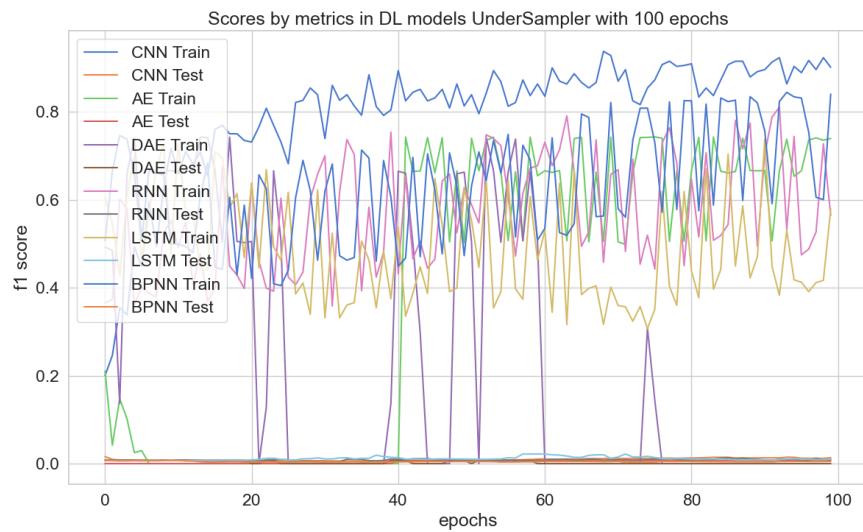


Figura 3.17: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 100 epochs.

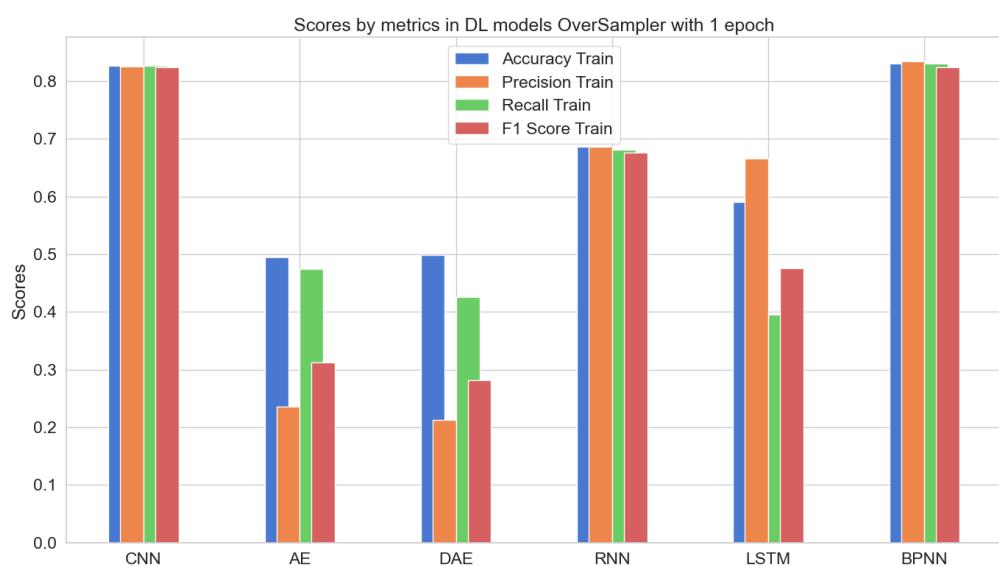


Figura 3.18: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 1 epoch.

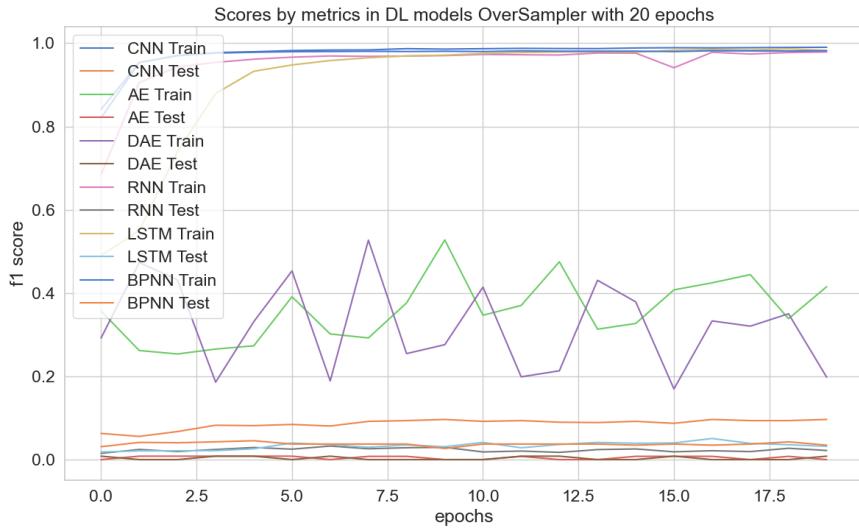


Figura 3.19: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 20 epochs.

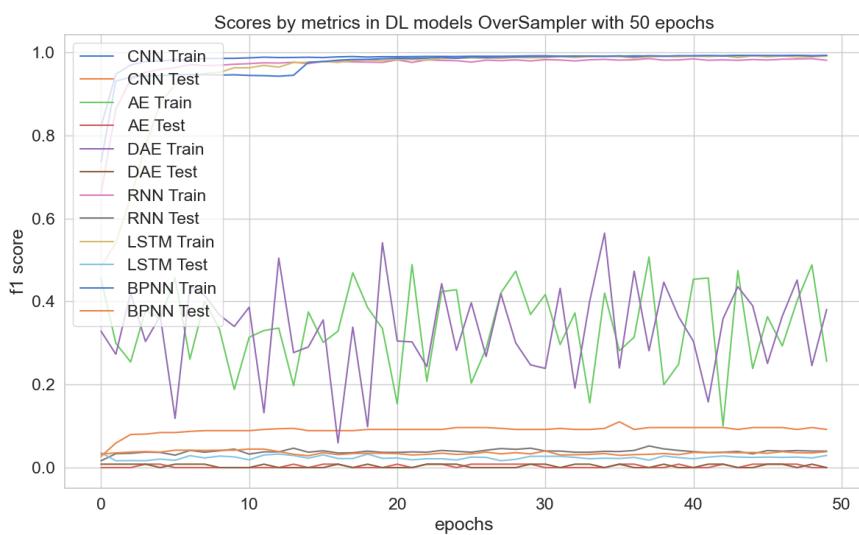


Figura 3.20: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 50 epochs.

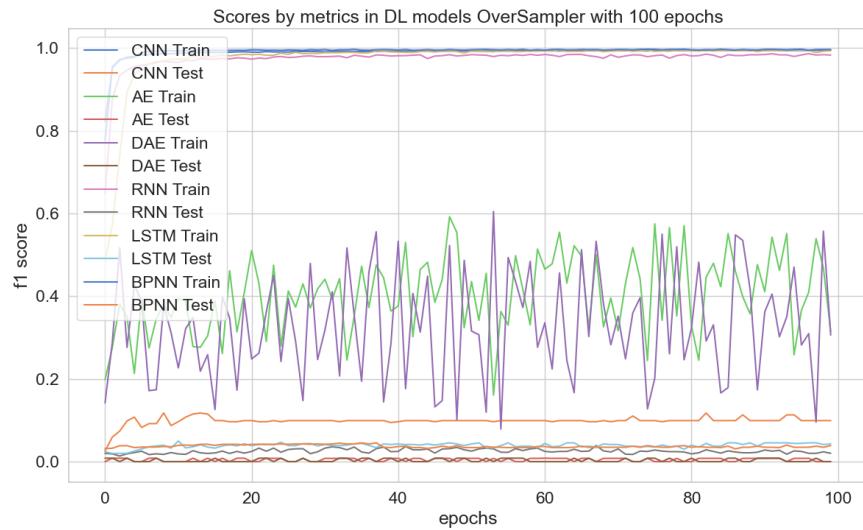


Figura 3.21: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 100 epochs.

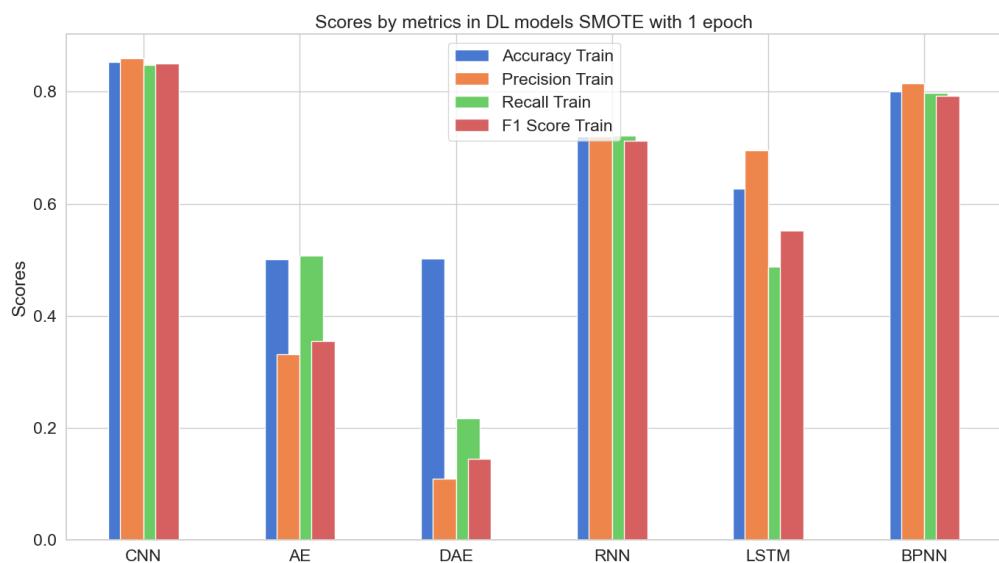


Figura 3.22: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 1 epoch.

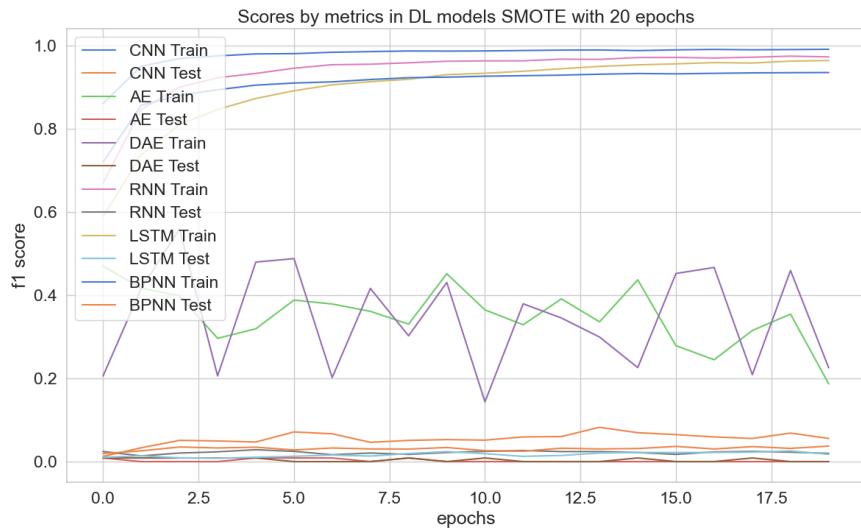


Figura 3.23: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 20 epochs.

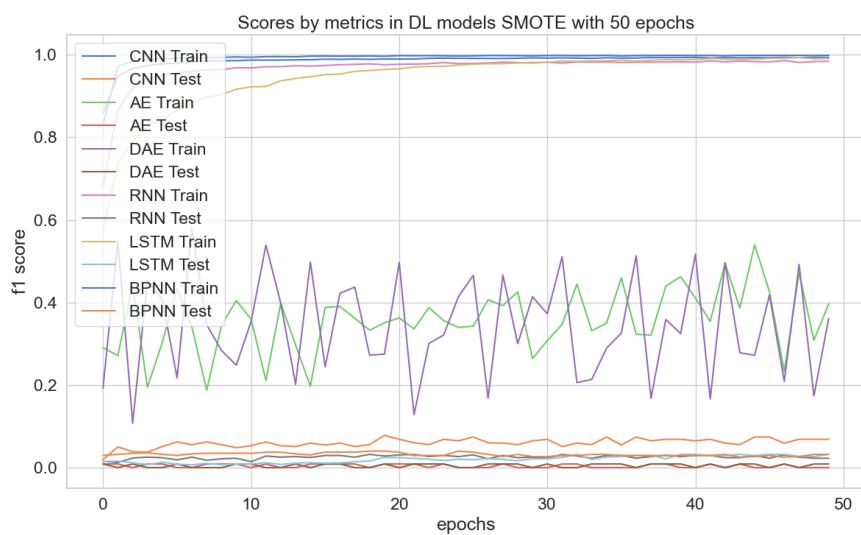


Figura 3.24: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 50 epochs.

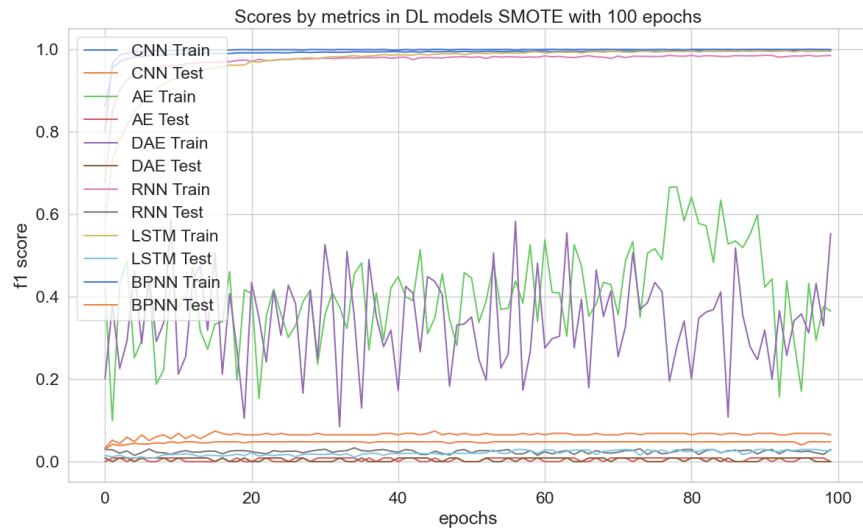


Figura 3.25: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 100 epochs.

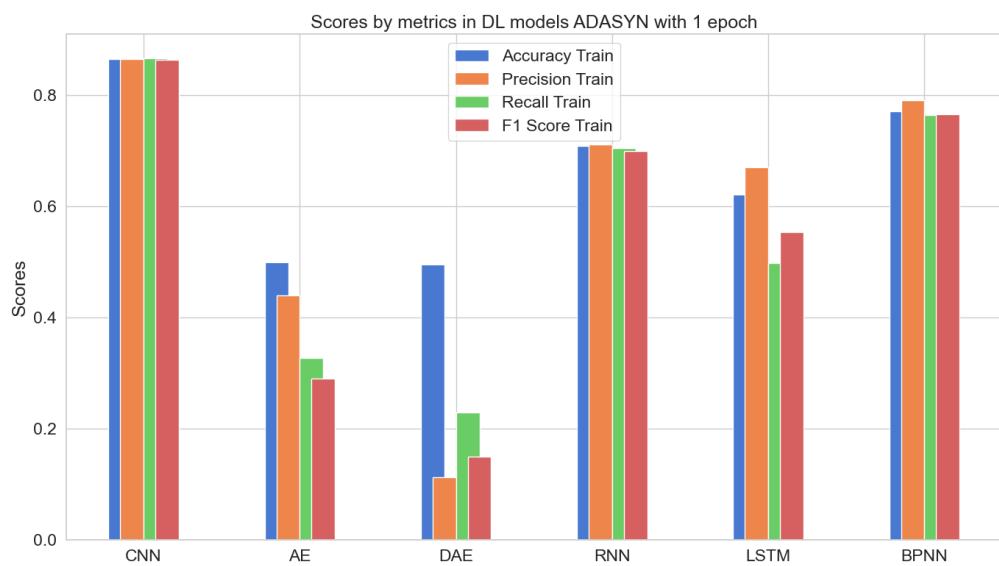


Figura 3.26: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 1 epoch.

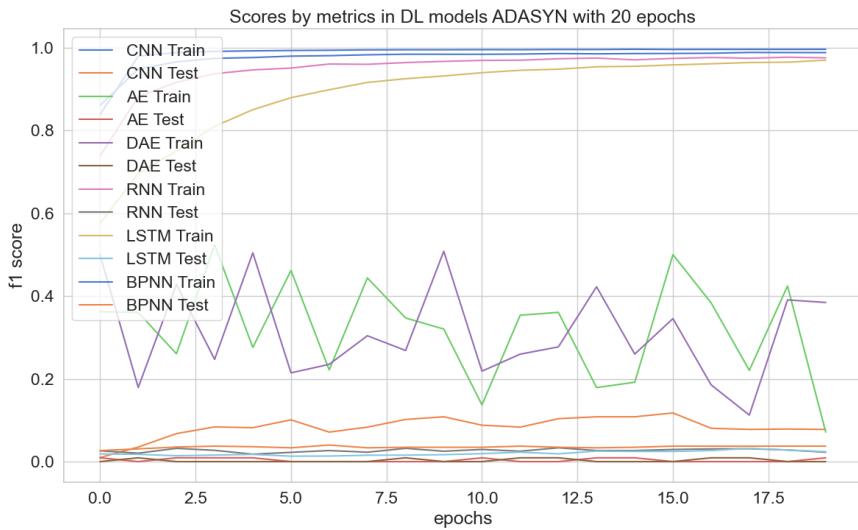


Figura 3.27: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 20 epochs.

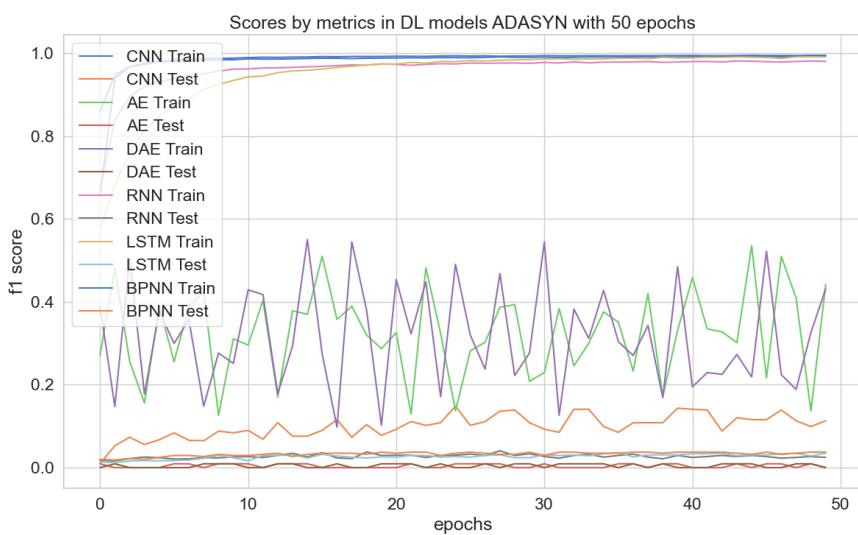


Figura 3.28: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 50 epochs.

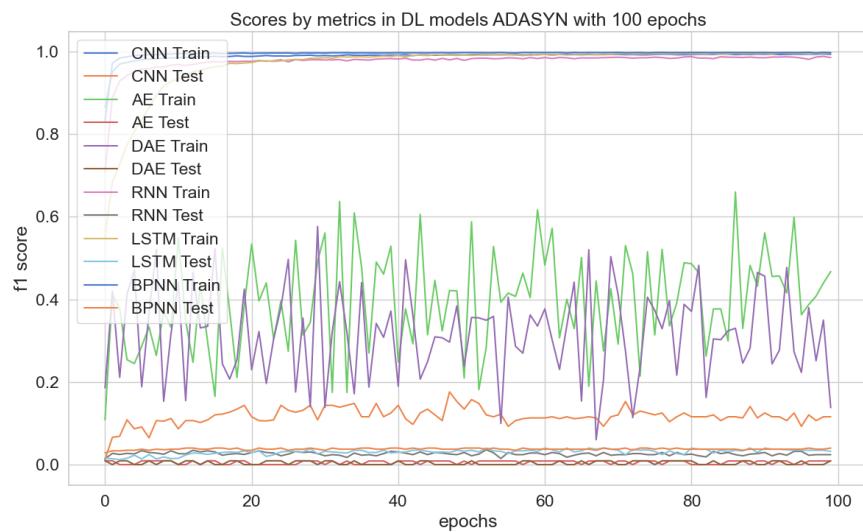


Figura 3.29: Puntuaciones del entrenamiento de los modelos DL con datos desbalanceados en 100 epochs.

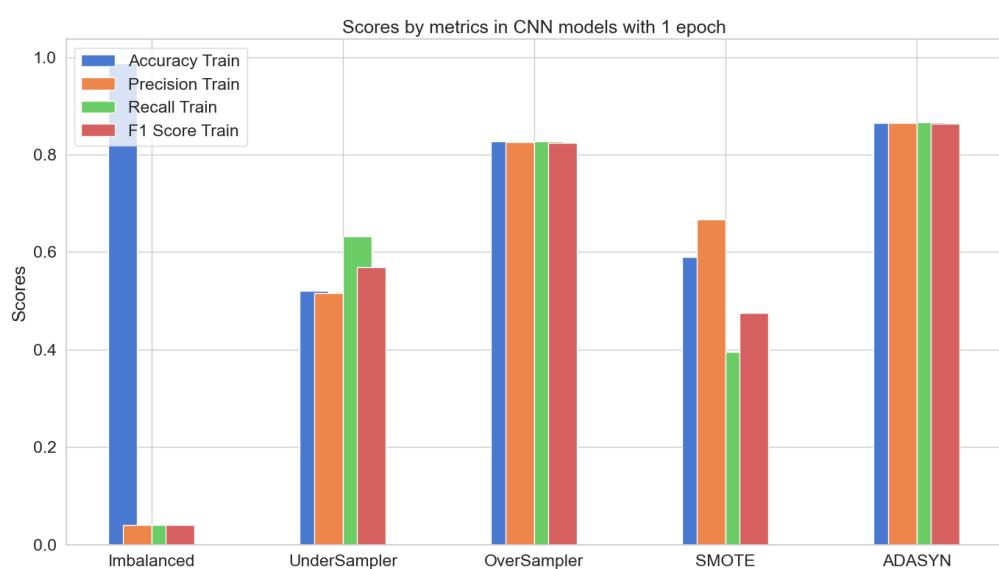


Figura 3.30: Puntuaciones del entrenamiento del modelo CNN por estrategias en 1 epoch.

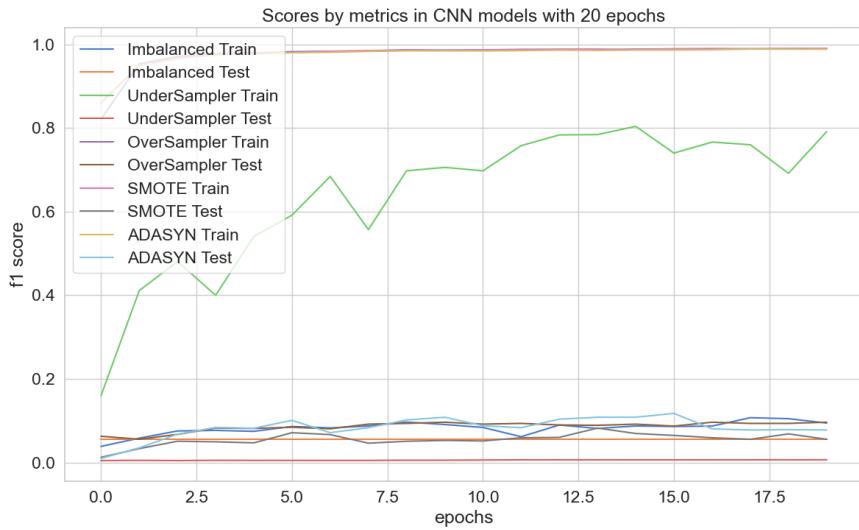


Figura 3.31: Puntuaciones del entrenamiento del modelo CNN por estrategias en 20 epochs.

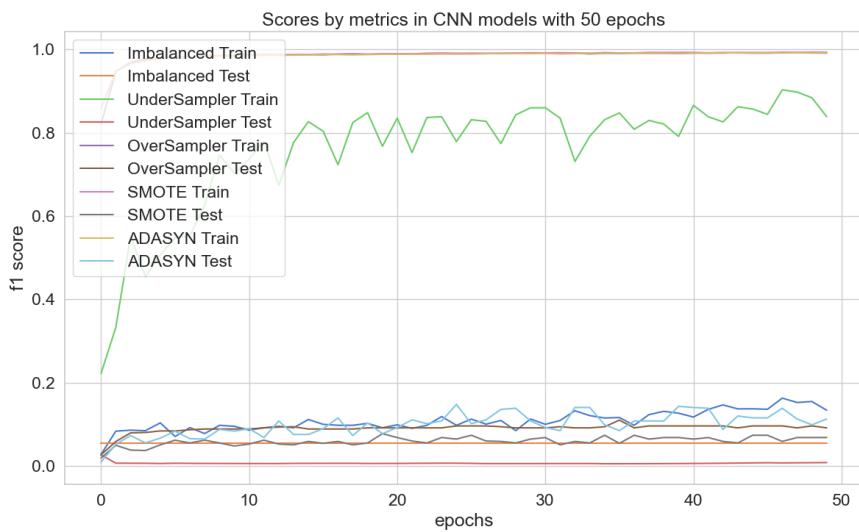


Figura 3.32: Puntuaciones del entrenamiento del modelo CNN por estrategias en 50 epochs.

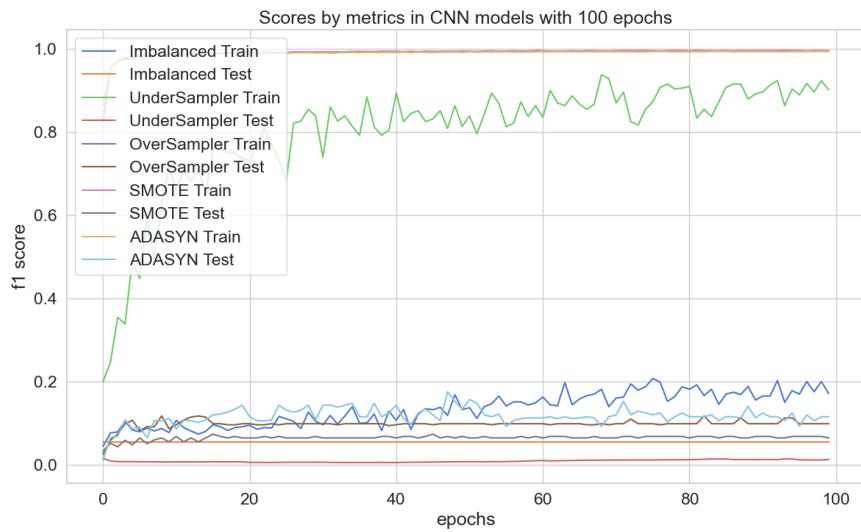


Figura 3.33: Puntuaciones del entrenamiento del modelo CNN por estrategias en 100 epochs.

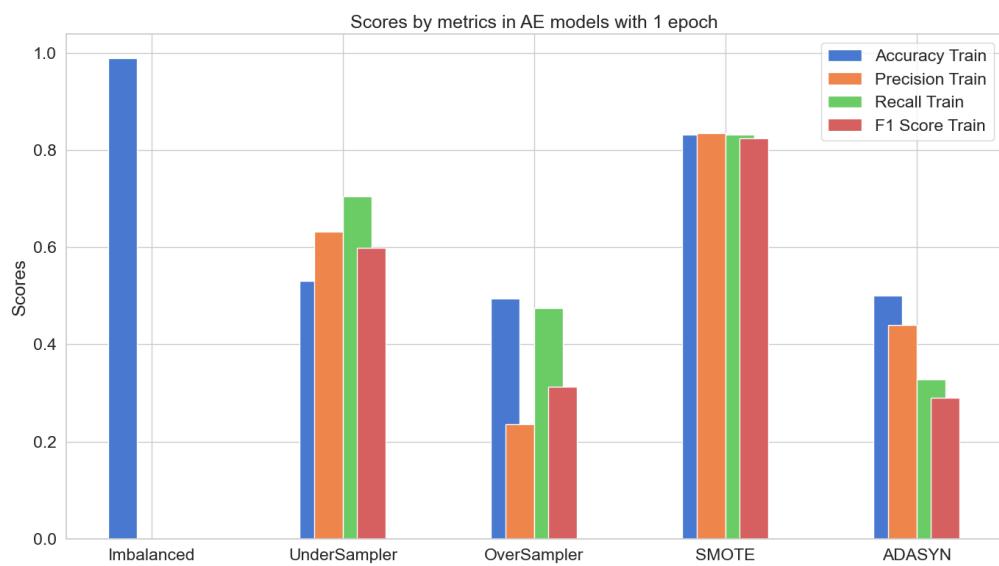


Figura 3.34: Puntuaciones del entrenamiento del modelo AE por estrategias en 1 epoch.

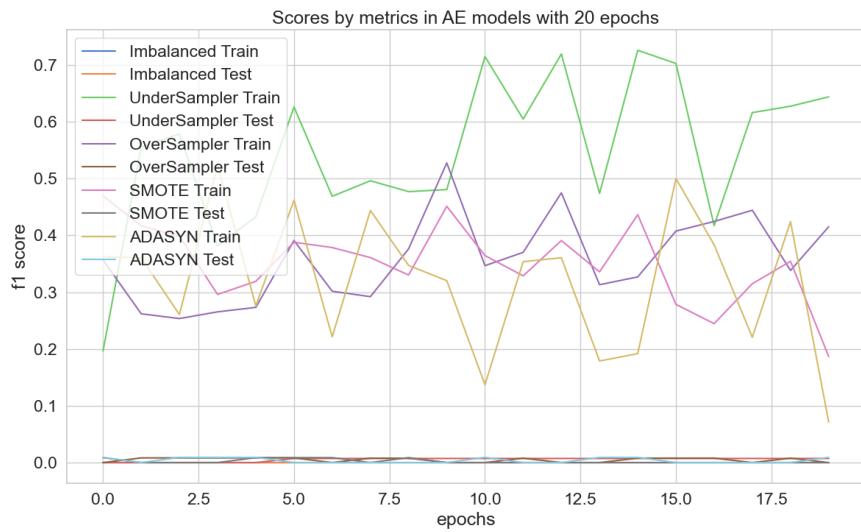


Figura 3.35: Puntuaciones del entrenamiento del modelo AE por estrategias en 20 epochs.

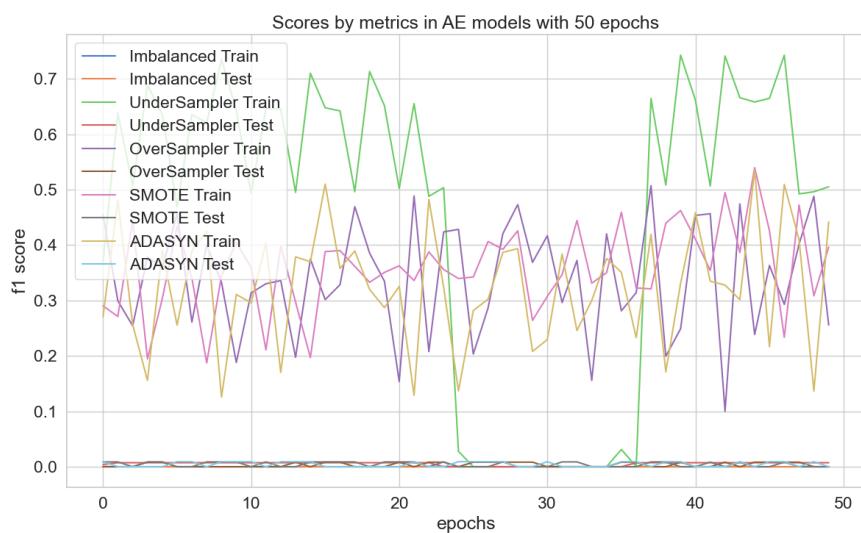


Figura 3.36: Puntuaciones del entrenamiento del modelo AE por estrategias en 50 epochs.

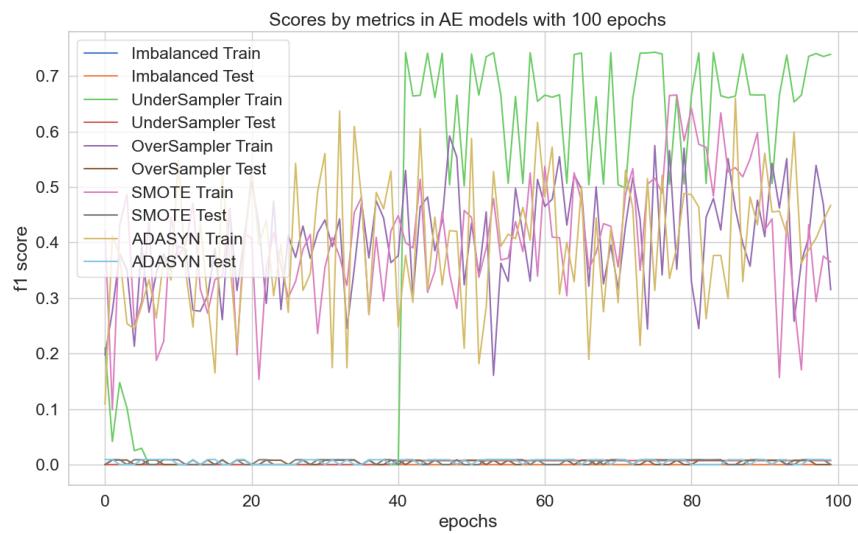


Figura 3.37: Puntuaciones del entrenamiento del modelo AE por estrategias en 100 epochs.

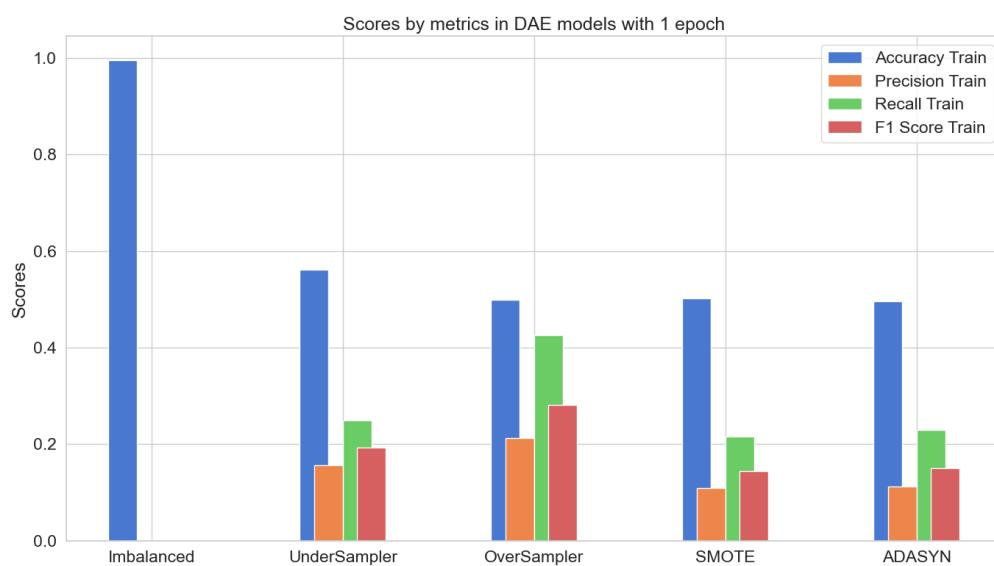


Figura 3.38: Puntuaciones del entrenamiento del modelo DAE por estrategias en 1 epoch.

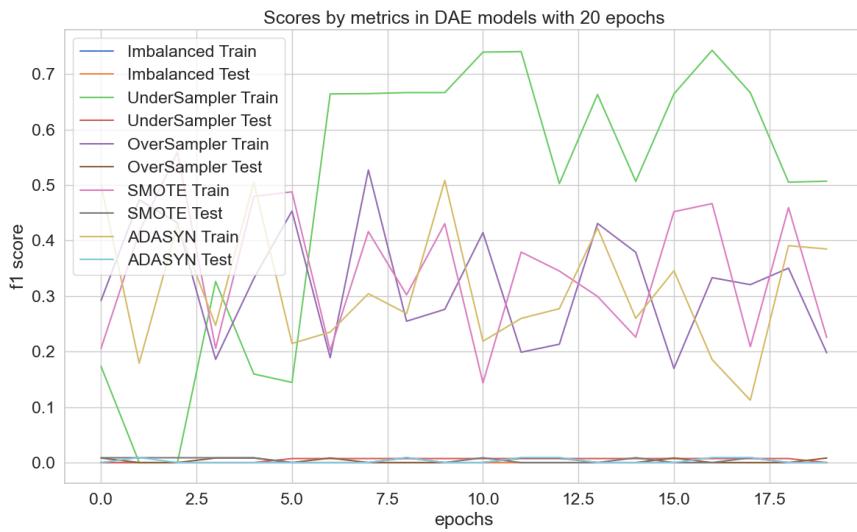


Figura 3.39: Puntuaciones del entrenamiento del modelo DAE por estrategias en 20 epochs.

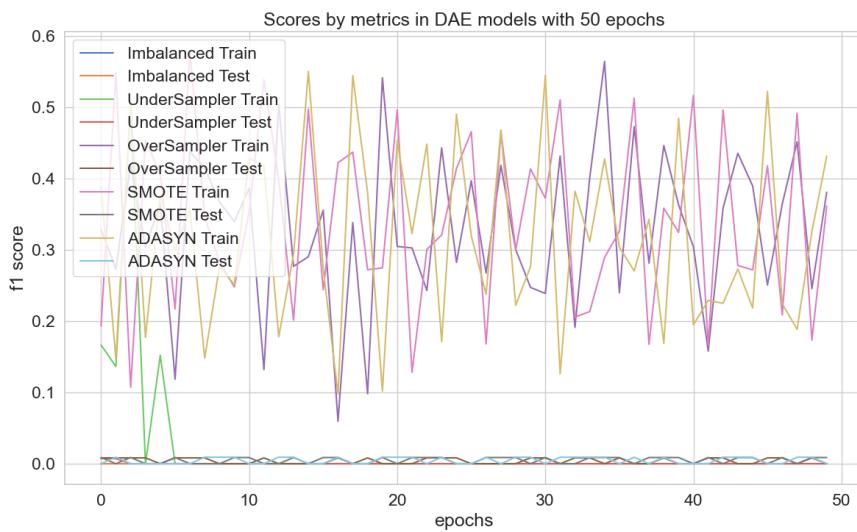


Figura 3.40: Puntuaciones del entrenamiento del modelo DAE por estrategias en 50 epochs.

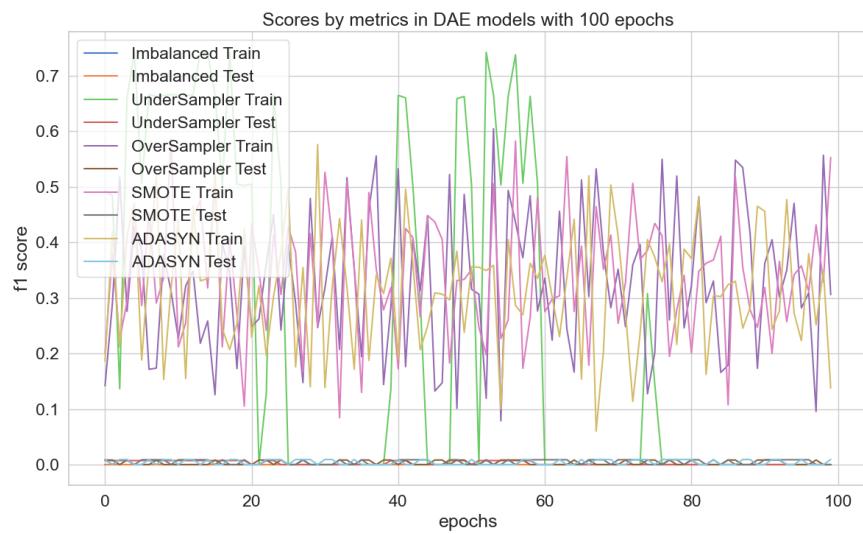


Figura 3.41: Puntuaciones del entrenamiento del modelo DAE por estrategias en 100 epochs.

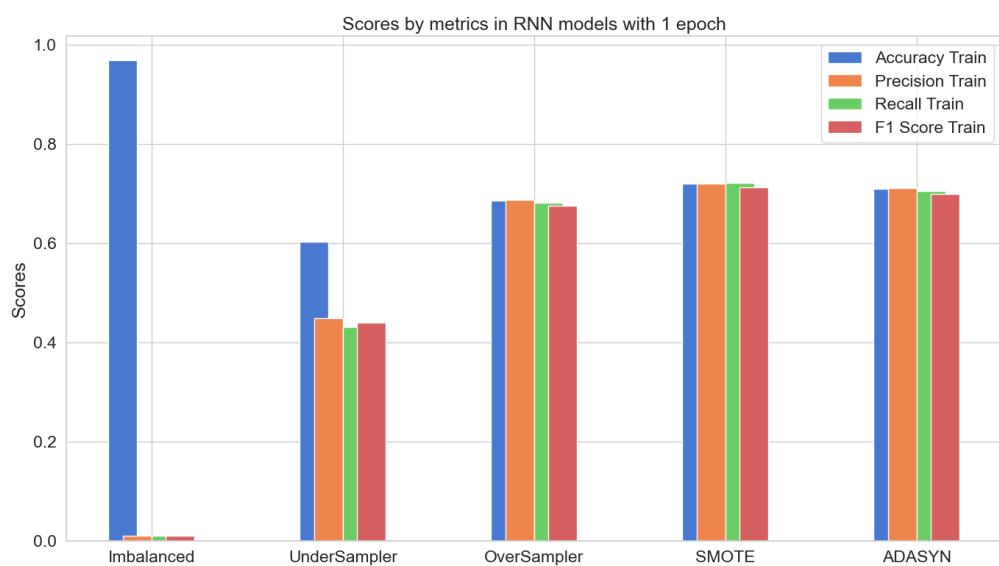


Figura 3.42: Puntuaciones del entrenamiento del modelo RNN por estrategias en 1 epoch.

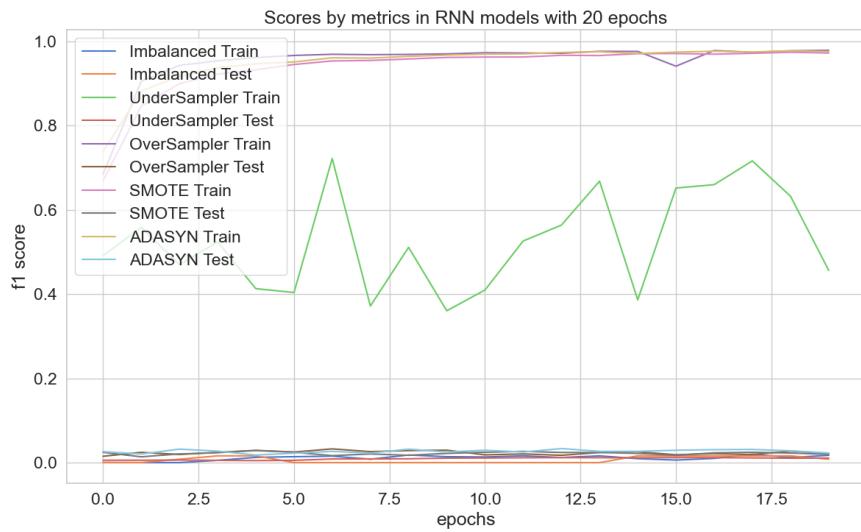


Figura 3.43: Puntuaciones del entrenamiento del modelo RNN por estrategias en 20 epochs.

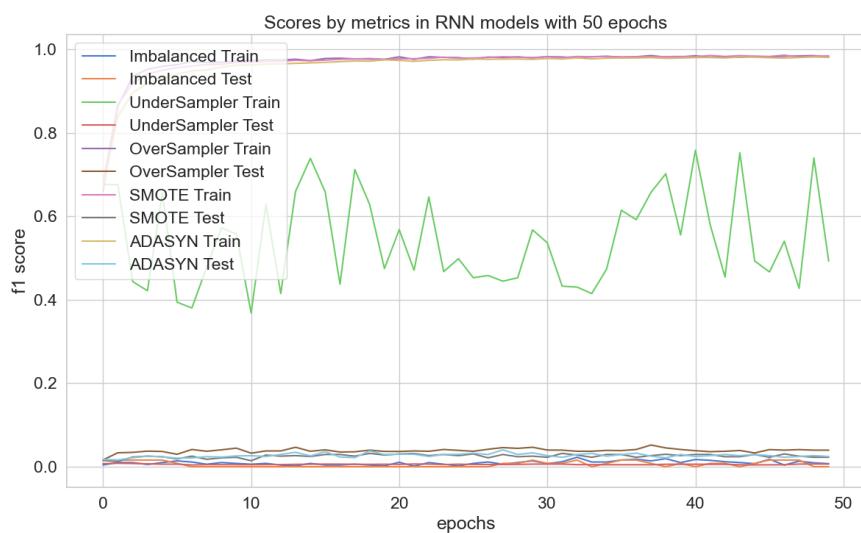


Figura 3.44: Puntuaciones del entrenamiento del modelo RNN por estrategias en 50 epochs.

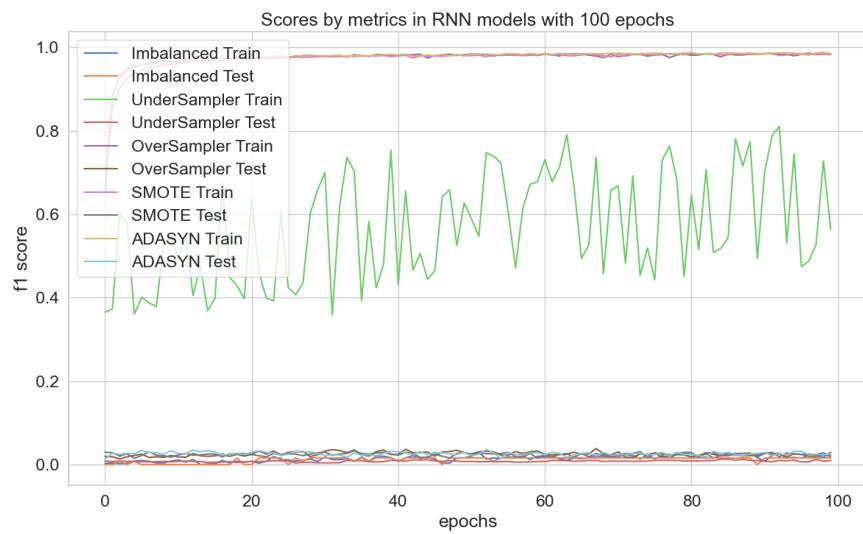


Figura 3.45: Puntuaciones del entrenamiento del modelo RNN por estrategias en 100 epochs.

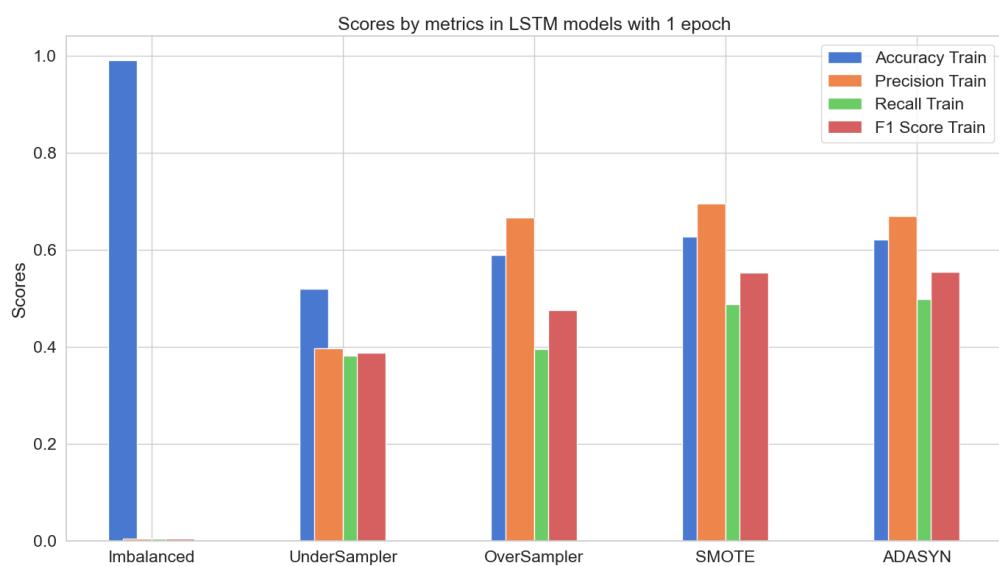


Figura 3.46: Puntuaciones del entrenamiento del modelo LSTM por estrategias en 1 epoch.

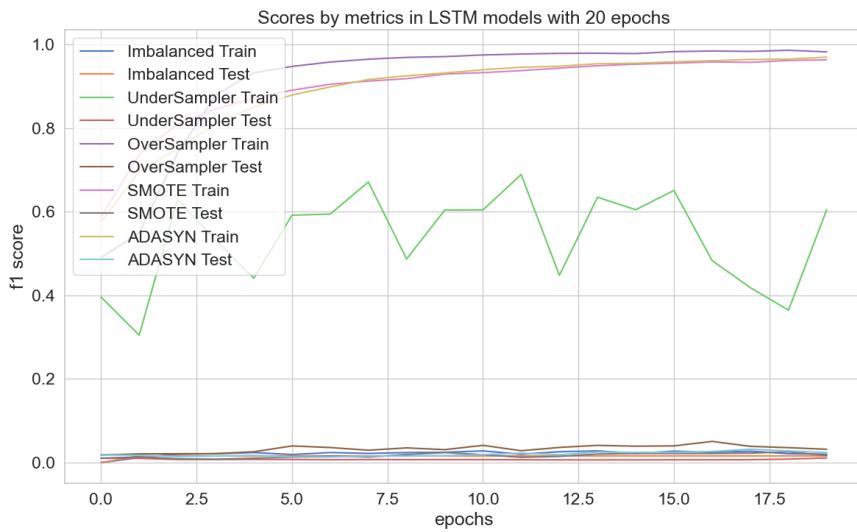


Figura 3.47: Puntuaciones del entrenamiento del modelo LSTM por estrategias en 20 epochs.

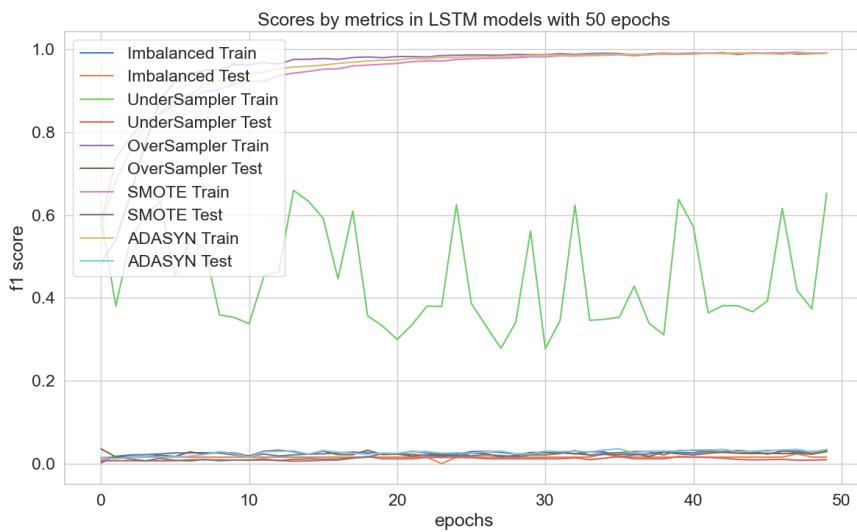


Figura 3.48: Puntuaciones del entrenamiento del modelo LSTM por estrategias en 50 epochs.

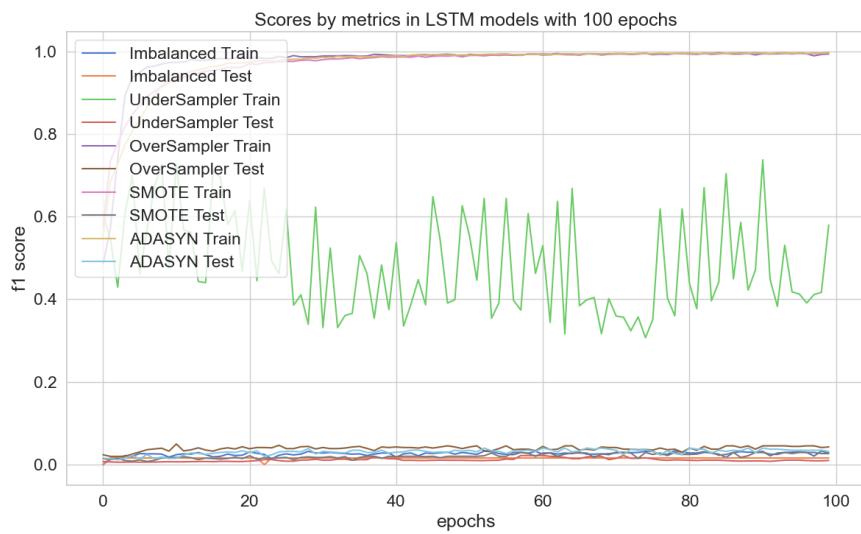


Figura 3.49: Puntuaciones del entrenamiento del modelo LSTM por estrategias en 100 epochs.

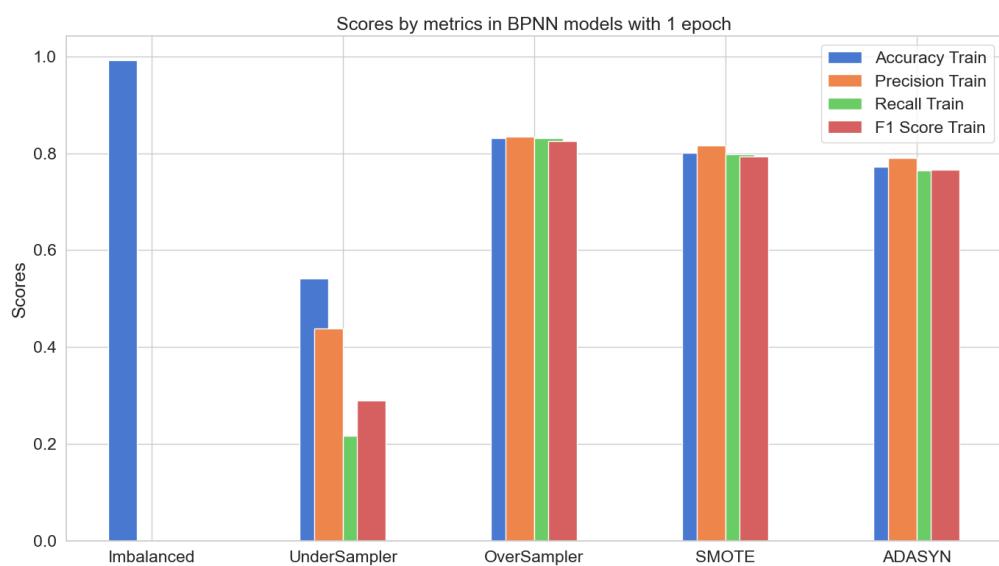


Figura 3.50: Puntuaciones del entrenamiento del modelo BPNN por estrategias en 1 epoch.

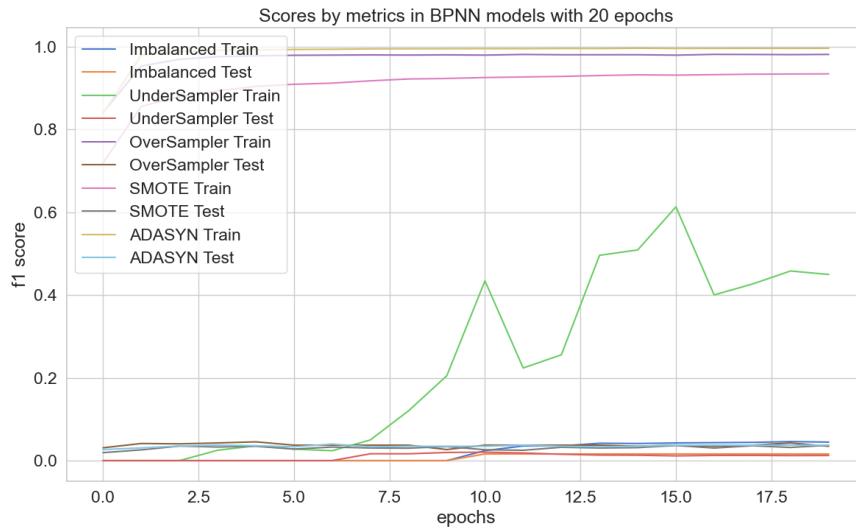


Figura 3.51: Puntuaciones del entrenamiento del modelo BPNN por estrategias en 20 epochs.

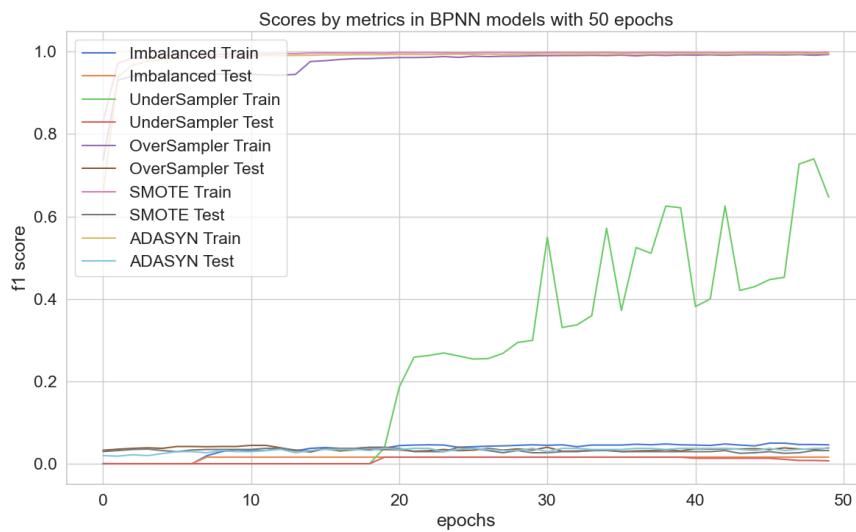


Figura 3.52: Puntuaciones del entrenamiento del modelo BPNN por estrategias en 50 epochs.

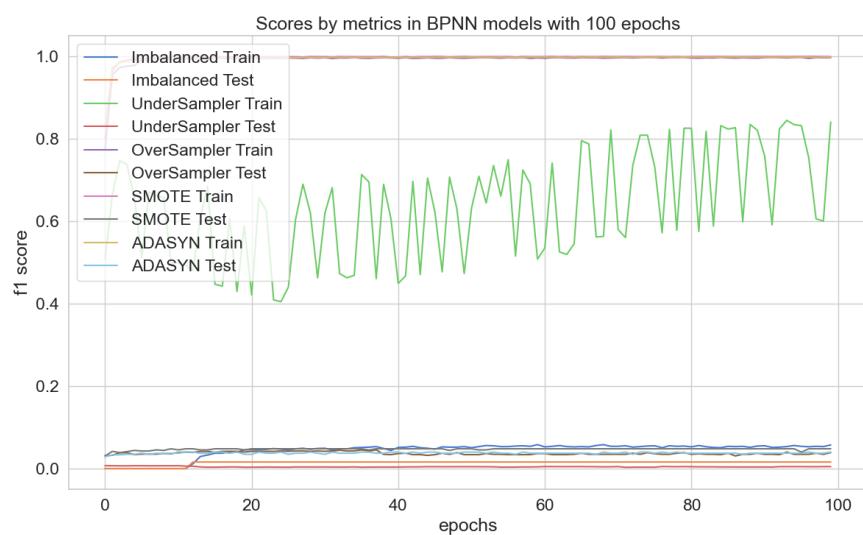


Figura 3.53: Puntuaciones del entrenamiento del modelo BPNN por estrategias en 100 epochs.