Escuela de ingeniería y ciencias.

Campus **Monterrey**

Unidad de formación **TC3006C.102**

Inteligencia artificial avanzada para la ciencia de datos II

Semana 1: Dense Layers

Equipo **3**

Grupo: **102**

Regina Reyes Juárez - **A01275790**
Nadia Salgado Alvarez - **A01174509**
Gilberto Angel Camacho Lara - **A01613895**
Santiago Miguel Lozano Cedillo - **A01198114**

06 de octubre

Tarea_1_Deep.ipynb

Introducción.
Actualmente, es bien sabido que el rendimiento académico no solamente está ligado con las capacidades mentales de los estudiantes sino también con distintos factores de su vida personal. Por ello, nuestro objetivo es determinar si a través de variables como las horas de estudio, las horas de sueño, la asistencia a clase y sus calificaciones pasadas se puede aproximar o estimar cuál podría ser el desempeño académico actual de un estudiante.

Para este análisis utilizaremos la información disponible en la base de datos "student_exam_scores.csv", la cual nos proporciona un conjunto de variables que permiten validar o cuestionar la creencia común de que la calificación no depende únicamente de las capacidades cognitivas, sino también de los hábitos y condiciones del alumno.

Con el fin de abordar este problema, se implementará un modelo de red neuronal de capas densas (Multilayer Perceptron, MLP), ya que este tipo de arquitecturas son capaces de capturar relaciones no lineales entre variables y ofrecen un buen balance entre complejidad y capacidad de generalización en datasets de tamaño medio como el utilizado en este estudio.

Exploración, explicación y limpieza de datos.
La base de datos utilizada en este trabajo corresponde al conjunto "Student exam score dataset analysis" disponible en Kaggle (https://www.kaggle.com/datasets/grandmaster07/student-exam-score-dataset-analysis) . Este dataset contiene 6 columnas, de las cuales 4 son float64, una int64 y una más de tipo object. De igual manera contamos con 200 entradas en las cuales, en ninguna se presentan nulos por lo que no es necesaria ningún tipo de imputación de datos.

Para el tratamiento de los datos se eliminó la columna de "student_id" debido a que no aporta ningún tipo de información valiosa para el modelo, debido a que este es un identificador único de cada estudiante y no tenemos información sobre si se puede extraer algo interesante o importante sobre este valor como carrera, ubicación o demás.

Como se mencionaba antes, no contamos inicialmente con valores nulos ni tampoco contamos con filas duplicadas, se realizó una normalización y una estandarización de la información en la base de datos.

Posteriormente, como se observa en la figura 1, podemos observar que dentro de la variable objetivo del score tenemos un claro desbalance de clases, en el caso de agrupar los datos como se ve a continuación, lo cual puede llevar a que las redes neuronales, técnica con la cuál se plantea el análisis, falle al predecir en los extremos y solo aprenda a predecir valores promedio.

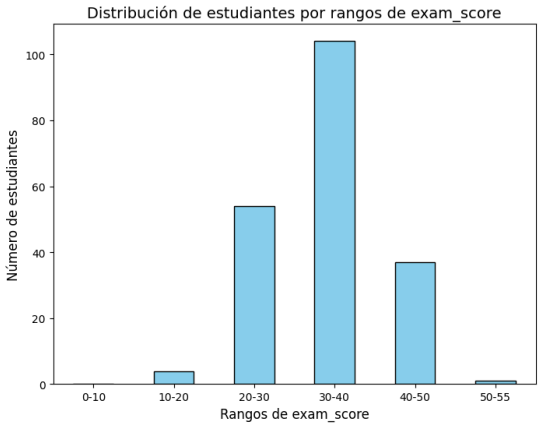

Figura 1. Gráfica comparativa de estudiantes por calificación.

Por esta razón, se decidió hacer un balance entre los rangos dejando la variable *df_balanced* lista para entrenar y probar un modelo utilizando redes neuronales

En la implementación de la red neuronal se decidió probar distintos hiperparámetros basados en el análisis exploratorio previo. Dado que el dataset es pequeño (alrededor de 200 observaciones) y no contiene valores negativos, se puede inferir que no se requieren muchas capas para evitar overfitting, además de que no existe una alta correlación entre las variables.

Por ello, el modelo se diseñó con una arquitectura de dos capas densas (32 y 16 neuronas), suficiente para capturar relaciones no lineales sin sobreajustar.

Descripción de los Hiperparámetros

1. **Capas (32–16)**

    - Se eligió una red compacta que equilibra complejidad y capacidad de generalización.

    - Modelos con más capas no ofrecían mejoras significativas y tendían al sobreajuste.

2. **Funciones de activación**

    - **ReLU:** utilizada porque los datos son positivos y permite aprender relaciones no lineales entre las variables (por ejemplo, entre horas de estudio y desempeño).

    - **Linear:** usada como línea base (“baseline”), útil para comparar en caso de que las relaciones entre las variables sean esencialmente lineales.

3. **Inicialización de pesos**

    - **He Normal:** mantiene estable la varianza de las activaciones, evitando que los gradientes exploten o se desvanezcan. Ideal para ReLU.

    - **Glorot (Xavier):** más apropiada para funciones simétricas (tanh, sigmoid). Puede ser menos estable en redes con capas ReLU puras.

4. **Tamaño del batch**

    - **Batch = 20:** tamaño pequeño, lo que permite actualizaciones más frecuentes del gradiente y aprendizaje más fino.

    - **Batch = 120:** más rápido, pero tiende a suavizar en exceso los gradientes, adecuado para datasets grandes.

5. **Regularización**

    - **EarlyStopping:** detiene el entrenamiento cuando la pérdida de validación deja de mejorar, evitando el sobreentrenamiento.

Función de costo y optimizador

El modelo fue compilado utilizando la función de pérdida categorical_crossentropy, ya que se trata de un problema de clasificación multiclase en el que es necesario comparar la distribución real de clases con las probabilidades predichas por la red. Esta función penaliza con mayor fuerza los errores de clasificación y permite un entrenamiento estable al trabajar junto con la activación softmax en la capa de salida.

Por otro lado, se empleó el optimizador Adam, debido a que combina las ventajas de AdaGrad y RMSProp, ajustando dinámicamente la tasa de aprendizaje para cada parámetro. Esto lo convierte en un método eficiente y robusto, especialmente en

datasets de tamaño medio como el utilizado en este proyecto, asegurando una convergencia más rápida y estable.

Selección de mejores hiper parámetros

Hiperparámetro	**Valor Óptimo**	**Justificación**
Capas / Unidades	(32,16)	Capacidad media; evita sobreajuste
Función de activación	ReLU	Buen equilibrio entre precisión y recall
Inicialización de pesos	HeNormal	Estabiliza gradientes y mejora convergencia
Batch Size	20	Gradientes más ricos y entrenamiento más preciso
Regularización	EarlyStopping + LR Scheduler	Controla el sobreentrenamiento
Métrica clave (F1-Macro)	≈ 0.93	Buen equilibrio entre precisión y recall

La activación ReLU funciona mejor en este caso porque el problema presenta una estructura no lineal (como se observa en la figura 2). Esto significa que la relación entre las variables de entrada y la calificación final no puede ser capturada correctamente por un modelo puramente lineal.

En consecuencia, la complejidad del modelo debe reflejar la complejidad de los datos, y una red neuronal con ReLU permite modelar esas relaciones no lineales de forma eficiente, sin requerir una arquitectura excesivamente profunda.

Esto tiene sentido, ya que variables como horas de sueño, porcentaje de asistencia y calificaciones previas no afectan el resultado de manera estrictamente lineal; su impacto puede variar dependiendo del comportamiento del estudiante. Por ello, una red con ReLU fue adecuada para capturar estos patrones complejos y no lineales.

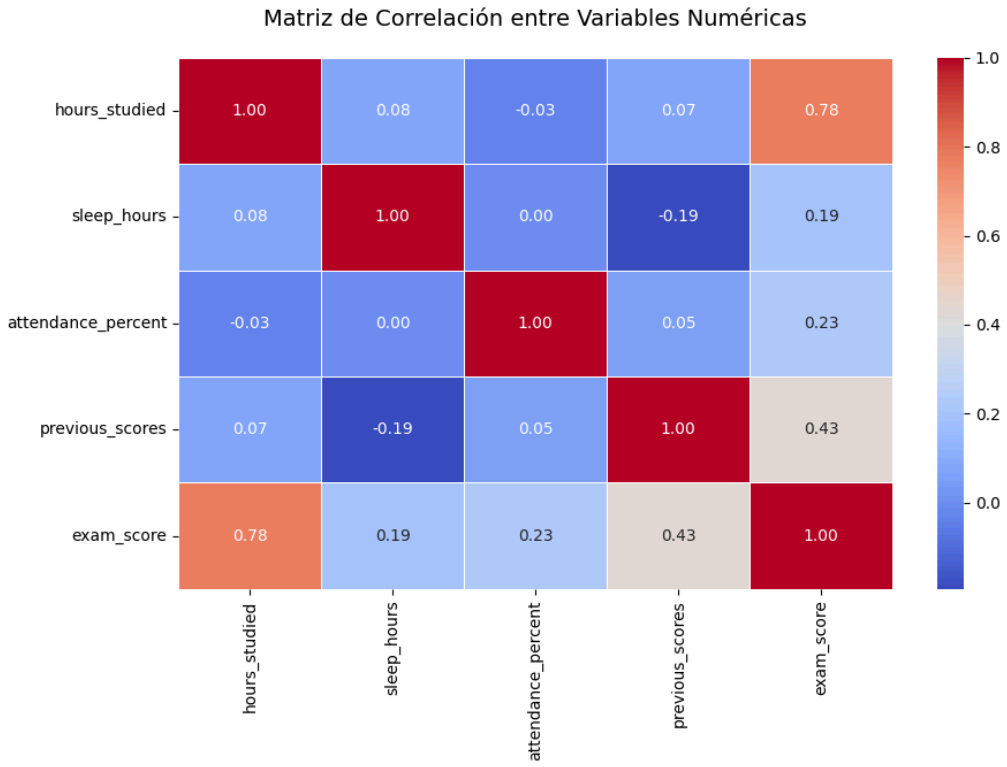

Figura 2. Matriz de correlación entre las variables.

Al analizar las gráficas de pérdida (loss) y precisión (accuracy) en la figura 3, se observa que ambas curvas presentan una fase inicial de crecimiento rápido y posteriormente alcanzan una zona de estabilidad conforme avanzan las épocas.
Esto indica que el modelo logra aprender de manera progresiva y controlada, reduciendo el error de entrenamiento hasta llegar a un punto donde las mejoras adicionales son mínimas.

Este comportamiento refleja un equilibrio adecuado entre aprendizaje y generalización, lo que sugiere que los hiperparámetros seleccionados,como el número de capas, el tipo de activación, el tamaño del batch y los métodos de regularización— fueron efectivos para evitar tanto el subajuste como el sobreajuste.

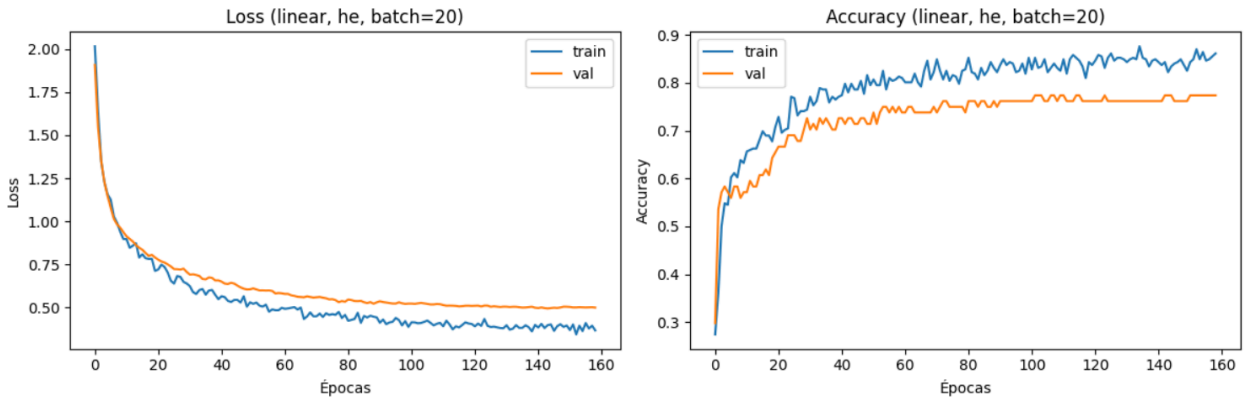

Figura 3. Gráficas de pérdida y accuracy durante las épocas

Interpretación

Al comparar las distintas configuraciones evaluadas, se observa que la combinación ReLU + He Normal + batch de 20 obtuvo los mejores resultados, alcanzando una Accuracy de 0.933 y un F1-Macro de 0.932. Esto significa que el modelo no solo tuvo un buen porcentaje de predicciones correctas, sino que también logró un equilibrio adecuado entre precisión y recall en todas las clases, algo fundamental dado que el dataset presenta cierto desbalance. La activación ReLU resultó ser más efectiva que Linear porque permitió capturar relaciones no lineales entre las variables de estudio y desempeño académico, mientras que la inicialización He contribuyó a mantener estables los gradientes y acelerar la convergencia. El batch pequeño (20) favoreció un aprendizaje más fino al actualizar los pesos con mayor frecuencia, lo que se refleja en curvas de entrenamiento más estables y en una mejor generalización.

En contraste, las configuraciones con Glorot (Xavier) y batch de 120 lograron métricas menores (F1-Macro entre 0.89 y 0.91) y mostraron una diferencia más marcada entre las curvas de entrenamiento y validación, lo que sugiere un mayor riesgo de sobreajuste o menor capacidad de ajuste fino a los patrones de los datos. Asimismo, los modelos con activación lineal, aunque útiles como línea base, no lograron capturar la complejidad de las interacciones entre variables, quedando por debajo en desempeño respecto a las configuraciones con ReLU.

El análisis de las gráficas de pérdida y precisión confirma estas observaciones: en el caso del mejor modelo, tanto la curva de entrenamiento como la de validación muestran un crecimiento sostenido y alcanzan una fase de estabilidad después de varias épocas, lo que indica un aprendizaje progresivo y controlado. Además, el uso de EarlyStopping evitó entrenamientos excesivos, reduciendo el riesgo de sobreajuste y asegurando que el modelo se detuviera en el punto óptimo de desempeño.

Finalmente, se decidió reportar el F1-Macro como métrica principal por encima de la Accuracy, debido a que este indicador es más adecuado en situaciones con clases desbalanceadas, ya que evalúa el rendimiento de manera equilibrada en cada clase. Gracias a este criterio, se pudo confirmar que el modelo no solo predice bien los casos más frecuentes, sino que también mantiene un buen nivel de desempeño en los casos menos representados, lo que refuerza la validez del modelo seleccionado.

Conclusión

El modelo neuronal propuesto logró predecir con alta precisión el desempeño académico, alcanzando una Accuracy de 0.933 y un F1-Macro de 0.932, lo que evidencia un equilibrio sólido entre precisión y generalización. Se evaluaron cuatro combinaciones distintas de hiper parámetros, de las cuales la configuración óptima red compacta (32–16), activación ReLU, inicialización He Normal, batch pequeño y regularización con EarlyStopping ofreció los mejores resultados al capturar relaciones no lineales entre las variables sin perder estabilidad en el entrenamiento.

Estos resultados confirman que el problema presenta una estructura no lineal, por lo que una red compacta y bien regularizada resulta suficiente para modelar los patrones relevantes sin caer en sobreajuste. En conjunto, los hallazgos demuestran la efectividad del proceso de selección de hiperparámetros y la pertinencia de usar ReLU para representar con mayor realismo la complejidad del fenómeno estudiado.

Liga al repositorio:
https://github.com/santiagolc02/deep_learning