Tipo de artículo: Artículo original

Problemas desbalanceados en el aprendizaje profundo

Imbalanced problems on Deep Learning

Odeynis Valdés Suárez[*] **https://orcid.org/0000-0002-5847-1257**
Héctor Raúl González Diez[2], **https://orcid.org/0000-0002-7601-4201**

[1] Facultad 2, Universidad de las Ciencias Informáticas (UCI). ovaldes@estudiantes.uci.cu

[2] Departamento de Ciencia y Técnica, Facultad CITEC, Universidad de las Ciencias Informáticas (UCI). hglez@uci.cu

*** Autor para correspondencia:** ovaldes@estudiantes.uci.cu

Resumen

El contenido de este trabajo está centrado en la obtención de resultados, que demuestren la eficiencia de los modelos de aprendizaje profundo con estrategias para la solución al desbalance de la información en la detección de anomalías, en específico, en la detección de fraude de tarjeta de crédito. Para ello se realiza una fundamentación previa de los conceptos básicos relacionados con este campo de estudio, ya sean los avances y retos que presenta, así como las métricas, herramientas y tecnologías que se usan para su estudio, metodología y otras técnicas que se usan. Se definen las soluciones al problema del desbalance de la información a utilizar, se eligen los modelos de *Deep Learning* para realizar la detección de fraude de tarjeta de crédito, además de definir su estructura. Con estos modelos definidos se realizan las evaluaciones y comparaciones correspondientes para comprobar su efectividad mediante las métricas definidas, lo que va a permitir sentar las bases para la obtención de resultados concluyentes con respecto a la efectividad de los modelos, los cuales son el resultado final de este artículo.
Palabras clave: modelos de aprendizaje profundo, detección de fraude de tarjeta de crédito, detección de anomalías, problema del desbalance de la información.

Abstract

The content of this work is focused on obtaining results, which demonstrate the efficiency of deep learning models with strategies for the imbalanced solution in the detection of anomalies, especially, in the detection of credit card fraud. For this, a preliminary foundation of the basic concepts related to this field of study is carried out, whether they are the advances and challenges it presents, as well as the metrics, tools and technologies that are used for its study, methodology and other techniques that are used. The solutions to the problem of the imbalance of the information to be used are defined, the Deep Learning models are chosen to carry out the detection of the credit card fraud, in addition to defining its structure. With these defined models, the corresponding evaluations and comparisons are carried out to verify their effectiveness through the defined metrics, which will allow to lay the foundations for obtaining conclusive results regarding the effectiveness of the models, which are the final result of this article.
Keywords: *Deep learning models, credit card fraud detection, anomaly detection, information imbalanced problem.*

Recibido: 22/12/2021
Aceptado: 18/03/2022
En línea: 01/06/2022

Introducción

Las Tecnologías de la Información y las Comunicaciones (TICs) se han convertido en un eslabón fundamental en el desarrollo de la sociedad como es 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 es posible con solo un dispositivo móvil conectado a *internet*. El fraude de tarjetas crédito y débito encabeza la lista de fraudes bancarios por las diversas formas 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)(Doerr, Gambacorta, y Serena 2021) 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 le 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.

La clasificación desbalanceada tiene dos principales causas que son las muestras de datos y las propiedades del dominio, en el caso de este artículo 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. Con el surgimiento del *Big Data*, comienza el incremento del conjunto de datos e información que se está creando y recolectando constantemente. 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 *Artificial Neural Network* (ANN) para llevar procesos de ML. Comparada con técnicas de ML como *Support Vector Machine* (SVM) y *K-Nearest Neighbors* (KNN), DL posee ventajas de los aprendizajes no supervisados, una fuerte capacidad de generalización, y un poderoso entrenamiento robusto para Big Data. El DL posee grandes ventajas con respecto a ML, una de ellas es la redundancia de la extracción de características.

Los modelos DL permiten una 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. Estos modelos junto a estrategias de solución al desbalance de la información permiten una mayor precisión en la detección de los fraudes bancarios, en específico, en las transferencias de tarjetas de débito y crédito(Delamaire, Abdou, y Pointon 2009).

Este artículo va orientado en la demostración mediante resultados de la alta efectividad que se obtiene en la detección de fraudes con los modelos DL aplicando estrategias para la solución al desbalance de la información.

Materiales y métodos

En esta sección se presentarán las estrategias para solucionar el desbalance de la información y los modelos DL utilizados en la experimentación, así como sus resultados y una comparativa teniendo en cuenta las métricas y parámetros aplicados. Para ello es necesario una fundamentación de algunos conceptos fundamentales para lograr un entendimiento del proceso que se va a llevar a cabo.

Detección de anomalías

La detección de anomalías no es más que la identificación de valores anómalos que distan de un valor normal, llevado a la detección de fraudes(Thudumu et al. 2020), no es más que la identificación de una transferencia anómala que no sigue un comportamiento normal a las demás transferencias. Este comportamiento fraudulento se vuelve más detectable cuando se posee un gran conjunto de transferencias, aquí es donde entra el *Big Data* como la mayor fuente de datos.

Estrategias para el desbalance de la información

La clasificación desbalanceada tiene dos principales causas que son las muestras de datos y las propiedades del dominio, 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.

Una de las técnicas para la solución de problemas desbalanceados es el sub muestreo (*Undersample* 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.

Otra forma de resolver este problema es el sobre muestreo (*Oversample* en inglés) 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. El enfoque más utilizado para sintetizar nuevos ejemplos es *Synthetic Minority Oversampling TEchnique* (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 (Brownlee 2021). 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.

Estas técnicas o estrategias serán aplicadas al conjunto de entrenamiento de forma tal que se trabajen con 4 conjuntos de entrenamientos con diferentes estrategias aplicadas, además se mantendrá el conjunto de entrenamiento desbalanceado. A continuación, se puede apreciar las diferencias que existen en los conjuntos con estrategias aplicadas con respecto al original:

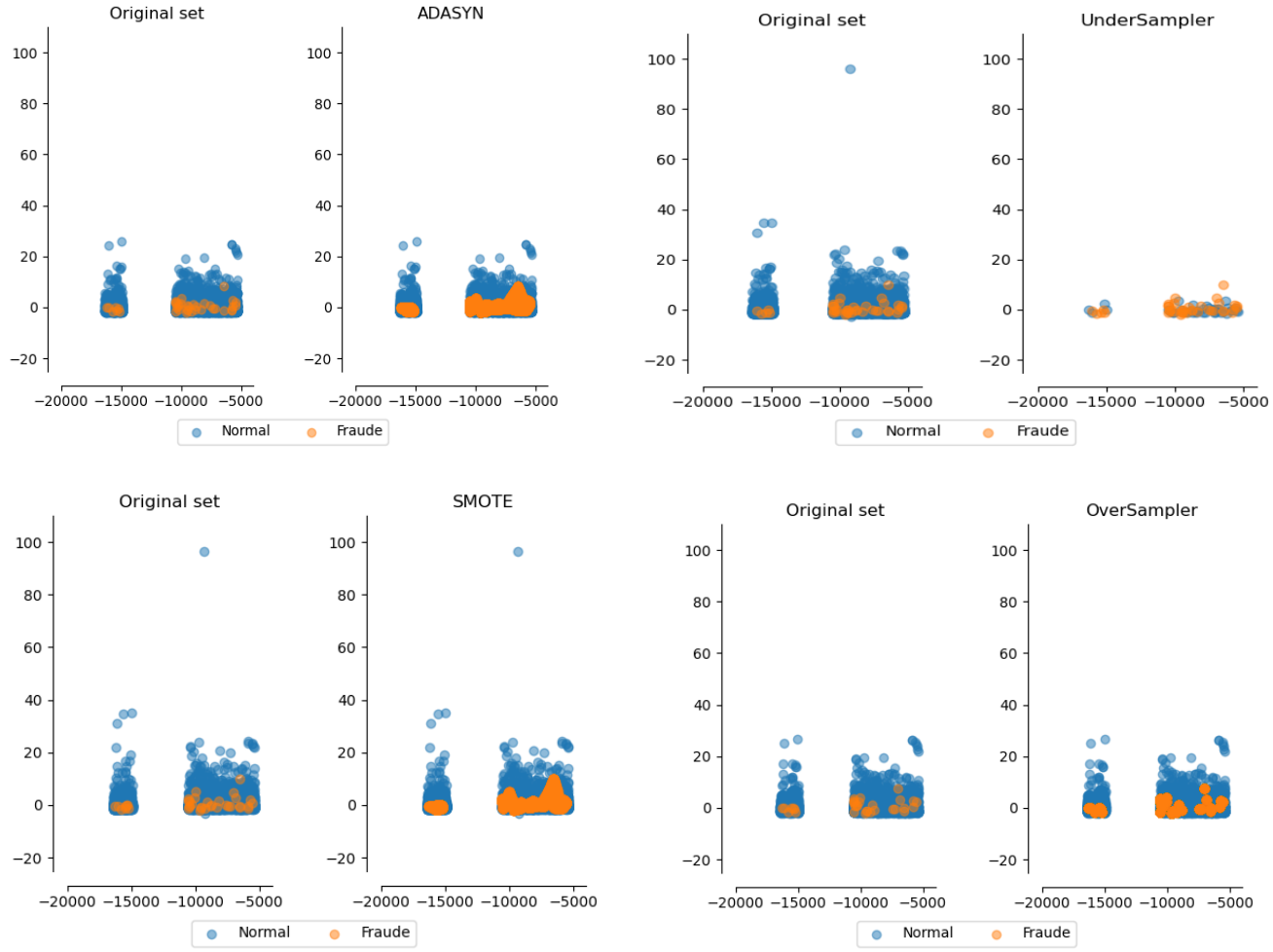

Figura 1. Valores del conjunto de entrenamiento con estrategias aplicadas.

Modelos DL y sus parámetros

Los modelos DL, no son más que redes neuronales artificiales, que permiten la clasificación de información entrante y la obtención de patrones o tendencias. Estos modelos 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 (Nguyen et al., s. f.). 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, estos 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.(Bulusu et al. 2020)

Para la detección de anomalías se han creado varios modelos DL representativos, de los cuales se emplearán los siguientes:

- ***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(Zhang et al. 2018).
- ***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 (Faculty of Sciences IPSS, University Mohammed V, Rabat, Morocco et al. 2021). En otras palabras, RNN es una red neuronal con memoria que tiende a tener un corto término de memoria por el problema de la 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 (Nengcheng et al. 2019). 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.

Antes de definir la arquitectura de los modelos a utilizar es necesario definir los parámetros que se utilizarán para ajustar los modelos, paso muy importante ya que un modelo DL mal ajustado puede lanzar valores falsos.

Funciones de activación

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 por su peso asociado y se suman junto con la parcialidad, luego este valor pasa por una función conocida como función de activación, transformando el valor neto de entrada en un valor de salida. Estas 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.
- *Sigmoide*: transforma los valores en el rango $(-\infty, +\infty)$ a valores en el rango $(0, 1)$.

Función de coste

La función de coste 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*.

Disminución del error

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 qué pesos de la red hay que modificar para mejorarla.
- 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. Es uno de los optimizadores con mayor tendencia en la actualidad.

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. Para este experimento se usará 100 épocas.

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 máximo 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. Teniendo en cuenta lo anterior 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.

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. *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 experimento se les aplicarán a los algoritmos tres capas de *dropout* con capas intermedias, cada uno con un porcentaje de neuronas desactivadas diferentes.

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.

Para el experimento 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*. Finalmente, una capa de salida; además se aplicará Adam como optimizador. La arquitectura se muestra a continuación:

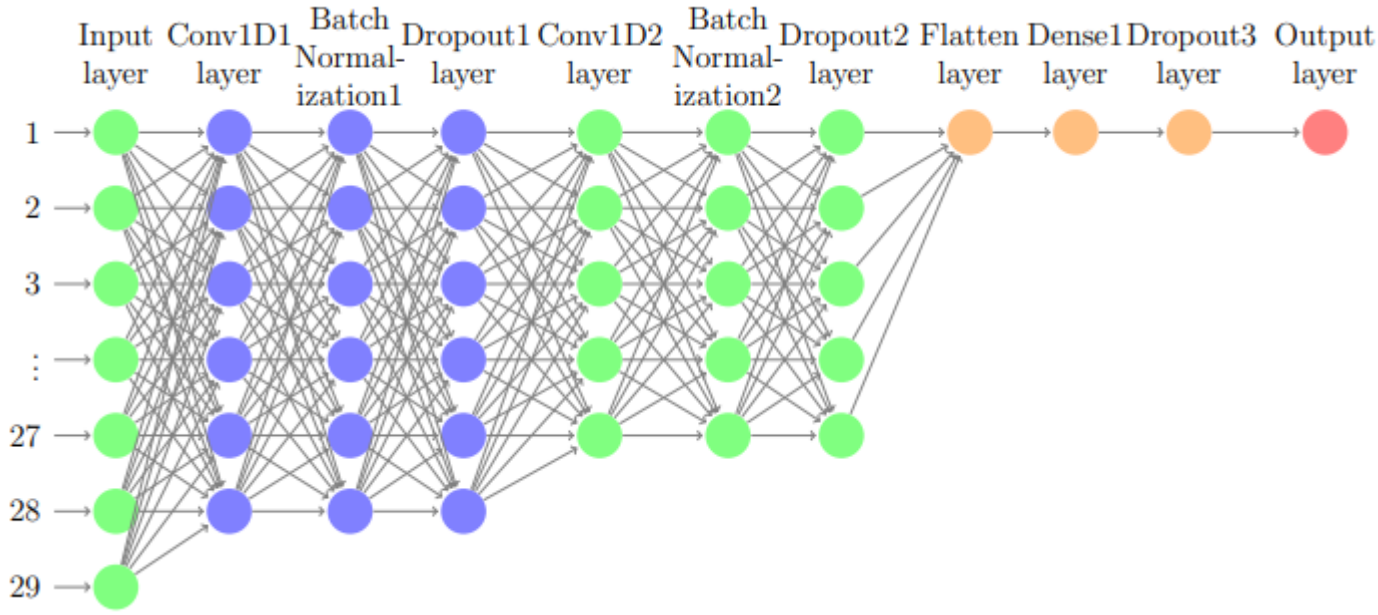

Figura 2. Modelo CNN

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.

El modelo 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, se muestra su arquitectura:

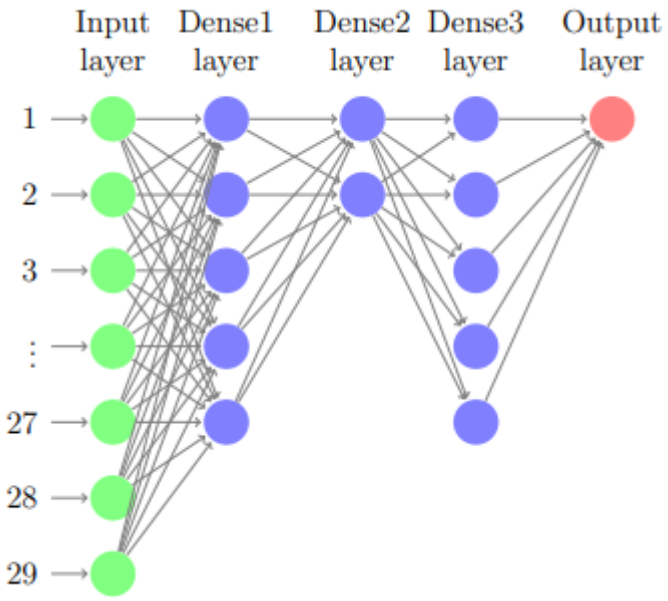

Figura 3. Modelo BPNN

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 en que se realizan las transacciones. En el procesamiento de los datos desde la entrada a la salida intervienen los vectores de peso dependiendo de las capas.

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. Su arquitectura es la siguiente:

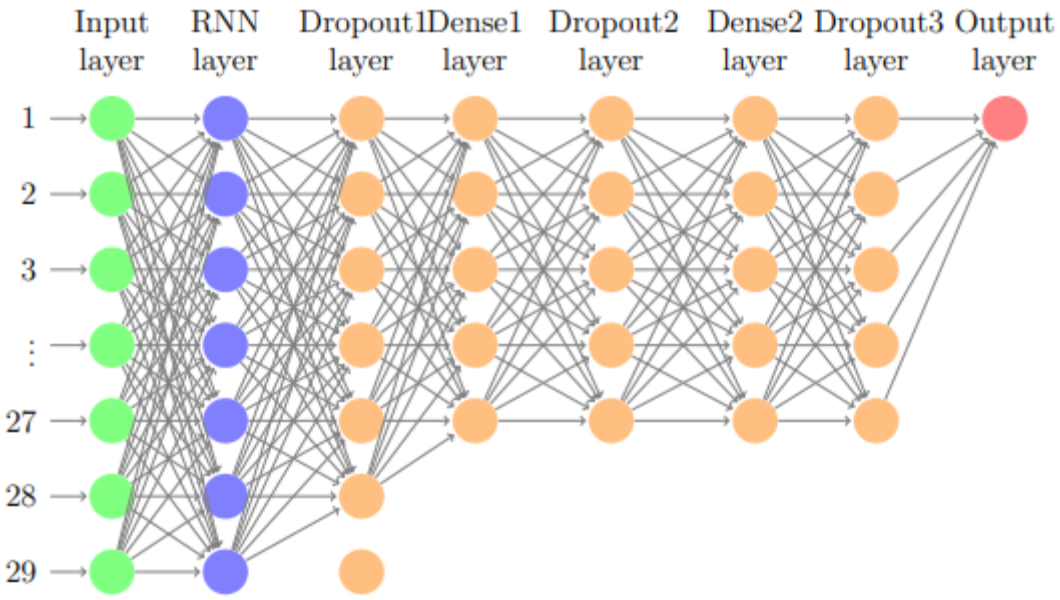

Figura 4. Modelo RNN

Métricas

Existen métricas para demostrar la efectividad de los modelos DL por la cantidad de aciertos, permitiendo definir cuando un modelo es útil en la detección de fraudes, incluso en la comparación entre modelos. Las métricas a comparar en este artículo son:

- ***Accuarcy***: 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 harmonía 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.

Haciendo uso de estas métricas se puede lograr una comparación entre los modelos DL a utilizar, teniendo en cuenta además, las estrategias de solución y demás parámetros de los modelos.

Tecnologías y Herramientas

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 para la utilización de modelos 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: Python, C++4, Java y lenguajes de JVM, JavaScript, Swift y lenguaje R. 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.

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.
- 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.
- 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.
- Sickit-learn: es una herramienta de código abierto, simple y eficiente para realizar análisis de datos predictivos.
- Matplotlib: es una librería comprensiva para la agrupación estática, animada y visualizaciones interactivas en Python.
- Imbalanced-learn: proporciona herramientas para tratar el problema de información desbalanceada. En esta librería se encuentran los algoritmos de *UnderSample*, *OverSample*, SMOTE y ADASYN.

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

Con la definición de las herramientas y teconologías a aplicar se procede la presentación de los resultados de los experimentos realizados.

Metodología de la minería de datos

Para este proyecto se aplicará la metodología KDD que es un análisis exploratorio, automático y modelado de grandes repositorios de datos(Maimon y Rokach 2010). Es un proceso organizado de identificación válida, novel, útil, y entendibles patrones de un gran y complejo conjunto de datos. Los pasos de esta metodología para orientar el desarrollo se definen como:

1. Entendimiento, conocimientos previos e identificación de la meta.
2. Selección de un conjunto de datos.
3. Limpieza y preprocesamiento 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.

Teniendo en cuenta el enfoque de este trabajo se decide utilizar la clasificación como método de minería de datos, el cual usa algoritmos de aprendizaje supervisados para modelar o predecir variables aleatorias discretas.

Basados en todo lo antes expuesto, se procede a la presentación de los resultados obtenidos en la experimentación.

Resultados y discusión

Los resultados obtenidos se muestran a continuación:

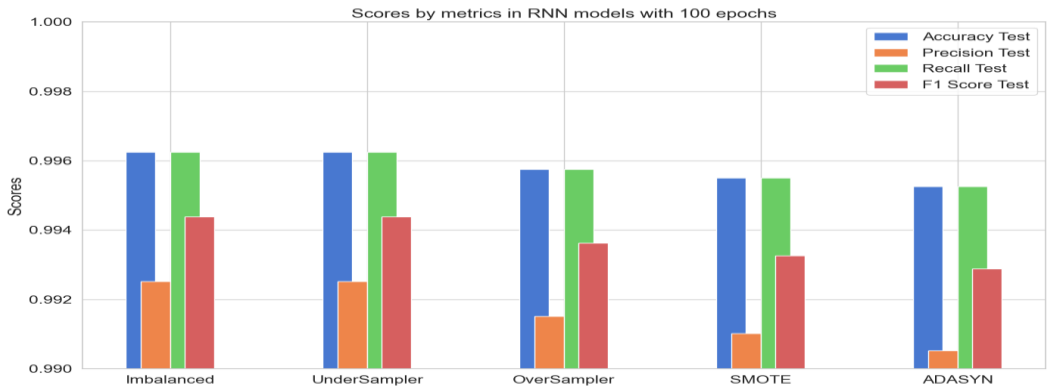

Figura 5. Puntuaciones por métricas en modelos RNN con 100 épocas.

En la figura anterior se aprecia que en el modelo RNN se obtienen los mejores resultados con la estrategia *UnderSampler* y con los datos desbalanceados, mientras que con la estrategia ADASYN se obtienen los peores resultados.

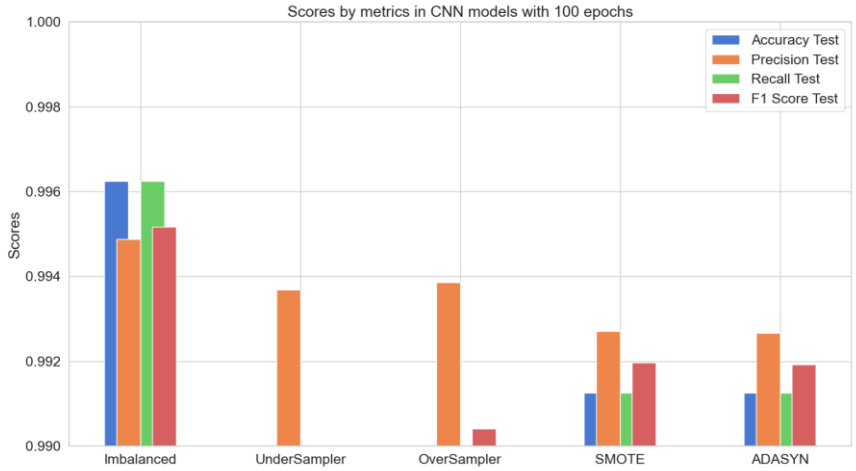

Figura 6. Puntuaciones por métricas en modelos CNN con 100 epochs.

De la figura 6 se evidencia la superioridad que existe en resultados del modelo CNN con datos desbalanceados con respecto a las estrategias de solución al desbalance de la información. En el caso de *UnderSampler* se obtienen pésimos resultados en este modelo.

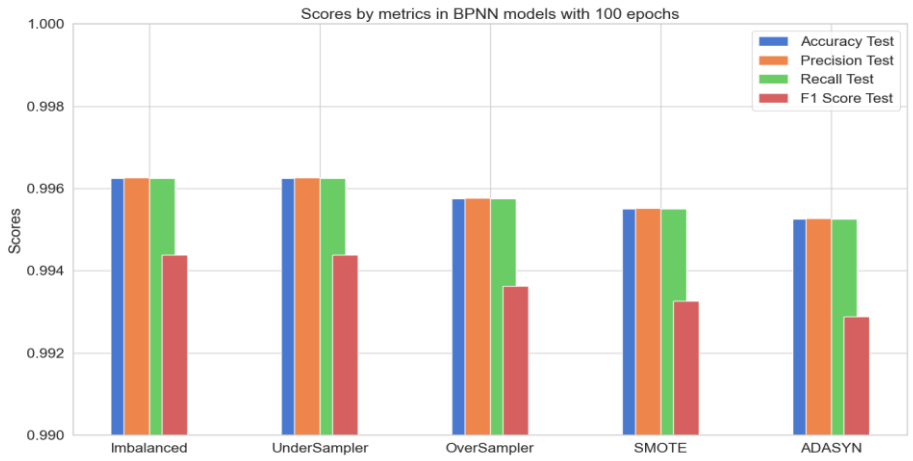

Figura 7. Puntuaciones por métricas en modelos BPNN con 100 epochs.

En la figura anterior se muestran los resultados obtenidos en la experimentación con los modelos BPNN, donde los mejores resultados se obtienen con los datos desbalanceados y con la estrategia *UnderSampler*. El peor resultado se obtiene con la estrategia ADASYN.

Tabla 1. Puntuaciones por métricas de los modelos DL por estrategias**.**

Eposh	Model	Accuracy	AUC	Precision Score	Recall Score	F1 Score
100	CNN IMBALANCE	0.99625	0.56641573	0.99488395	0.99625	0.99516706
100	BPNN IMBALANCE	0.99625	0.5	0.99626406	0.99625	0.99437852
100	BPNN UNDERSAMPLE	0.99625	0.5	0.99626406	0.99625	0.99437852
100	RNN IMBALANCE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	RNN UNDERSAMPLE	0.99625	0.5	0.99251406	0.99625	0.99437852
100	BPNN OVERSAMPLE	0.99575	0.5	0.99576806	0.99575	0.99362953
100	RNN OVERSAMPLE	0.99575	0.5	0.99151806	0.99575	0.99362953
100	BPNN SMOTE	0.9955	0.5	0.99552025	0.9955	0.99325507
100	RNN SMOTE	0.9955	0.5	0.99102025	0.9955	0.99325507
100	BPNN ADASYN	0.99525	0.5	0.99527256	0.99525	0.99288065
100	RNN ADASYN	0.99525	0.5	0.99052256	0.99525	0.99288065
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
100	CNN OVERSAMPLE	0.9875	0.70086101	0.99386401	0.9875	0.99040606
100	CNN UNDERSAMPLE	0.76675	0.61727311	0.99368038	0.76675	0.86451737

Observando los resultados de la tabla anterior se pueden extraer varias conclusiones, las cuales son:

- El modelo CNN con los datos desbalanceados obtiene el mejor resultado basado en las puntuaciones de las métricas.
- El segundo mejor modelo es BPNN con los datos desbalanceados y con la estrategia *UnderSampler*, empatados con las puntuaciones iguales en las métricas.
- El tercer mejor modelo es RNN con los datos desbalanceados basado en las puntuaciones de las métricas.
- El peor modelo es CNN con la estrategia *UnderSampler* con las peores puntuaciones de las métricas.
- Los modelos obtienen sus mejores resultados con los datos desbalanceados, lo que permite afirmar que los modelo DL aprenden del desbalance de la información, siendo innecesario aplicar una estrategia para mejorar los resultados.
- Los modelos RNN y BPNN mantienen un comportamiento similar en las puntuaciones, a diferencia de la métrica *precision*, donde el modelo BPNN es superior.
- Los modelos BPNN y RNN obtienen mejores resultados en las estrategias con respecto al modelo CNN.
- Para los modelos BPNN y RNN se obtienen sus mejores resultados con los datos desbalanceados y la estrategia *UnderSampler*, y sus peores resultados con ADASYN.
- Para el modelo CNN se obtiene su mejor resultado con datos desbalanceados y su peor resultado con *UnderSampler*.

Teniendo en cuenta los resultados obtenidos mediante la tabla y las figuras anteriores, se demuestra la efectividad de los modelos DL con respecto a la detección de fraude bancario sin necesidad de solución al desbalance de la información, ya que estos modelos aprenden de este problema.

Conclusiones

Los resultados de este trabajo han sido satisfactorios y cumplen con los objetivos propuestos. Estos resultados 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. Además, son la mejor opción cuando se trata de trabajar con Big Data, ya que obtienen sus mejores resultados cuando trabajan con datos desbalanceados. 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 datos desbalanceados con respecto a las estrategias aplicadas.

Referencias

Brownlee, Jason. 2021. «SMOTE for Imbalanced Classification with Python», 17 de enero de 2021. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/.

Bulusu, S, B Kailkhura, B Li, P Varshney, y D Song. 2020. «Anomalous Instance Detection in Deep Learning: A Survey».

Delamaire, Linda, Hussein Abdou, y John Pointon. 2009. «Credit Card Fraud and Detection Techniques: A Review». *Banks and Bank Systems* 4 (2): 13.

Doerr, Sebastian, Leonardo Gambacorta, y Jose Maria Serena. 2021. «Big data and machine learning in central banking». BIS Working Papers.

Faculty of Sciences IPSS, University Mohammed V, Rabat, Morocco, Ibtissam Benchaji, Samira Douzi, y Bouabid El Ouahidi. 2021. «Credit Card Fraud Detection Model Based on LSTM Recurrent Neural Networks». *Journal of Advances in Information Technology* 12 (2): 113-18. https://doi.org/10.12720/jait.12.2.113-118.

Maimon, Oded, y Lior Rokach, eds. 2010. *Data Mining and Knowledge Discovery Handbook*. Boston, MA: Springer US. https://doi.org/10.1007/978-0-387-09823-4.

Nguyen, Thanh Thi, Hammad Tahir, Mohamed Abdelrazek, y Ali Babar. s. f. «Deep Learning Methods for Credit Card Fraud Detection», 8.

Thudumu, Srikanth, Philip Branch, Jiong Jin, y Jugdutt (Jack) Singh. 2020. «A Comprehensive Survey of Anomaly Detection Techniques for High Dimensional Big Data». *Journal of Big Data* 7 (1): 42. https://doi.org/10.1186/s40537-020-00320-x.

Zhang, Zhaohui, Xinxin Zhou, Xiaobo Zhang, Lizhi Wang, y Pengwei Wang. 2018. «A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection». Editado por Zhaoqing Pan. *Security and Communication Networks* 2018 (agosto): 5680264. https://doi.org/10.1155/2018/5680264.