Taller de redes neuronales artificiales

Tovar, Daniel Federico. *Estudiante Maestría en Inteligencia Artificial*. Gómez Parra, Héctor., *Estudiante Maestría en Inteligencia Artificial*

Abstract— In this paper, we address the optimization of a neural network to perform accurate classification on a dataset. In Section 1, the hyperparameter tuning of the network is explored and its performance is evaluated using classification metrics. Model selection decisions are justified by analyzing learning curves and the decision boundaries found are visualized. In Point 2, the task of classification of mystery data provided in two sets: public_dataset and quiz is described. The creation of a neural network model to predict the variable 'label' is detailed.

Resumen— En este artículo, se aborda la optimización de una red neuronal para llevar a cabo una clasificación precisa en un conjunto de datos. En el Punto 1, se explora el ajuste de hiperparámetros de la red y se evalúa su rendimiento utilizando métricas de clasificación. Se justifican las decisiones de selección del modelo mediante el análisis de las curvas de aprendizaje y se visualizan las fronteras de decisión encontradas. En el Punto 2, se describe la tarea de clasificación de datos misteriosos proporcionados en dos conjuntos: public_dataset y quiz. Se detalla la creación de un modelo de redes neuronales para predecir la variable 'label'.

***Index Terms*— Artificial Neural Networks (ANN), Activation Function, Learning Curves, Neural Network Training**

I. Introduction

En el campo del aprendizaje automático, la tarea fundamental de clasificar datos se ha convertido en un área de gran relevancia. En este contexto, las redes neuronales artificiales (RNAs) han surgido como herramientas de gran potencial para abordar este desafío. No obstante, alcanzar un rendimiento óptimo en la clasificación mediante RNAs es un proceso que con frecuencia demanda una minuciosa afinación de los hiperparámetros del modelo. Además, es de suma importancia adentrarse en la naturaleza subyacente de los datos para obtener resultados efectivos.

En este estudio, enfrentamos el desafío de mejorar el rendimiento de una red neuronal en un conjunto de datos donde su desempeño inicial se muestra insatisfactorio. Nuestra estrategia se centra en la exploración de los hiperparámetros de la RNA con el propósito de lograr una clasificación precisa. Además, medimos el desempeño del modelo mediante el empleo de métricas de clasificación estándar, tales como precisión, sensibilidad y especificidad. Para respaldar la elección del modelo final, recurrimos al análisis de las curvas de aprendizaje, lo cual nos permite comprender la evolución de la RNA durante el proceso de entrenamiento y proporciona una validación de nuestra selección.

Así mismo, en la segunda parte se dispone de dos conjuntos de datos: "public_dataset" y "quiz." El objetivo primordial en esta parte es la construcción de un modelo eficaz de redes neuronales capaz de realizar predicciones acertadas para la variable objetivo, identificada como "label" en el conjunto de datos.

En las siguientes secciones, expondremos detalladamente nuestra metodología para optimizar los hiperparámetros de la red neuronal, evaluar su rendimiento en la clasificación y respaldar la elección del modelo final mediante el análisis de curvas de aprendizaje y la visualización de las fronteras de decisión. A través de estos pasos, buscamos arrojar luz sobre las técnicas y consideraciones que pueden conducir a mejoras significativas en la clasificación en escenarios del mundo real.

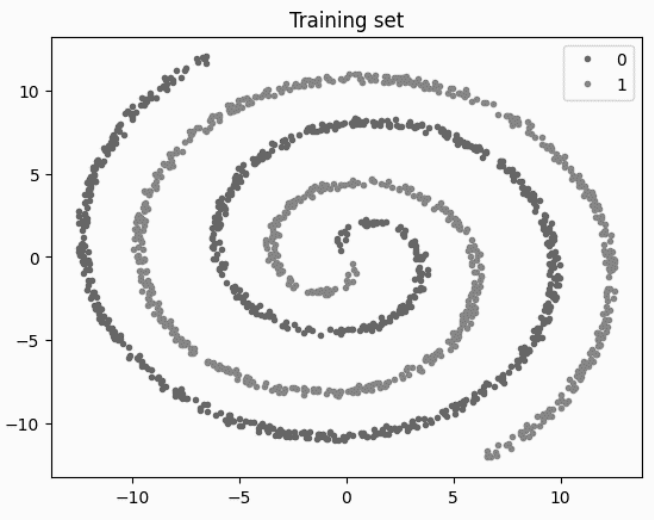
II. MEJORAMIENTO DE RENDIMIENTO DE RED NEURONAL

Para este ejercicio, se creó una función de dos espirales; para esto se definió una función denominada "twospirals" que se encarga de generar un conjunto de datos con una forma de dos espirales. Esta función toma dos parámetros: "n_points" (el número de puntos de datos) y "noise" (la cantidad de ruido presente en los datos). La función genera dos espirales opuestas

en sentido, y el nivel de ruido se controla mediante el parámetro "noise". Los resultados se devuelven en dos matrices: una para las coordenadas de entrada (X e Y) y otra para las etiquetas de clase (0 o 1).

Una vez que se ha generado el conjunto de datos, se procede a realizar una división esencial en conjuntos de entrenamiento y prueba mediante el uso de la función train_test_split de la biblioteca scikit-learn. En esta división, se asigna un 80% de los datos al conjunto de entrenamiento, representado por las variables X_train e Y_train, mientras que el restante 20% se destina al conjunto de prueba, identificado como X_test e Y_test.

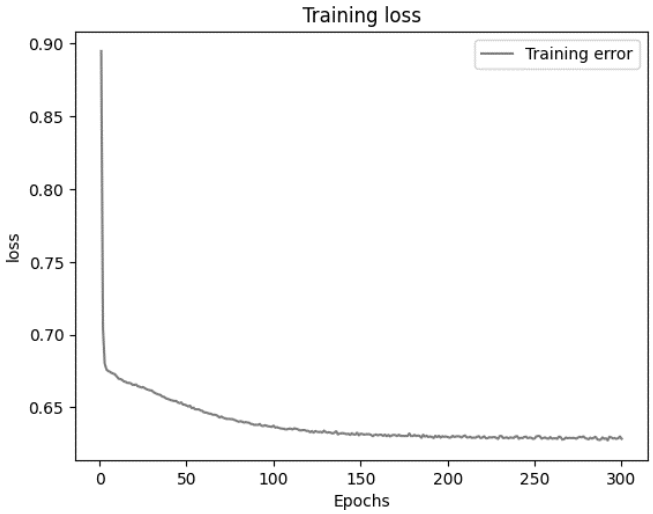
Posteriormente, con el objetivo de proporcionar una visualización clara y representativa del conjunto de entrenamiento, se emplea un gráfico especialmente diseñado para mostrar la disposición de los puntos de datos en el espacio bidimensional. En esta representación gráfica, los puntos pertenecientes a distintas clases se resaltan con colores diferentes. En concreto, aquellos puntos etiquetados como clase 0 se visualizan en un color específico (Azul), mientras que los puntos etiquetados como clase 1 se representan en otro color distinto (Naranja). Esta técnica gráfica resulta fundamental para comprender la estructura y distribución de los datos en el espacio bidimensional, lo que permite apreciar de manera visual cómo las dos espirales se entrelazan y se relacionan en el conjunto de entrenamiento (Ver gráfica)



A partir de esto, se procede a la creación de un modelo secuencial de red neuronal utilizando la librería TensorFlow/Keras. El modelo se compone de una capa de entrada con 8 neuronas, las cuales están activadas mediante la función ReLU, seguida de una capa de salida con una sola neurona y activación sigmoide. Para la compilación del modelo, se emplea la función de pérdida de binary_crossentropy, y se configura el optimizador Adam. Además, se mide el rendimiento utilizando la métrica de precisión (accuracy).

Siguiente a esto, el modelo se somete a un proceso de entrenamiento utilizando el conjunto de entrenamiento, realizando 300 iteraciones (épocas) y utilizando lotes de datos de tamaño 16 (batch size). Se registra un historial del entrenamiento que captura la evolución de la función de pérdida en cada época.

A continuación, se presenta el gráfico que ilustra cómo varía la función de pérdida en relación con el número de épocas tanto en el conjunto de entrenamiento como en el conjunto de validación. Esta representación gráfica permite evaluar el desempeño del modelo y detectar posibles indicativos de sobreajuste o subajuste.



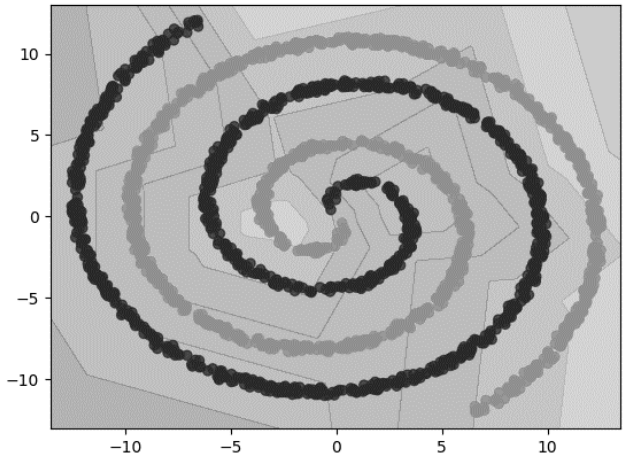
Luego de esto, se evalua las predicciones de un modelo de clasificación en relación con las etiquetas reales en un conjunto de prueba. El proceso se divide en varios pasos esenciales. Primero, se generan las predicciones del modelo y se almacenan en una matriz 'predictions'. Luego, estas predicciones se combinan con las etiquetas reales en un DataFrame llamado 'df_y', que consta de dos columnas. Una tercera columna, 'predict_bin', se agrega al DataFrame, donde las predicciones se convierten en valores binarios basados en un umbral de 0.5. Posteriormente, se muestra el DataFrame con las predicciones binarias (*Ver tabla*). Se utiliza la función 'pd.value_counts' para contar las etiquetas verdaderas en el conjunto de prueba, proporcionando una visión cuantitativa de las observaciones en cada clase. Finalmente, se calcula una matriz de confusión de 2x2 para evaluar el desempeño del modelo, permitiendo la comparación de las etiquetas verdaderas con las predicciones binarias.

	predictions	true	predict_bin
0	0.630950	0.0	1
1	0.353252	1.0	0
2	0.517967	0.0	1
3	0.702897	1.0	1
4	0.248921	0.0	0
...	...	...	...
195	0.715349	1.0	1
196	0.564741	0.0	1
197	0.650493	0.0	1
198	0.310748	0.0	0
199	0.261369	1.0	0

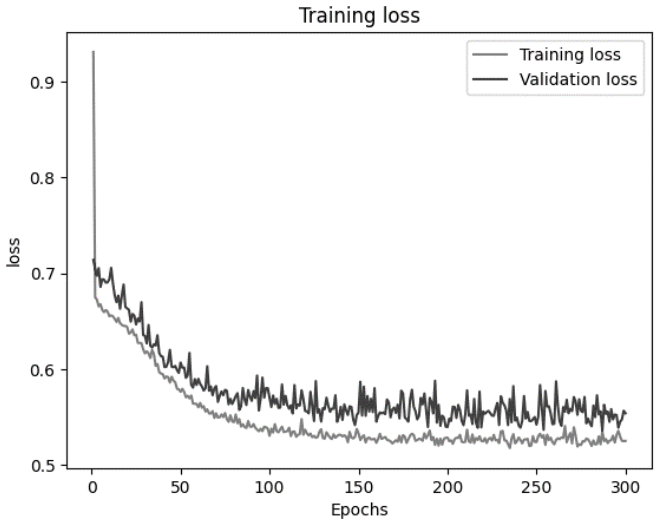
200 rows × 3 columns

Siguiente a esto, se gráfica las fronteras de decisión del modelo de clasificación. Para esto, se crea una malla de puntos bidimensional con un paso específico (h) y se obtienen las etiquetas de clasificación para cada punto en esa malla utilizando el modelo. Luego, se generan predicciones de clasificación para la malla completa y se visualizan como áreas coloreadas (fronteras de decisión) en el espacio bidimensional. Además, se superponen los puntos originales del conjunto de datos, coloreados según sus etiquetas reales. Esto permite observar cómo el modelo separa las diferentes clases en el espacio de características (*Ver gráfica*).

```
51199/51199 [==============================] - 76s 1ms/step
2219/2219 [==============================] - 4s 2ms/step
<matplotlib.collections.PathCollection at 0x7efff3bd8b20>
```



Dado que la anterior gráfica no nos dice mucho acerca del performance del modelo en los datos de test, se construyó nuevamente el modelo de red neuronal con dos capas ocultas, cada una compuesta por 4 neuronas y utilizando la función de activación ReLU. Se emplearon callbacks, EarlyStopping para prevenir el sobreajuste y ModelCheckpoint para guardar el modelo con el menor error de validación. El modelo se compila con pérdida de binary_crossentropy y métricas de precisión, luego se entrenó con datos de entrenamiento, supervisado por los callbacks. El historial de pérdida se registró y visualizó a lo largo de las épocas, lo que permitió evaluar el aprendizaje del modelo y su mejora con el tiempo. Así mismo, se graficó la función de pérdida junto con la de validación (*Ver gráfica*).



Como con el modelo anterior, también se evaluó las predicciones del modelo de clasificación en relación con las etiquetas reales en un conjunto de prueba. Se muestra el DataFrame con las predicciones binarias (*Ver tabla*). Se utilizó la función 'pd.value_counts' para contar las etiquetas verdaderas en el conjunto de prueba, proporcionando una visión cuantitativa de las observaciones en cada clase. Finalmente, se calculó una matriz de confusión de 2x2 para evaluar el desempeño del modelo, permitiendo la comparación de las etiquetas verdaderas con las predicciones binarias.

	predictions	true	predict_bin
0	0.732837	0.0	1
1	0.590433	1.0	1
2	0.635873	0.0	1
3	0.656477	1.0	1
4	0.000033	0.0	0
...	...	...	...
195	0.659125	1.0	1
196	0.541492	0.0	1
197	0.735179	0.0	1
198	0.319734	0.0	0
199	0.488259	1.0	0

200 rows × 3 columns

Dado que la red neuronal no presenta un buen comportamiento en el conjunto de datos. Se van a explorar los hiperparámetros de la red para realizar una clasificación adecuada, se medirá el performance utilizando métricas de clasificación y se graficará las fronteras de decisión.

A. Experimentation

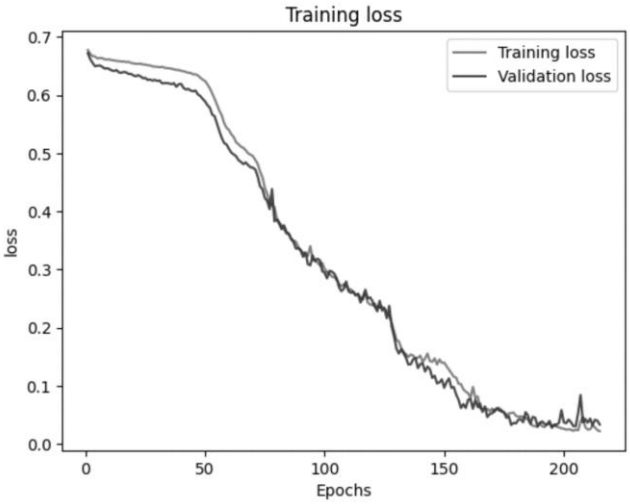
La fase experimental de este estudio se centró en la exploración de la configuración de capas y neuronas más apropiada para la tarea de clasificación, con el objetivo de reducir eficazmente el error asociado. Un total de 11 iteraciones meticulosas se llevaron a cabo para descubrir la combinación ideal de capas y neuronas que maximizara el rendimiento del modelo. Inicialmente, se introdujo una capa oculta adicional con un modesto conjunto de cuatro neuronas, y se procedió a evaluar su impacto en el comportamiento del modelo. Sin embargo, a pesar de estos esfuerzos, no se evidenciaron mejoras sustanciales en la eficiencia del clasificador.

Como respuesta a esta falta de mejoras notables, se decidió explorar ajustes más significativos en la arquitectura de la red. Esto incluyó la introducción de una segunda capa oculta con un incremento de cuatro neuronas adicionales. A pesar de estos ajustes, la mejora deseada no se materializó completamente. Como resultado, se expandió el enfoque y se agregaron cinco capas ocultas más, cada una de las cuales fue sometida a un rango de neuronas en constante variación, oscilando entre 4 y 8 neuronas.

Paralelamente, se llevó a cabo una serie de ajustes en los hiperparámetros del modelo, con especial atención a la tasa de aprendizaje, que se redujo desde 0.01 a 0.001. Además, se perfeccionó el mecanismo de earlystopping mediante el incremento del número de épocas permitidas, con la intención de otorgar al modelo una mayor flexibilidad en la mejora de su rendimiento, en caso de ser necesario. Estos esfuerzos combinados, respaldados por un riguroso proceso experimental, se realizaron con el firme propósito de encontrar la configuración que maximizara el potencial del modelo de clasificación.

B. Results

I. Se observa que la función de perdida mejora significativamente; esto quiere decir que el error de predicción disminuye a medida que avanza en las iteraciones de épocas. La similitud entre las curvas de pérdida de entrenamiento y validación sugiere que el modelo está bien equilibrado y generaliza eficazmente, evitando el sobreajuste. La disminución constante de la pérdida a lo largo de las épocas indica que el modelo se está acercando a su rendimiento óptimo, reflejando una elección adecuada de hiperparámetros y mostrando margen para mejoras adicionales, sin llegar a una pérdida nula.



II. Así mismo, también se evaluó las predicciones del modelo de clasificación en relación con las etiquetas reales en un conjunto de prueba. Se muestra el DataFrame con las predicciones binarias (*Ver tabla*). Se utilizó la función 'pd.value_counts' para contar las etiquetas verdaderas en el conjunto de prueba, proporcionando una visión cuantitativa de las observaciones en cada clase.

	predictions	true	predict_bin
0	0.520155	0.0	1
1	0.999292	1.0	1
2	0.000006	0.0	0
3	0.993301	1.0	1
4	0.000343	0.0	0
...	...	...	...
195	0.999745	1.0	1
196	0.001358	0.0	0
197	0.000028	0.0	0
198	0.025172	0.0	0
199	0.999978	1.0	1

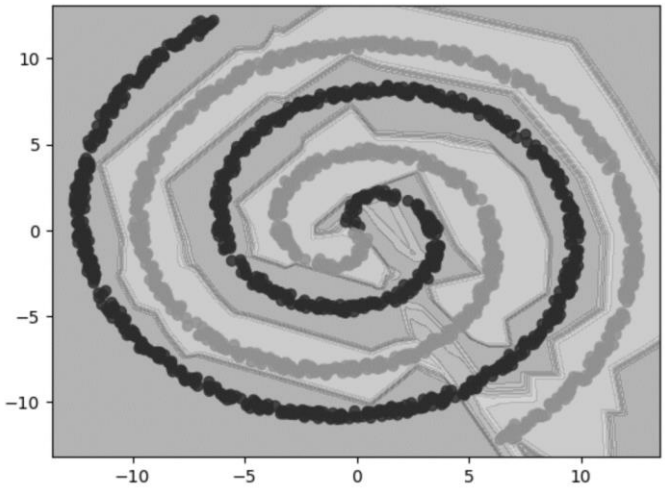
200 rows × 3 columns

III. Ahora bien, analizando la matriz de confusión, en este caso, indica que 91 observaciones se clasificaron correctamente como clase “0”, 107 observaciones se clasificaron correctamente como clase “1”, 1

observación se clasificó incorrectamente como clase "1" cuando en realidad era de clase "0", y 1 observación se clasificó incorrectamente como clase "0" cuando en realidad era de clase 1. Estos resultados sugieren que el modelo presenta un buen rendimiento, ya que la mayoría de las predicciones son precisas, y solo se observa un pequeño número de errores.

```
array([[ 91,   1],
       [  1, 107]])
```

IV. Para reconfirmar lo dicho anteriormente, se graficó las fronteras de decisión. Los resultados obtenidos resaltan la capacidad de aprendizaje del modelo, que logra discernir patrones y relaciones de manera eficiente. Además, la habilidad de generalización es sobresaliente, ya que el modelo es preciso incluso con datos no utilizados durante el entrenamiento, lo que sugiere que no se ha sobreajustado. La elección del algoritmo y la arquitectura del modelo parecen adecuados, puesto que las fronteras de decisión separan las clases de forma clara en el espacio de características y la alta precisión de las predicciones refuerza la fiabilidad del modelo.



V. Finalmente, se hallaron varias métricas de clasificación para confirmar lo dicho anteriormente. El análisis de los resultados del modelo demuestra su buen rendimiento en la tarea de clasificación. La métrica de "precisión" alcanza un sólido 0.989, indicando su capacidad para realizar clasificaciones precisas. La sensibilidad es notable, con un valor de 0.968, lo que destaca la eficacia al identificar la mayoría de los positivos reales. El equilibrio entre precisión y sensibilidad se refleja en un F1-score de 0.978. Además, la precisión general alcanza un 0.98, confirmando la fiabilidad de las predicciones del modelo.

C. Conclusions

- Se observa que el modelo predice y clasifica bien. Tanto la matriz de confusión como la gráfica de fronteras de decisión respaldan un buen rendimiento del modelo de clasificación. La matriz de confusión revela una alta precisión en la mayoría de las predicciones, lo que sugiere que el modelo realiza clasificaciones precisas con un mínimo de errores. La gráfica de fronteras de decisión muestra una buena capacidad de aprendizaje, destacando la idoneidad de la arquitectura del modelo. En conjunto, estos hallazgos respaldan la eficacia del modelo.

- El análisis de los resultados muestra un óptimo rendimiento en la tarea de clasificación, teniendo en cuenta los resultados de las métricas de precisión, sensibilidad y F1-score que refuerzan la confiabilidad del modelo.

III. TAREA DE CALIFICACIÓN MISTERIOSA

A. Methodology

Se llevaron a cabo seis macroprocesos para abordar la tarea de realizar una calificación de una base de datos misteriosa, en primer lugar, se realizó un análisis primario de cada variable en el dataset, su tipo, cantidad de variables, distribución (analisis univariado) y adicionalmente se revisó el balance de la bases de datos en cuanto a etiquetas; posteriormente se probaron distintas herramientas de aprendizaje automático para redes neuronales(incremento de neuronas, incremento de capas, funciones de activación, cambio de tasas de aprendizaje, cambio de optimizadores) hasta lograr una calificación con un accuracy superior al 99% y un f1-score superior al 80%.

B. Experimentation

La experimentación se llevó a cabo siguiendo una metodología sistemática y detallada. En primer lugar, se utilizó un modelo de redes neuronales como base para lograr el rendimiento deseado. Para abordar el desequilibrio en las etiquetas, se implementó una técnica de oversampling, que resultó en un incremento significativo, pasando del 0.0047% al 5% y luego al 20%. Tras la validación exhaustiva del mejor método de oversampling, se procedió a un primer ajuste del modelo mediante el incremento de neuronas y capas. Sin

embargo, este enfoque inicial resultó en un bajo rendimiento en la clasificación de las variables positivas, En consecuencia, se decidió modificar las funciones de activación de la red neuronal.

Se evaluaron diversas opciones, entre las cuales se incluyeron TanH, TanH + Relu, LeakyRelu, PReLu y Swish. Después de determinar la función de activación que mejor se adaptaba al modelo y contribuía al rendimiento deseado, se procedió a ajustar otros parámetros clave. Entre estos ajustes se incluyó la normalización de las variables para mejorar aún más el rendimiento.; además, se realizaron ajustes en los valores de la tasa de aprendizaje mediante el uso del optimizador Adam. Al observar mejoras significativas, se tomó la decisión de cambiar a un optimizador dinámico, considerando opciones como AdaDelta, FTLR y Nadam; para una revisión exhaustiva de cada modelo (curva Loss, matriz de confusión, indicadores de performance) remitirse al notebook.

Por último, a medida que se observaban mejoras progresivas en cada fase, se decidió aumentar nuevamente el número de neuronas y capas en el modelo, este enfoque final permitió alcanzar el rendimiento esperado, consolidando así los resultados obtenidos a lo largo de este proceso exhaustivo de optimización y ajuste.

C. Results

I. Modelo de entrenamiento con una red neuronal y prueba de oversampling, el modelo 0 (original) cuenta con un balance de 0.0047 de clase positiva por cada dato negativo, se realizaron pruebas al 5% y 20% de oversampling manteniendo los demás parámetros fijos (modelo de 3 capas con una distribución de 4,4,1 neuronas y activaciones ReLu para las primeras dos capas y Sigmoid para la última).

	Modelo 0	Over. 5%	Over. 20%
Accuracy	99.53%	99.47%	98.57%
Precision	0.00%	45.21%	23.03%
Recall	0.00%	55.92%	86.12%
F1-Score	0.00%	50.00%	36.35%

Tabla 1. Exp. Oversampling

Se observa que el mejor modelo utiliza un oversampling del 5% y que al aumentarlo disminuye la precisión, por lo que los demás experimentos se realizarán con esta distribución

II. Se utilizaron diferentes arquitecturas (Modelo 1: 8,16,32,1; Modelo 2: 8,16,32,64,1 neuronas, Modelo 3: 4,8,8,4,1, Modelo 4: 4,8,16,1).

	Modelo 0	Modelo 1	Modelo 2	Modelo 3	Modelo 4
Accuracy	99.47%	99.53%	99.30%	99.53%	98.90%
Precision	45.21%	0.00%	36.41%	0.00%	26.03%
Recall	55.92%	0.00%	64.49%	0.00%	72.24%
F1-Score	50.00%	0.00%	46.54%	0.00%	38.27%

Tabla 2. Exp. Neuronas y Capas

Se opta por utilizar el modelo 4 que es más estable durante las corridas y tiene buenos performance, pues el modelo 0 es muy básico.

III. Se cambió la función de activación para el modelo 4 con las siguientes funciones y resultados:

	Modelo 4	TanH	ReLu/TanH	LeakyReLu	LeakyRelu 2	PReLu	SWISH
Accuracy	98.90%	99.53%	99.53%	99.57%	99.72%	99.58%	99.53%
Precision	26.03%	0.00%	0.00%	56.32%	68.52%	55.51%	0.00%
Recall	72.24%	0.00%	0.00%	40.00%	75.51%	53.47%	0.00%
F1-Score	38.27%	0.00%	0.00%	46.78%	71.84%	54.47%	0.00%

Tabla 3. Exp. Funciones activación

Debido a que LeakyRelu genera unos buenos resultados, pero poco estables, se continua con la experimentación para lograr un modelo estable.

IV. Se realizaron experimentos aleatorios para revisar si la función de activación Leaky ReLu es una buena función para los datos, se incrementó el número de neuronas y capas, también se normalizaron los datos y se realizó un cambio en las métricas (Accuracy por Recall)

	LeakyReLu	Leaky + Capas	Leaky + Neuron	Normalizar	Normalizar 2	Cambio metricas
Accuracy	99.57%	99.58%	99.57%	99.62%	99.63%	99.53%
Precision	56.32%	56.22%	55.17%	62.90%	65.50%	50.45%
Recall	40.00%	53.47%	52.24%	47.76%	45.71%	46.12%
F1-Score	46.78%	54.81%	53.67%	54.29%	53.85%	48.19%

Tabla 4. Exp. sobre Leaky ReLu

La normalización incrementó la estabilidad del modelo y su performance en Recall y Precision, por lo que se continua la experimentación con las variables normalizadas.

V. Debido a que el modelo era estable, se probó ajustar el learning rate del optimizador Adam, y al ver mejoras se optó por también cambiar el optimizador a optimizadores que ajustan de manera dinámica el learning rate:

	L.R. 0.01	L.R. 0.0001	L.R. 0.00001	AdaDelta	FTLR	NADAM
Accuracy	99.57%	99.48%	99.19%	99.24%	99.15%	99.55%
Precision	56.32%	46.43%	33.14%	25.41%	28.98%	51.89%
Recall	40.00%	68.98%	68.98%	31.43%	54.29%	55.92%
F1-Score	46.78%	55.50%	44.77%	28.10%	37.78%	53.83%

Tabla 5. Exp. de optimizadores

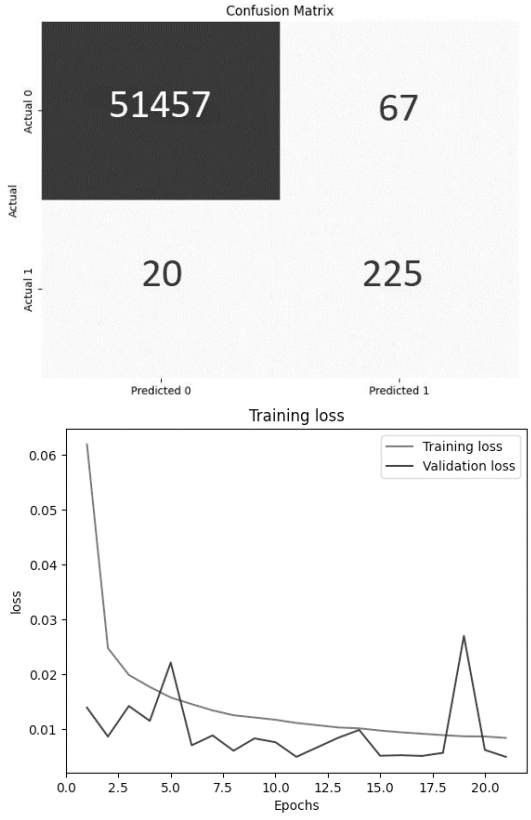
Aunque el optimizador NADAM presenta un buen performance, es inestable en cuanto a la pérdida del modelo en la validación y entrenamiento, por lo que se decide seguir utilizando Leaky Relu con un Learning rate de 0.0001.

VI. Teniendo un modelo normalizado, con una función de activación responsiva para los datos, estable, se incrementaron nuevamente neuronas y capas (primer modelo 3 capas a 64 neuronas + 1, segundo modelo 5 capas a 128 + 1, tercer modelo 7 capas a 512 +1 y último modelo 9 capas a 2048 neuronas +1):

	0.0001	+ Neuron	+Neuron + Capas	+Neuron + Capas	Modelo final
Accuracy	99.48%	99.72%	99.82%	99.84%	99.83%
Precision	46.43%	75.50%	91.71%	78.25%	77.05%
Recall	68.98%	61.63%	67.76%	91.02%	91.84%
F1-Score	55.50%	67.87%	77.93%	84.15%	83.80%

Tabla 6. Exp. Final

El modelo final (8 capas con función de activación Leaky Relu, con 16, 32, 64, 128, 256, 512, 1024, 2048 neuronas, más una capa final de 1 neurona con activación sigmoid, oversampling para balancear el modelo al 5%, variables normalizadas, con métrica objetivo de accuracy, optimizador Adam con Learning Rate de 0.0001) corrió en unas 5 horas a 15 minutos por época, se corrió 3 veces y mantuvo el performance alto, este es el resumen del modelo:



Como se observa, aunque en entrenamiento es estable la pérdida, en validación varía, esto se debe principalmente al oversampling realizado (SMOTE), pero el modelo tiene un muy buen valor de pérdida en validación y oscila sobre valores bajos; La Matriz de confusión demuestra las capacidades del modelo.

D. Conclusions

I. Si los datos no están equilibrados, puede que los modelos de aprendizaje automático no funcionen como se espera, ya que el algoritmo minimiza el error al no clasificar correctamente los datos, lo que podría llevar a clasificarlos todos como negativos. Entre las diversas técnicas disponibles, el oversampling utilizando la técnica SMOTE para aumentar la proporción de la clase positiva al 5% de los datos proporcionó los resultados óptimos en nuestro estudio.

II. La activación es fundamental en el modelo, aunque algunas funciones pueden ser más efectivas para ciertos conjuntos de datos, en la práctica, esto no siempre resulta ser el caso. Un ejemplo de esto se observa en la función SWISH, que se sitúa como un término medio entre Sigmoid y ReLU y se presume capaz de descubrir asociaciones complejas entre las entradas y salidas. Sin embargo, en esta situación específica, se constató que Leaky ReLU funcionó de mejor, evitando así el problema del "Dying ReLU".

III. Incrementar el número de capas y neuronas no constituye una estrategia eficaz si los hiperparámetros del modelo no se encuentran debidamente definidos; en la experimentación, el proceso se repitió en más de tres ocasiones, pero no se obtuvieron resultados prometedores. Sin embargo, una vez que los hiperparámetros del modelo, como las funciones de activación, el learning rate, la normalización, entre otros, fueron ajustados según las características de los datos, se observó un notable incremento del 51% en el rendimiento del modelo.

IV. Conclusion

- Como conclusión general se puede decir se logró mejorar el rendimiento del modelo en la clasificación. Junto con las pruebas que se realizaron para comprobar el buen desempeño del modelo, se pudo observar que la gráfica de fronteras de decisión mostró una buena capacidad de aprendizaje, destacando la idoneidad de la arquitectura del modelo. En conjunto, estos hallazgos respaldan la eficacia del modelo. Esto demostró que a medida que se ajustaban y exploraban diversas combinaciones, se evidenció que un equilibrio adecuado entre el número de capas ocultas y neuronas era esencial para lograr un desempeño

óptimo. La elección de estas estructuras no solo influyó en la capacidad de aprendizaje del modelo, sino también en su capacidad de generalización y adaptación a datos desconocidos.

- Este estudio ha subrayado la importancia de abordar los desequilibrios en los conjuntos de datos, la selección precisa de funciones de activación y la definición cuidadosa de los hiperparámetros para garantizar un rendimiento óptimo en los modelos de aprendizaje automático. Se ha demostrado que el equilibrio de datos mediante técnicas como el oversampling fue crucial para mejorar la capacidad predictiva del modelo. Además, la adaptación específica al contexto en la elección de funciones de activación. Por último, se confirmó que un enfoque más preciso y personalizado en el ajuste de los hiperparámetros del modelo condujo a mejoras sustanciales en la capacidad predictiva. Estos hallazgos resaltan la importancia de considerar estos aspectos para desarrollar modelos de aprendizaje automático robustos y efectivos en la resolución de problemas complejos de clasificación y predicción de datos.

References

[1] Moreno Barbosa, A. M. B. (2023.). *Aprendizaje de Máquina Perceptrón Clase 5*.

[2] Moreno Barbosa, A. M. B. (2023.). *Aprendizaje de Máquina Redes Neuronales Poco Profundas Clase 6.*

[3] Moreno Barbosa, A. M. B. (2023.). *Aprendizaje de Máquina Estrategias de entrenamiento Clase 3*.

[4] Kumawat, P. (s. f.). Better activation functions. www.linkedin.com. https://www.linkedin.com/pulse/better-activation-functions-prince-kumawat/

[5] Nikhade, A. (2023, 7 marzo). Swish Activation Function - Amit Nikhade - Medium. Medium. https://amitnikhade.medium.com/swish-activation-function-d106fe13930e

[6] Satpathy, S. (2023). SMOTE for imbalanced classification with Python. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/overcoming-class-imbalance-using-smote-techniques/