Predicción de ingresos con datos del Censo

Parcial 2 - IA Aplicada a la Economía
Samuel Suárez - Jade Manon Nicolas

Este reporte tiene como objetivo explicar las decisiones tomadas en la construcción de un modelo logit y de una red neuronal para predecir los ingresos de un individuo a partir de sus datos del censo. El documento se organiza en dos secciones principales. En la primera, se analizan los datos disponibles en la base y se aplican las modificaciones necesarias, justificando cada decisión. En la segunda, se emplean estos datos para crear los modelos, explicando la elección de los hiperparámetros utilizados en el proceso.

1. Procesamiento de datos

1.1 Manejo de valores faltantes (missing values)

Nuestro análisis de datos exploratorios sugiere tres conclusiones sobre los valores faltantes en nuestra base de datos. Primero, solo las variables predictoras contienen missing values. No la variable objetivo. Segundo, la cantidad de *missing values* en las variables frente al total de datos es muy pequeña. En los datos de entrenamiento, hay 2106 datos sin un valor determinado. Esto equivale a tan solo el 0.92%. Tercero, tan solo tres variables contienen la totalidad de los missing. Las tres son categóricas, y, como puede ver en las *figuras 1,2, y 3*, para las tres variables, la cantidad de valores faltantes es baja frente a la cantidad de datos en la variable. Estas variables son el tipo de empleo, el tipo de ocupación y el país de orígen.

Figura 1: Distribución de Clases de Trabajo en el conjunto de Entrenamiento

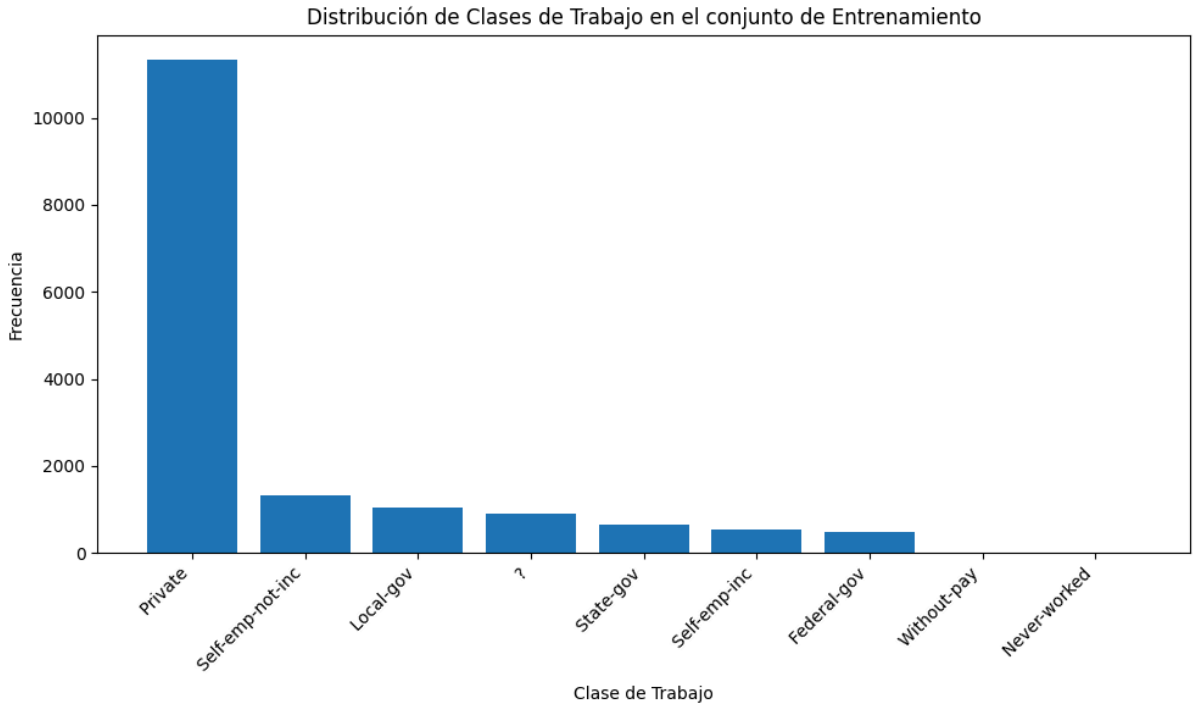


Figura 2: Distribución de Ocupaciones en el conjunto de Entrenamiento

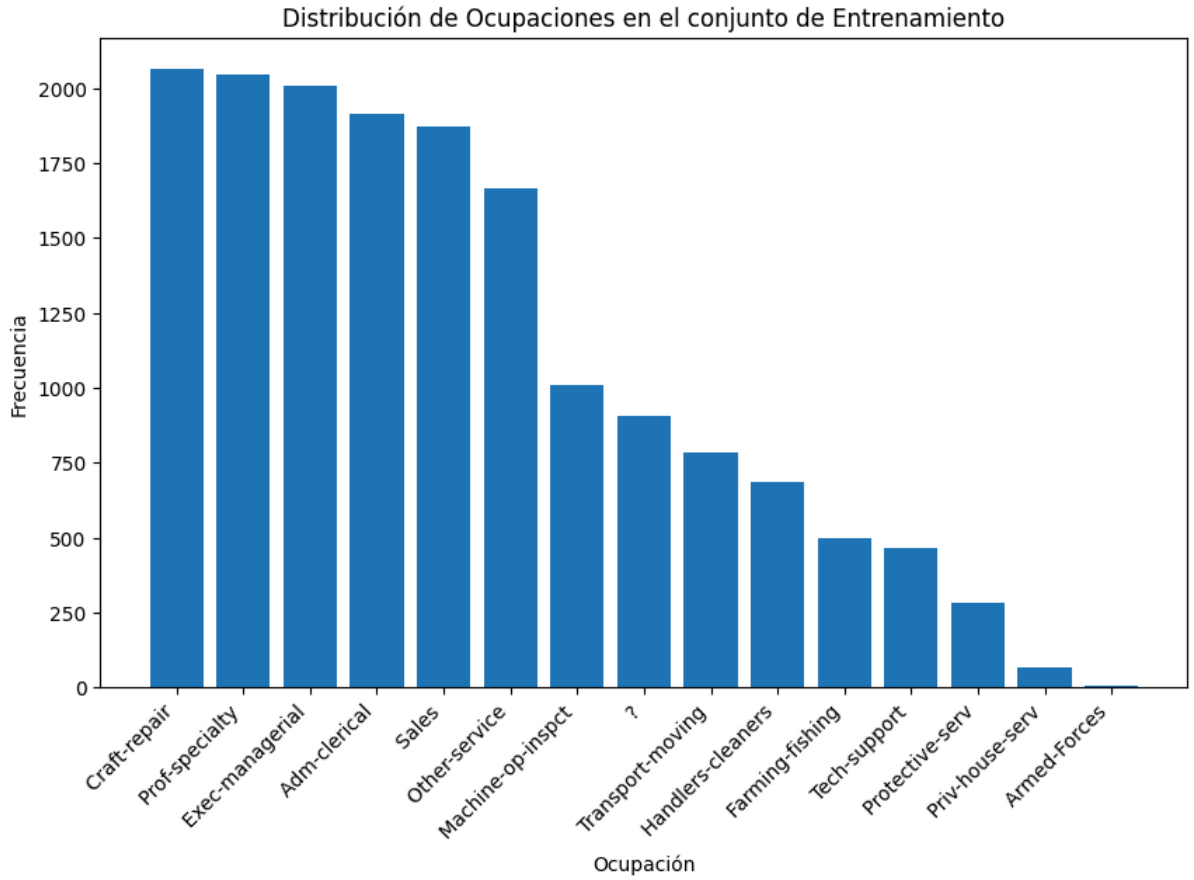
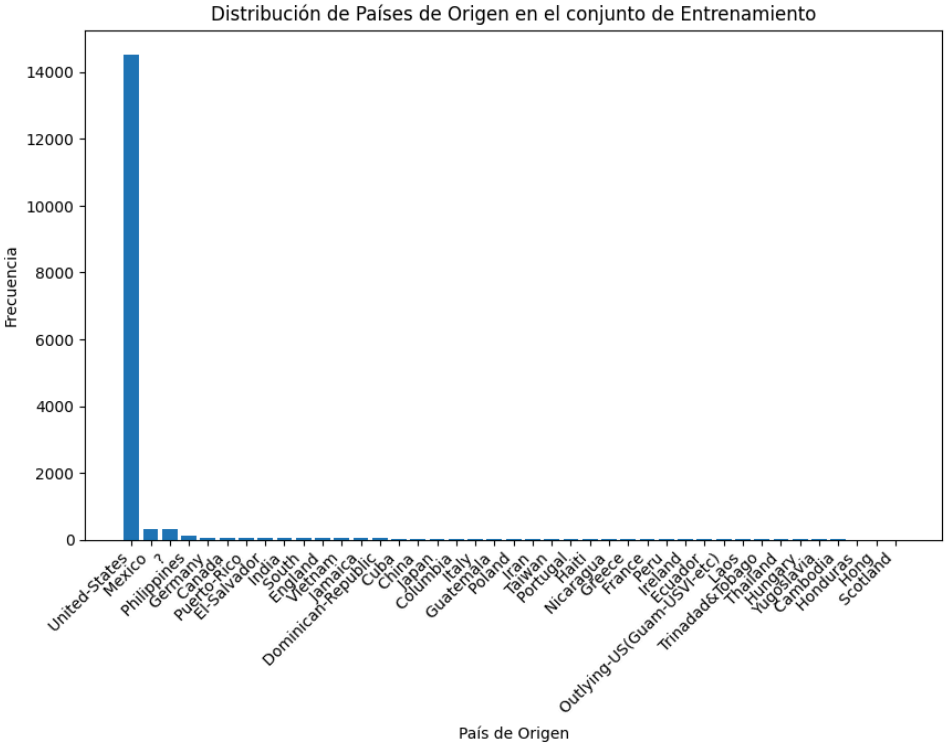


Figura 3: Distribución de Nacionalidades de orígen en el conjunto de Entrenamiento



Para manejar los valores faltantes existen distintas opciones. Gerón (2019), por ejemplo, sugiere que una opción puede ser llenar los valores faltantes, usando datos como el promedio (el cual en el caso categórico no nos sirve) o el dato más popular. Sin embargo, otra sugerencia es que si la cantidad de datos faltantes son muy pocos, podemos eliminar la fila. La segunda opción puede sonar más favorable para nuestros datos, donde los valores faltantes no superan el 1% de los valores totales. Sin embargo, aunque son pocos valores, la cantidad de filas que se terminarían eliminando se aproximan a 1200 o alrededor del 8% de los datos, lo cual no es despreciable. Dado que no son pocos datos, el riesgo de perder información importante y generar un sesgo es moderado. En consecuencia, procedemos a irnos por la primera opción, reemplazando cada categoría con un " ?" (valor faltante) por una que contenga el dato más popular de la variable. Dado que en los algoritmos que usaremos de machine learning no se pueden correr con datos faltantes, es importante corregirlos en los datos de validación también, pero usando la clase más popular para la variable respectiva de los datos de entrenamiento. Esto evita que tengamos *data leakages*, donde terminaríamos usando la categoría más popular de los datos de validación, de los cuales deberíamos asumir que no conocemos su distribución y por ende su moda.

1.2 Manejo de variables

Nuestra base de datos contiene tres parejas de variables a las que les daremos mayor atención, por sus características específicas que pueden traer dilemas a la hora de hacer un modelo de ML. Dos de estas parejas pueden sufrir de una multicolinealidad. En particular, las variables de education y education-num muestran ambas el máximo nivel educativo alcanzado, lo cual puede generar estadísticos con mayor varianza en modelos como un logit. Lo mismo puede llegar a ocurrir con las variables de clases de trabajo y de distribución de ocupaciones. Aunque las cuatro variables tienen información levemente diferenciada, existe el riesgo de multicolinealidad. Sin embargo, aunque el logit puede verse afectado negativamente por esto, un modelo de redes profundas puede incluso beneficiarse de tener la información adicional que ofrecen dos variables así sufran de multicolinealidad. Esto, porque los modelos de redes tienden a ser más flexibles. En estos modelos los pesos se ajustan de manera conjunta mediante optimización no lineal. A diferencia de los modelos lineales, donde los coeficientes se vuelven inestables si las variables son altamente correlacionadas, en una red los parámetros redundantes se pueden redistribuir y adaptar sin afectar la función de predicción. Además, las capas ocultas actúan como transformaciones que combinan y comprimen la información, por lo que la red puede aprender representaciones internas que eliminan la redundancia de forma automática durante el entrenamiento.

Por otro lado, las variables *capital gain* y *capital loss* presentan otro reto. Como puede apreciar en las *figuras 4 y 5*, estas variables tienen un fuerte desbalance, ya que en la mayoría de los registros su valor es 0, lo cual limita la variabilidad y aporta poca información al modelo. Además, en la práctica algunos individuos suelen reportar únicamente ganancias o pérdidas, pero no ambas de manera simultánea. Por ello, una estrategia adecuada es combinar ambas en una sola variable de ganancia neta de capital (*capital gain – capital loss*). Esta

transformación reduce la cantidad de ceros, aumenta la dispersión de los valores observados y concentra la información relevante en una única dimensión, lo que potencialmente mejora la capacidad predictiva del modelo. Procedemos entonces a implementar esta variable en nuestros datos, reemplazando las variables de ganancia y perdida por sí solas.

Figura 4: Distribución de Ganancia de Capital en el conjunto de entrenamiento

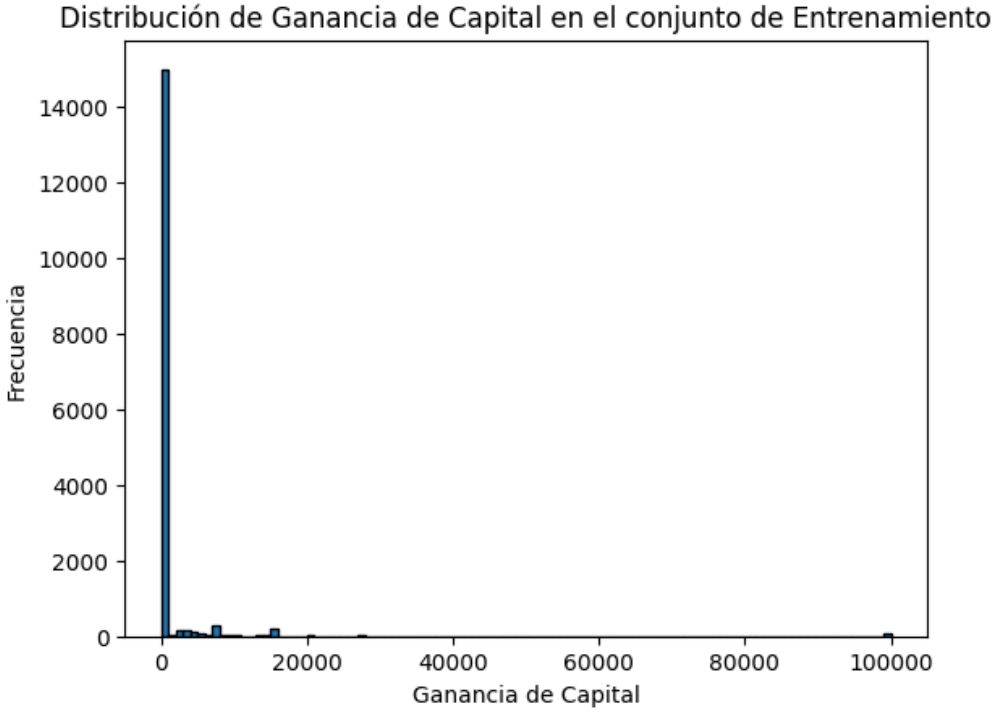
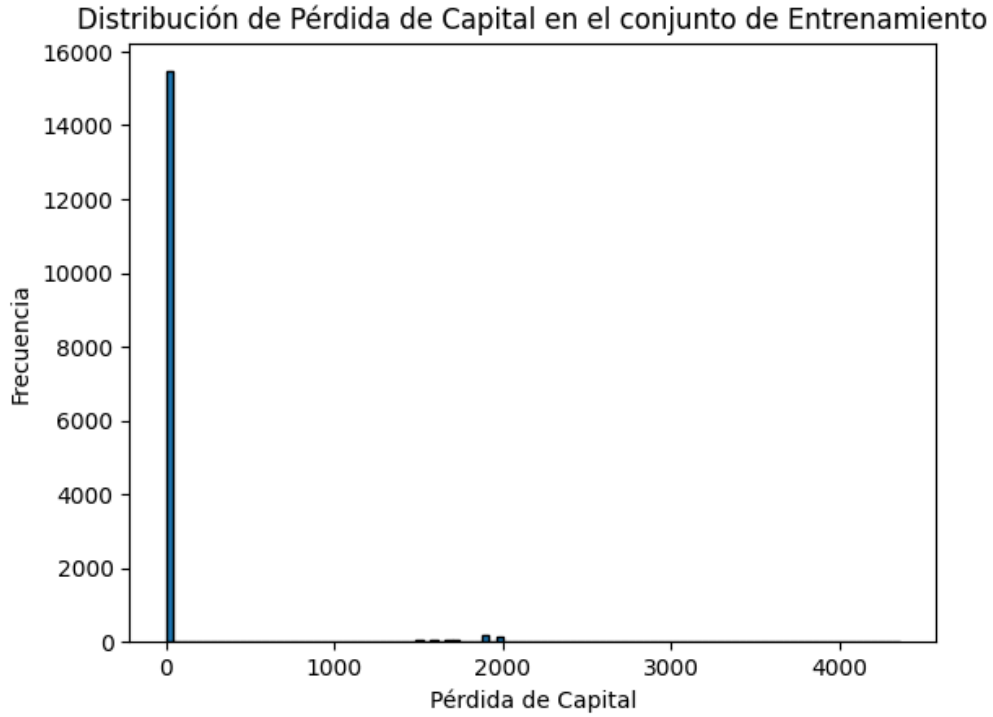


Figura 5: Distribución de Pérdida de Capital en el conjunto de entrenamiento



Finalmente, tenemos otro desbalance evidente con la variable de nacionalidades (vea la *figura 3*). El hecho de que existan tan pocos individuos para la mayoría de los países, puede llevar a que el modelo se sobreajuste alrededor de estos datos para los cuales tiene una muestra baja. Para evitar este riesgo, transformamos la variable, dejando una categoría para estadounidenses y otra categoría para migrantes, que incluye a todos aquellos quienes tienen una nacionalidad distinta a Estados Unidos.

1.3 One hot encoding

Dada la alta cantidad de variables categóricas definidas como strings en nuestra base de datos, procedemos a realizar un one hot encoding con cada una de estas. Lo anterior nos da categorías con valores respectivos de 0 o 1. Esto evita que tengamos categorías con valores numéricos con valores que no aportan ninguna información.

1.4 Estandarización

Ahora todas nuestras variables categóricas tienen valores entre 0 y 1, lo cual evita también posibles errores al realizar un backwards propagation. Si los errores tuviesen valores fuera de este rango, podríamos correr el riesgo de que al realizar el descenso de la gradiente, las primeras capas obtengan gradientes excesivamente altos (resultado de usar la regla de la cadena), lo cual llevaría a que la gradiente explote hacia el infinito. Esto haría que crear la red fuese imposible. Nuestras variables continuas, sin embargo, no tienen esta corrección todavía. Por lo tanto, estandarizamos sus valores, asumiendo que se distribuyen normalmente, lo cual nos permite implementar nuestros modelos con mayor facilidad.

2 Desarrollo de algoritmos

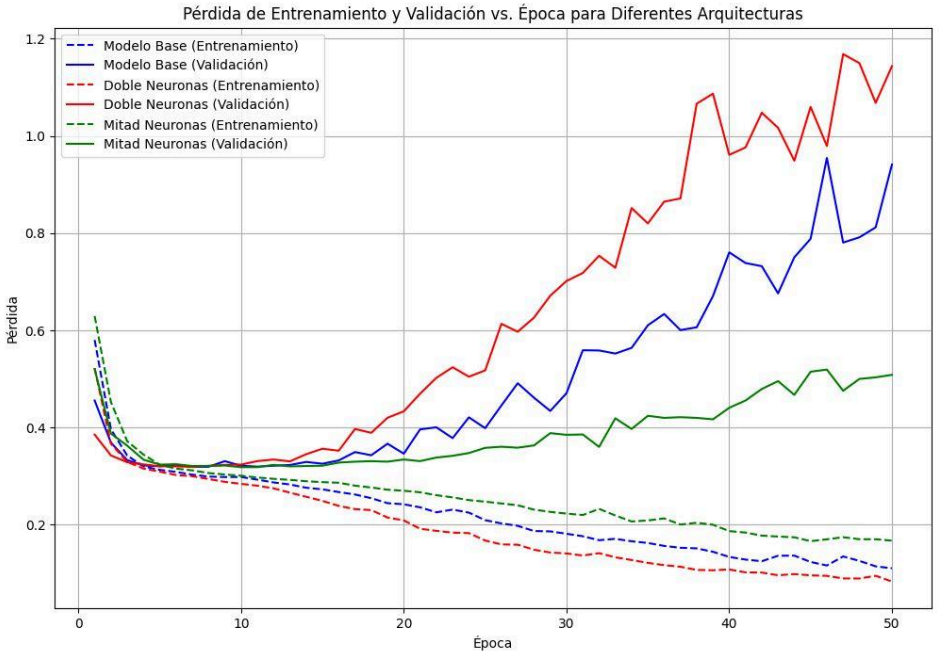
2.1 Experimentos para modelo sin regularizar

Comparamos cinco configuraciones de redes neuronales con el objetivo de evaluar cómo distintas decisiones de diseño impactan en el rendimiento del modelo. En particular, analizamos la influencia de las funciones de activación, la profundidad y anchura de la arquitectura, el tamaño del batch y la tasa de aprendizaje. El seguimiento de la pérdida en entrenamiento y validación nos permitió identificar fenómenos de *overfitting* y *underfitting*, y así determinar cuál configuración generaliza mejor. Como punto de partida utilizamos un modelo base con las siguientes especificaciones: un tamaño de entrada (*input_size*) de 64, seis capas ocultas (*hidden_layers*) con 256 neuronas cada una (*hidden_neurons*), función de activación ReLU (con posibilidad de ser sustituida por LeakyReLU), salida binaria (*output_size* = 1) mediante logits, sin *dropout* inicial, una tasa de aprendizaje de 0.001, tamaño de batch de 1024 y 50 épocas de entrenamiento. Estos parámetros corresponden a un conjunto de recomendaciones propuestas por Géron (2019) y sugeridas por la literatura en inteligencia artificial. La intención no es que estos parámetros iniciales sean óptimos, sino utilizarlos como un punto de referencia realista y consistente con prácticas comunes en redes neuronales, a partir del cual podamos ajustar y encontrar configuraciones más adecuadas.

2.1.1 Experimento 1

El modelo con la mitad de neuronas mantiene una pérdida de validación más baja y estable, mientras que los modelos con el doble de neuronas que el "base", muestran overfitting a partir de la época 20. Así, aumentar la cantidad de neuronas no parece mejorar la capacidad de generalización en este caso.

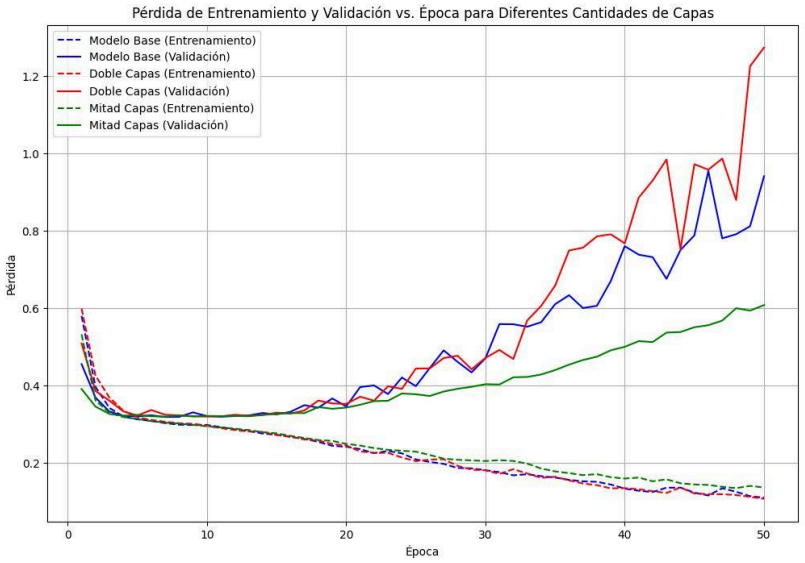
Figura 6: Pérdida de entrenamiento y validación vs época para diferentes cantidades de neuronas



2.1.2 Experimento 2

El modelo con la mitad de capas mantiene la pérdida de validación más controlada. Los modelos con más capas empiezan a sobre ajustarse a partir de la época 20, con curvas de validación que aumentan mientras las de entrenamiento bajan, sugiriendo que el modelo está memorizando los datos sin poder aplicarse a datos nuevos.. Una arquitectura más sencilla parece ser más eficaz en este caso.

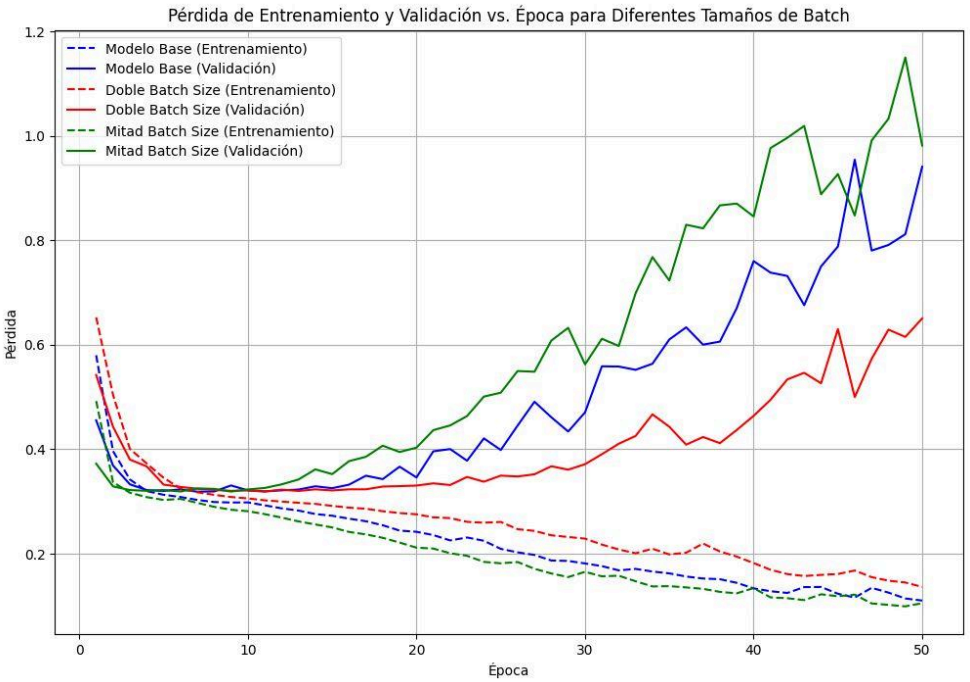
Figura 7: Pérdida de entrenamiento y validación vs época para diferentes cantidades de capas



2.1.3 Experimento 3

El modelo con el doble del *batch size* tiene una pérdida de validación más baja y constante que las otras curvas. El modelo con mitad de batchsize en cambio, muestra más variabilidad y pérdida más alta en validación. Para mejorar la estabilidad y la generalización, sería una buena opción aumentar el tamaño del batch.

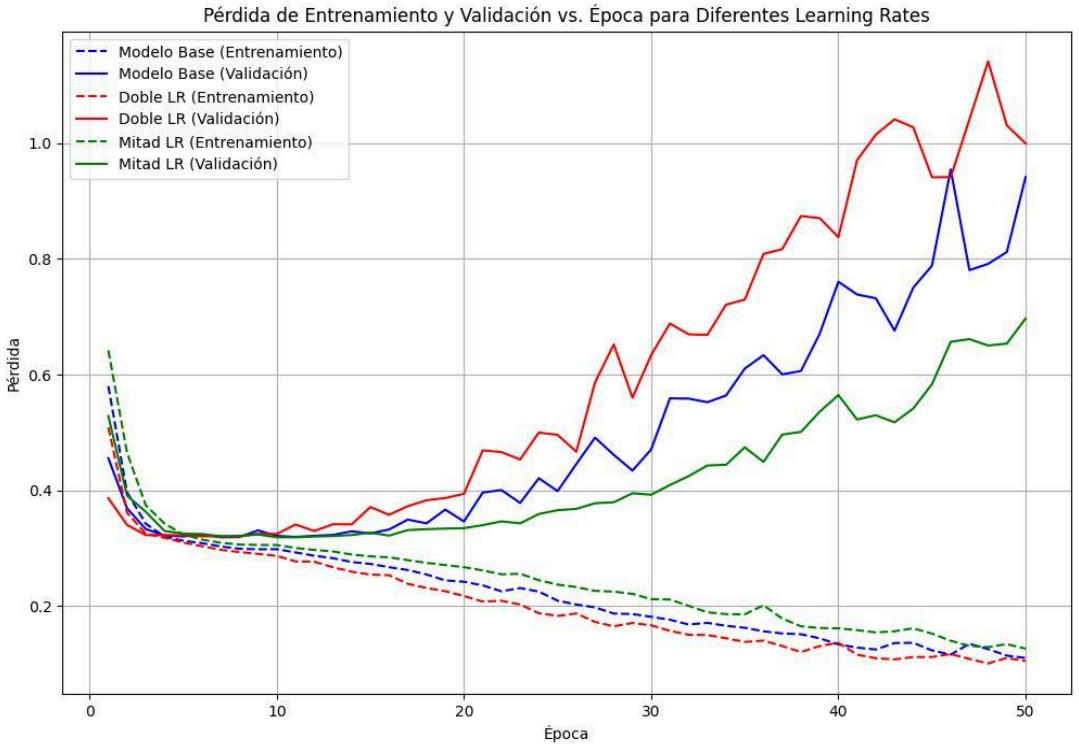
Figura 8: Pérdida de entrenamiento y validación vs época para diferentes tamaños de batch



2.1.4 Experimento 4

El modelo con mitad de learning rate mantiene la pérdida de validación más baja y constante. Las configuraciones con tasa base y doble presentan overfitting desde la época 20, con curvas de validación que suben mientras las de entrenamiento siguen bajando. Una tasa de aprendizaje más baja permite un ajuste más progresivo y estable.

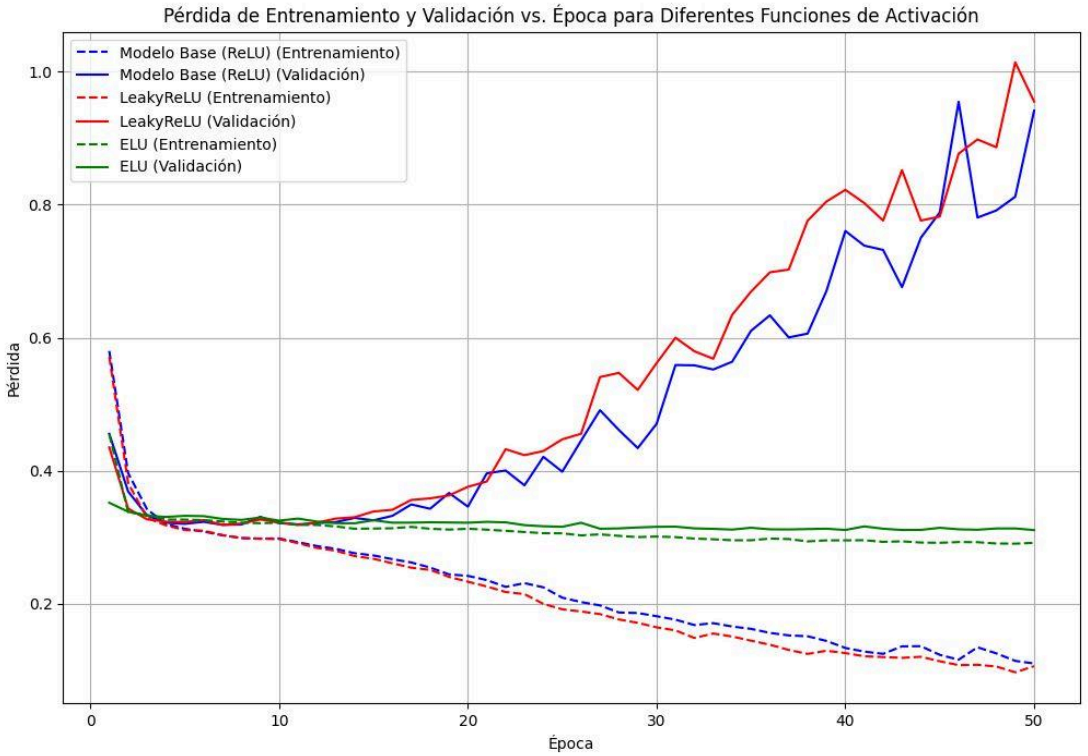
Figura 9: Pérdida de entrenamiento y validación vs época para diferentes learning rates



2.1.5 Experimento 5

En este experimento se comparan tres funciones de activación: ReLU, LeakyReLU y ELU. ReLU reduce bien la pérdida de entrenamiento, pero su curva de validación sube desde la época 10, lo que indica overfitting. LeakyReLU sigue un patrón similar, aunque el sobreajuste es menos marcado. ELU, en cambio, mantiene pérdidas bajas y estables en ambas curvas, sin señales de sobreajuste ni subajuste. Según el código, es el modelo con menor pérdida en validación, lo que confirma su buen rendimiento. Podemos concluir que de los tres modelos, ELU es el que logra el mejor equilibrio entre aprendizaje y generalización. Su rendimiento es consistente y evita tanto el sobreajuste como el subajuste, lo que representa la opción más eficaz en este experimento.

Figura 10: Pérdida de entrenamiento y validación vs época para diferentes funciones de activación



2.1.6 Mejor modelo sin regularizar

Con base en los cinco experimentos del modelo sin regularización, el mejor compromiso entre aprendizaje y generalización es el que combina: activación ELU, mitad de capas (3 en lugar de 6), mitad de neuronas por capa (128 en lugar de 256), batch size doble (2048) y learning rate a la mitad (0.0005). Esto, pues en el experimento 1, usar la mitad de neuronas mantiene la pérdida más baja y estable; en el experimento 2, la mitad de capas controla mejor la pérdida y evita el sobreajuste; en el experimento 3, duplicar el batch reduce la pérdida y su variabilidad; en el experimento 4, la mitad del learning rate logra la curva de valoración más baja y constante; y, finalmente, en el experimento 5, ELU obtiene la menor pérdida de validación sin señales de sobre/sub ajuste. En conjunto, estas elecciones minimizan la pérdida en validación y mitigan el overfitting observado en configuraciones más grandes o más agresivas. Finalmente, buscamos la época del modelo que tenga la menor pérdida en nuestros datos de validación. Encontramos que este modelo puede seguirse re-entrenando sin tender rápidamente al sobreajuste, por lo cual hasta la época 235 se obtiene el punto óptimo.

2.2 Datos regularizados

Habiendo regularizado nuestros datos, podemos pasar a elegir el mejor modelo de redes dados todos nuestros análisis. El modelo que obtendremos, no necesariamente es el modelo óptimo, pues nuestros experimentos no consideran todas las combinaciones de hiperparametros posibles. Sin embargo, buscamos basarnos en nuestras conclusiones para obtener un modelo ideal.

2.2.1 Elección de cantidades de neuronas por capa y cantidades de capas óptimas

Las *Tablas 1 y 2* presentan los resultados obtenidos al aplicar early stopping variando la cantidad de neuronas por capa y el número de capas, manteniendo constantes los demás parámetros. Se observa que, una vez regularizado el modelo, añadir más neuronas no genera una reducción significativa en la función de pérdida; en cambio, sí conduce a una disminución en el F1-score, métrica que utilizamos dado el desbalance de clases. El comportamiento es distinto cuando se incrementa el número de capas: en este caso, la reducción en la función de pérdida resulta más notoria, mientras que la caída en el F1-score es menos marcada. Esto sugiere que aumentar la profundidad del modelo tiene un efecto más favorable en la reducción del log-loss, sin deteriorar de manera tan fuerte la capacidad de acierto medida por el F1-score. En consecuencia, si queremos minimizar nuestra pérdida, sin reducir demasiado el F1-score, podría ser preferible incrementar moderadamente la cantidad de capas, aceptando un costo menor en el F1-score a cambio de una reducción más clara en las pérdidas. Es por esto, que pasamos de las seis capas ocultas de nuestro modelo base a 8, dejando la cantidad de neuronas por capa igual (256).

Tabla 1: Comparación de modelos regularizados con distintas cantidades de neuronas por capa (en su mejor estado)

Modelo	Pérdida de Validación	F1 Score de Validación
Base Regularizado	0.322235	0.692590
Doble Neuronas Regularizado	0.326914787	0.673640743
Mitad Neuronas Regularizado	0.32005524635	0.673922143

Tabla 2: Comparación de modelos regularizados con distintas cantidades de capas (en su mejor estado)

Modelo	Pérdida de Validación	F1 Score de Validación
Base Regularizado	0.322235	0.692590
Doble Batch Regularizado	0.3242183924	0.6674188999
Mitad Batch Regularizado	0.3172779679	0.6797313965

2.2.2 Elección de un batch size óptimo

La *Tabla 3* nos muestra los resultados que obtenemos en cuanto a nuestra función de pérdida y f1-score para el mejor modelo usando métodos regularizados, al duplicar nuestro batch size (a 2048), dividirlo en dos (a 512) y manteniéndolo igual (en 1024). Note que obtenemos la

menor pérdida con el modelo base, que no modifica el batch size. Este modelo también nos da el mayor F1 score, por lo cual mantenemos nuestro batch size en 1024.

Tabla 3: Comparación de modelos regularizados con distintas batch sizes (en su mejor estado)

Modelo	Pérdida de Validación	F1 Score de Validación
Base Regularizado	0.322235	0.692590
Doble Batch Regularizado	0.324186	0.672087
Mitad Batch Regularizado	0.328972	0.673381

2.2.3 Elección de un learning rate óptimo

En la *Tabla 4* se presentan los resultados obtenidos al evaluar nuestras métricas con tres configuraciones de learning rate: la mitad del valor base (0.0005), el doble (0.002) y el valor original (0.001). Se observa que el mejor F1-score se alcanza con el learning rate base, mientras que la menor pérdida corresponde al modelo con la tasa reducida a la mitad. Sin embargo, reducir el learning rate a la mitad da muy pocos beneficios a la función de pérdida. Aunque el efecto en la F1 Score también es marginal, es mayor, por lo cual cambiar el learning rate parece ser un sacrificio que no vale la pena. Es por esto que lo mantenemos en su valor base de 0.001.

Tabla 4: Comparación de modelos regularizados con distintos learning rates (en su mejor estado)

Modelo	Pérdida de Validación	F1 Score de Validación
Base Regularizado	0.322235	0.692590
Doble LR Regularizado	0.323499	0.677165
Mitad LR Regularizado	0.320274	0.689527

2.2.4 Elección de una función de activación óptima

Finalmente, la tabla 5 compara las distintas funciones de activación que podemos implementar en nuestro modelo. En particular decidimos probar las funciones de LeakyReLu

y ELU, pues estas tienden a evitar la muerte de neuronas, que es una consecuencia de que la ReLu impide que pasen neuronas con valores negativos. Dependiendo del modelo, esto puede ser un problema, por lo cual lo decidimos probar. Notamos que para ambas métricas, el mejor resultado lo da la función de activación ELU, lo cual sugiere que la muerte de neuronas puede ser importante en este ejercicio. Es por esto, que nuestro modelo final utiliza esta función.

Tabla 5: Comparación de modelos regularizados con distintas funciones de activación (en su mejor estado)

Modelo	Pérdida de Validación	F1 Score de Validación
Base Regularizado (ReLU)	0.322235	0.692590
LeakyReLU Regularizado	0.323021	0.687200
ELU Regularizado	0.317931	0.696554

2.2.5 Modelo final

Finalmente, elegimos un modelo con las siguientes características. Tendrá 8 capas ocultas, 256 neuronas por capa oculta, tendrá un propagation con batch sizes de 1024 y un learning rate de 0.001 y usará una función de activación ELU.

2.3 Comparación de modelos

Para evaluar el rendimiento de los modelos propuestos, comparamos sus métricas principales: exactitud, precisión, recall y F1-score en las fases de entrenamiento, validación y prueba. Principalmente, usamos el F1-score que es especialmente útil para evaluar el equilibrio entre precisión y recall, dado que hay un desbalance de clase (solo el 24% de los datos de prueba tienen ingresos mayores a 50,000 USD).

Al analizar los resultados obtenidos, vemos que el modelo sin regularización tiene el mejor desempeño en el conjunto de entrenamiento con un F1 de 0,72 y una precisión ligeramente superior a la de los otros modelos; lo que indica un excelente ajuste a los datos iniciales. Sin embargo, la ligera caída de este valor en la validación y la prueba puede indicar un posible sobreajuste, es decir que el modelo aprende demasiado los datos de entrenamiento y no generaliza tan bien con datos nuevos. Esto es consistente con el hecho de que no esté regularizado, lo cual lo hace en general peor en predecir personas con salarios mayores a 50k. Esto está evidenciado en los menores valores de exactitud y recall. Por su parte, el mejor modelo regularizado presenta un F1-score alto en los tres conjuntos (entrenamiento, prueba y validación), que es el más alto de los tres modelos en los conjuntos de validación y prueba. Esto refleja un equilibrio más estable entre precisión y recall y una mejor capacidad de generalización.

En comparación con el modelo base, podemos ver que a diferencia de los modelos de redes neuronales con y sin regularización, el logit tiende a un menor sobreajuste, pues sus métricas de predicción disminuyen poco entre los conjuntos de entrenamiento y prueba. Sin embargo, este modelo es peor que los otros dos en todas las métricas, lo cual sugiere que a menudo falla en predecir valores de ambas clases. Entonces, aunque los otros modelos tienden a sobre ajustarse, incluso con regularización, siguen teniendo mayores capacidades predictivas que el modelo de base, lo cual sugiere que las redes neuronales son mucho más poderosas a la hora de predecir ingresos usando datos del censo. De todas formas, los modelos siguen siendo similares y las diferencias entre estos no exceden unas centésimas, por lo cual el ahorro en poder de cómputo que proviene de usar un modelo logit no debe ser despreciado, dada su capacidad similar de predicción.

En resumen, las diferencias entre los modelos no son muy amplias pero nos dicen que las redes neuronales son efectivamente mejores en predecir datos de ingresos binarios, usando datos del censo. El modelo regularizado, efectivamente tiende al sobreajuste, sin embargo sus métricas de evaluación no se distancian mucho del modelo regularizado (aunque tampoco son mucho mejores que el logit). Esto sugiere que los tres modelos tienen poderes predictivos similares tanto para predecir valores positivos como negativos. Igualmente, en todas las métricas de prueba y validación, el modelo que mejores resultados ofrece es la red neuronal regularizada. Por eso, podemos decir que este último se presenta como la opción más adecuada para resolver nuestro problema de categorización.

Tabla 6: Estadísticas de medición para modelo base y para las mejores redes neuronales

Modelo	Exactitud Entrenamient	Precisión Entrenamient	Recall Entrenamient	F1 Entrenamient	Exactitud Validación	Precisión Validación	Recall Validación	F1 Validación	Exactitud Prueba	Precisión Prueba	Recall Prueba	F1 Prueba
Regresión Logística (Base)	0.84607	0.72240	0.58478	0.64634	0.84780	0.72186	0.59975	0.65516	0.84608	0.71149	0.58606	0.64271
Mejor Modelo Regularizado	0.86333	0.74744	0.65220	0.69658	0.85695	0.72581	0.65350	0.68776	0.85449	0.71613	0.63625	0.67383
Mejor Modelo Sin Regularización (época 235)	0.87899	0.80161	0.66037	0.72417	0.85302	0.73497	0.61045	0.66695	0.85363	0.72996	0.60374	0.66088

Referencias:

Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems (2nd ed.).O'Reilly.