TFM – Mario Novella Murillo

Predicción del rendimiento académico de estudiantes.

Una introducción al *Knowledge Tracing*.

Índice

Introducción

Se puede consultar este TFM en el siguiente enlace de GitHub:
https://github.com/MarioMNM/TFM-Prediccion_rendimiento_academico

En este trabajo pretendemos predecir el rendimiento académico de los estudiantes utilizando el dataset EdNet. El enfoque del trabajo estará dividido en dos partes.

Por un lado, emplearemos modelos de predicción clásicos de *Machine Learning* para predecir los aciertos en las preguntas respondidas por los estudiantes. Este enfoque permitirá explorar la precisión de modelos ampliamente utilizados en problemas de clasificación y regresión en tareas educativas.

Por otro lado, introduciremos el concepto de *Knowledge Tracing (KT)*, que se centra en modelar y evaluar el conocimiento de los estudiantes a lo largo del tiempo. Este enfoque se basa en la premisa de que el conocimiento de un estudiante no es estático, sino que cambia con el tiempo a medida que adquiere nuevas habilidades y conocimientos. Para ello veremos algunos ejemplos de modelos que utilizan técnicas de *Deep Learning* como son las redes neuronales recurrentes (**RNN**) o las redes **LSTM**.

Este trabajo incluye la creación de un nuevo conjunto de datos a partir de varios datasets de EdNet, la exploración de diversas técnicas de modelado y la comparación de los enfoques clásicos y modernos para predecir el rendimiento académico.

Descripción del Conjunto de Datos EdNet

Para este TFM vamos a utilizar el dataset EdNet. Debido a las restricciones de licencia del dataset, Creative Commons Attribution-NonCommercial 4.0 International license, no podemos utilizarlo para usos comerciales, lo cual no presenta problemas con la finalidad del trabajo. Además, debido al gran tamaño del dataset, no podemos adjuntarlo en la entrega del TFM. El dataset original está disponible en el GitHub de EdNet para poder replicar completamente el trabajo. Sin embargo, adjuntamos con la entrega el nuevo dataset creado a partir del conjunto de datos original. Este dataset se encuentra en la carpeta **data** y es el archivo llamado **ednet_dataset.csv**. La creación de este dataset la detallamos en el apartado siguiente.

EdNet es un conjunto de datos masivo que recopila interacciones entre estudiantes y el sistema Santa, una plataforma de tutoría con inteligencia artificial utilizada por más de 780,000 usuarios en Corea. Este dataset fue recopilado durante 2 años y se organiza en cuatro niveles jerárquicos (**KT1**, **KT2**, **KT3** y **KT4**), cada uno con características específicas.

Características principales:

1. Escala masiva:
   - Contiene 131,441,538 interacciones de 784,309 estudiantes.
   - Incluye 13,169 problemas y 1,021 conferencias etiquetadas con 293 habilidades diferentes.
   - Cada estudiante genera, en promedio, 441.2 interacciones.
2. Diversidad:
   - EdNet incluye múltiples actividades de aprendizaje como la lectura de explicaciones, ver conferencias y resolver problemas.
   - Esto permite analizar a los estudiantes desde distintas perspectivas, como su compromiso o el impacto de actividades específicas en su aprendizaje.
3. Jerarquía (Los datos están organizados en cuatro niveles principales y uno complementario):
   - **KT1**: Registros básicos de resolución de problemas.
   - **KT2**: Secuencias de acciones durante las sesiones de estudio.
   - **KT3**: Actividades adicionales como lectura de explicaciones y visualización de conferencias.
   - **KT4**: Registros completos de acciones, incluyendo eliminación de opciones, reproducción de videos, y pagos.
   - **Contents**: recopila los datos de cinco tipos distintos, *questions, lectures, payment items, coupons,* y *scores*.

Preprocesamiento y creación del nuevo dataset

Para este trabajo vamos a utilizar los datos de *EdNet-KT1*, *EdNet-KT3* y *questions*. La mayoría de los datos necesarios se encuentran en el dataset *KT3*, que contiene los datos relativos a las interacciones de los usuarios en la plataforma, incluyendo respuestas a preguntas, visionado de lecciones y visionado de explicaciones de respuestas. Los datos que faltan en este dataset son:

- El tiempo que los estudiantes gastan en cada pregunta (el dato está en *KT1*)
- La respuesta correcta de cada pregunta (el dato está en *questions*)

El dataset *KT3* tiene más de 1 millón 300 mil registros. Debido al alto número de registros, vamos a quedarnos con una muestra aleatoria de 100 estudiantes, lo que correspondería a 30 mil registros.

Ya que tenemos los datos repartidos en tres conjuntos de datos distintos, vamos a unirlos teniendo en cuenta el *user_id* y la interacción de cada respuesta. Para esta union necesitaremos mantener los datos de las explicaciones y lecciones que solo se encuentran en el dataset *KT3*.

Cabe destacar el formato que comparten los cuatro datasets principales (*KT1, KT2, KT3 y KT4*):

- Cada dataset está compuesto por miles de CSV
- Cada uno de estos CSV tiene como nombre el patrón uXXXXX.csv, donde el nombre representa el id del usuario, es decir, el id del estudiante.

Ahora que tenemos los tres conjuntos de datos unidos y hemos tomado una muestra aleatoria del dataset original, vamos a crear 7 nuevas columnas que serán útiles para el EDA y el entrenamiento de los modelos. Estas nuevas columnas son las siguientes (las detallaremos en el siguiente apartado):

- **bundle_had_explanation**
- **prior_question_elapsed_time**
- **cumulative_correct_answers**
- **recent_accuracy**
- **cumulative_explanations_seen**
- **cumulative_lectures_seen**
- **cumulative_responses_by_bundle**

De esta forma, ya hemos creado el nuevo dataset que utilizaremos en el resto del trabajo.

Análisis exploratorio de los datos (EDA)

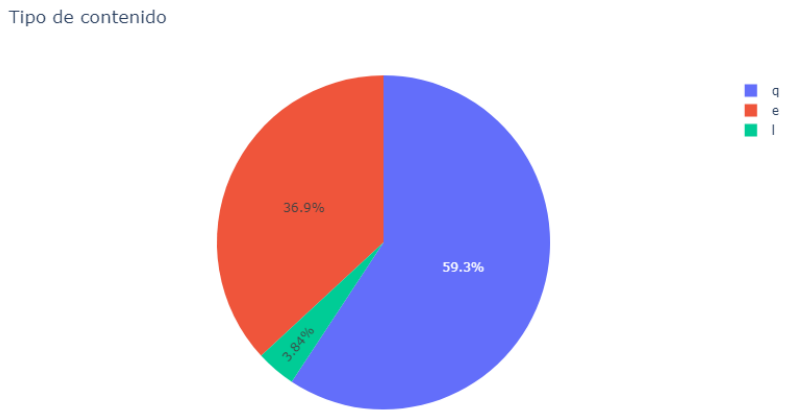
Podemos describir las columnas del dataset de la siguiente forma:

- **`timestamp`** (*int*): El tiempo en milisegundos en el que el usuario realizó una acción.
- **`item_id`** (*object*): Código de identificación para el ítem (preguntas, lecciones o explicaciones).
- **`user_answer`** (*string*): La respuesta del usuario a la pregunta, si existe. El valor -1 se corresponde a las lecciones y explicaciones, debe leerse como un null.
- **`user_id`** (*object*): Código de identificación para el usuario.
- **`content_type`** (*string*): Un código de letra para el tipo de contenido (q=preguntas, `l`=lecciones, e=explicaciones).
- **`bundle_id`** (*object*): Código de identificación para el lote de preguntas, lecciones o explicaciones. (Por ejemplo, tres cuestiones, una explicación y una lección pueden pertenecer al mismo `bundle_id`).
- **`answered_correctly`** (*int*): Indica si el usuario respondió correctamente a la pregunta, si existe. El valor 1 corresponde a una respuesta correcta y el 0 a una incorrecta. El valor es null para lecciones y explicaciones.
- **`content_id`** (*object*): Código de identificación para la interacción del usuario con el ítem.
- **`elapsed_time`** (*float*): El tiempo en milisegundos que tardó un usuario en responder una pregunta. El valor es null para lecciones y explicaciones.
- **`bundle_had_explanation`** (*bool*): Indica si el usuario vio una explicación después de responder al lote de preguntas anterior (`bundle_id`), ignorando cualquier lección entre medio. El valor se comparte a través del mismo lote de preguntas y es nulo para el primer lote de preguntas de un usuario. Generalmente, las primeras preguntas que ve un usuario forman parte de una prueba de diagnóstico inicial en la que no recibieron retroalimentación.
- **`prior_question_elapsed_time`** (*float*): El tiempo promedio en milisegundos que tardó un usuario en responder cada pregunta en el lote de preguntas anterior, ignorando cualquier lección entre medio (es nulo para el primer lote de preguntas).
- **`cumulative_correct_answers`** (*float*): El número acumulado de respuestas correctas que ha tenido el estudiante hasta ese momento. Aporta un indicador directo de cuánto ha acertado el estudiante hasta esa interacción, lo que puede ayudar a predecir su desempeño futuro.
- **`recent_accuracy`** (*float*): La precisión reciente del estudiante (en las últimas 5 preguntas), calculada como el porcentaje de respuestas correctas en un intervalo corto de tiempo. Este indicador puede ser útil para medir el

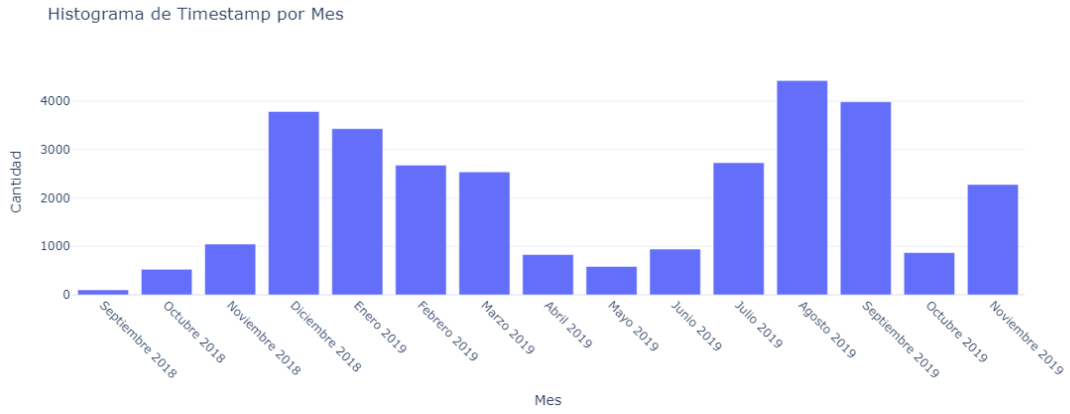
"momentum" del estudiante, es decir, si está mejorando o empeorando en su desempeño.

- **`cumulative_explanations_seen`** (*float*): El número acumulado de explicaciones que ha visto el estudiante hasta ese momento. Este valor puede correlacionarse con una mejor comprensión de los contenidos si el estudiante tiende a consultar explicaciones con frecuencia.
- **`cumulative_lectures_seen`** (*float*): El número acumulado de lecciones que ha visto el estudiante hasta ese momento. Este valor puede correlacionarse con una mejor comprensión de los contenidos si el estudiante tiende a consultar lecciones con frecuencia.
- **`cumulative_responses_by_bundle`** (*float*): El número acumulado de preguntas respondidas por cada lote para cada estudiante.

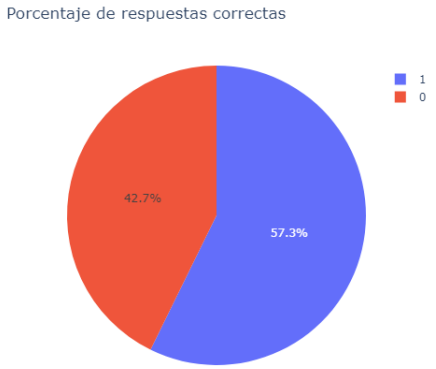
El dataset contiene 30771 filas y 16 columnas. De estas 30771 filas, 18243 filas corresponden a respuestas a preguntas por parte de los alumnos. Estas últimas son las que utilizaremos para el entrenamiento de los modelos. En el siguiente gráfico podemos ver el porcentaje del tipo de contenido del dataset, donde “q” se refiere a cuestiones, “e” a explicaciones y “l” a lecciones.



La distribución de interacciones de usuarios parece seguir un patrón estacional con picos en los primeros meses del año y verano, lo cual tiene sentido teniendo en cuenta que estamos hablando de datos relacionados con educación.



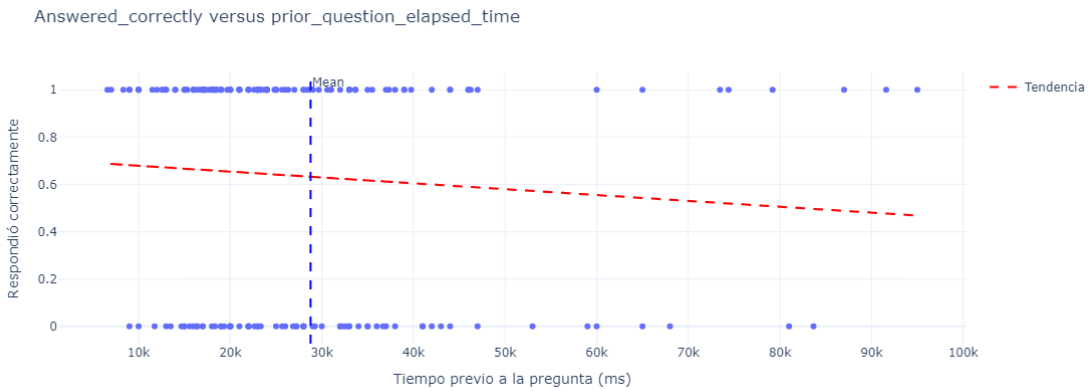
En cuanto a la distribución de respuestas correctas, observamos una distribución bastante pareja entre las respuestas correctas e incorrectas.



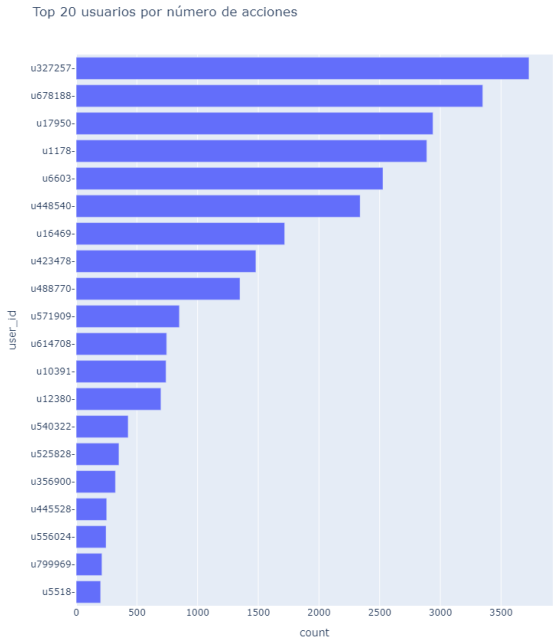
De igual forma observamos una distribución bastante igualada entre las distintas respuestas de los usuarios.

Los datos nos muestran que los estudiantes que respondieron incorrectamente tardaron más tiempo en responder cada pregunta en el lote de preguntas anterior (30551 ms) que aquellos que respondieron correctamente (27452 ms). Además, el gráfico refuerza esta relación con una leve tendencia negativa: a mayor tiempo previo, menor probabilidad de acierto. Esto sugiere que tiempos largos pueden indicar dificultad, afectando el rendimiento en la siguiente pregunta.

	answered_correctly	prior_question_elapsed_time
	count	mean
answered_correctly		
0	7794	30551.134766
1	10449	27452.048828

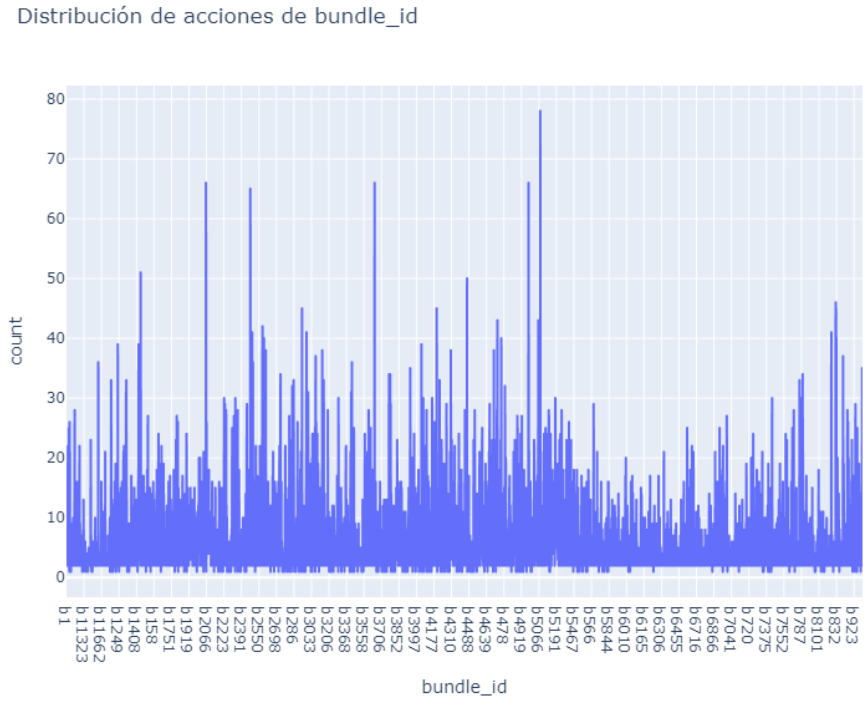


En cuanto a los usuarios, podemos ver en el siguiente gráfico los 20 usuarios con mayor número de acciones. Hay que tener en cuenta que esto incluye tanto las respuestas contestadas como las lecciones y explicaciones vistas.

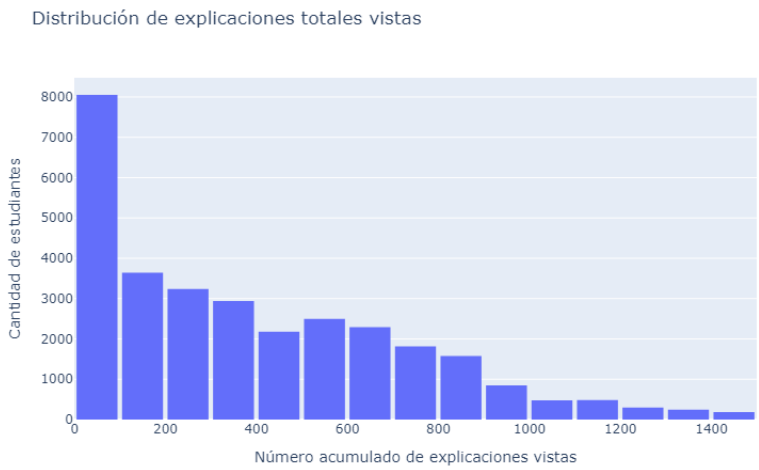


Podemos calcular el número más alto de preguntas respondidas por un usuario que es 2109, lo que corresponde a un 11,5% del total de respuestas.

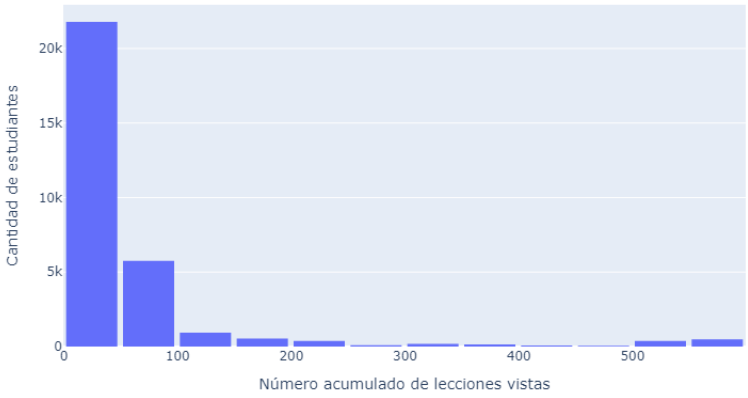
A continuación, podemos analizar los datos que relacionados con la columna de *bundle_id* que, recordemos, se refiere al lote de preguntas, lecciones o explicaciones. Tenemos 4959 lotes únicos de cuestiones o lecciones. Además, tenemos 4692 *bundle_id* con lecciones y 267 *bundle_id* sin lecciones. El mayor número de lecciones vistas dentro de un único *bundle_id* es 29. El siguiente gráfico muestra la distribución de acciones de *bundle_id*.



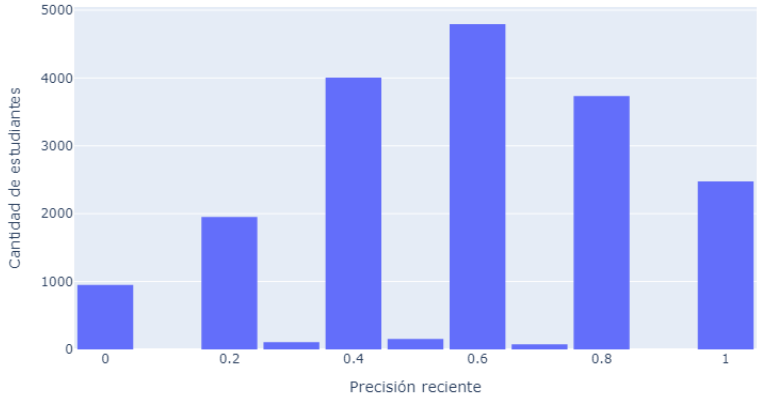
Por último, vamos a ver la distribución de datos de las explicaciones y lecciones totales vistas y de la precisión reciente (*recent_accuracy*):



Distribución de lecciones totales vistas



Distribución de recent_accuracy



Enfoque Basado en Modelos de Machine Learning

En primer lugar, vamos a utilizar varios modelos de *Machine Learning* para predecir el rendimiento académico de los estudiantes. Los modelos seleccionados (**Decision Tree**, **Random Forest**, **XGBoost**, **LightGBM** y **Stacking**) representan una evolución en la complejidad de los algoritmos de aprendizaje supervisado: desde modelos más sencillos e interpretables hasta modelos más avanzados basados en técnicas de boosting y ensamblado, que suelen lograr mejor rendimiento en tareas de clasificación.

Antes del modelado, realizaremos unas transformaciones al dataset para prepararlo para el entrenamiento de los modelos.

Además de corregir los pocos datos faltantes en el dataset, utilizaremos la función **create_feature_df** (se encuentra en el script **data_mining.py** de la carpeta **utils**) que crea un DataFrame con varias estadísticas agregadas a nivel de usuarios contenido y conjunto de preguntas. Este proceso calcula estadísticas como la media, el conteo, la desviación estándar, la mediana y el sesgo de las respuestas correctas para cada usuario, contenido y conjunto de preguntas. Luego, estos valores se agregan al DataFrame original. Además de manejar los datos faltantes, también se normalizan los datos mediante la función **scale_data** del script **data_mining.py** de la carpeta **utils**. Hay que tener en cuenta que los datos que se utlizan en el entrenamiento, tanto en *Machine Learning* como en *Deep Learning*, son los relativos a las preguntas, el resto de datos ya se han utilizado para el EDA y para el cálculo de las nuevas columnas.

A continuación veremos en detalle cada modelo de *Machine Learning* utilizado, así como su resultado (en los datos de test) y sus bondades y debilidades. Todas las funciones relacionadas con los modelos de *Machine Learning* se encuentran en el script **ml.py** de la carpeta **utils**.

El **Decision Tree Classifier** nos ha servido como una primera aproximación por su simplicidad y facilidad de interpretación. Este modelo permite visualizar de manera clara cómo se toman las decisiones en función de las características del conjunto de datos. Sin embargo, su desempeño ha sido algo inferior en comparación con otros modelos más avanzados, con un AUC de 0.91117 y una precisión del 78% para la clase negativa y 85% para la positiva. A pesar de ser fácil de explicar, presenta limitaciones al

ser propenso al sobreajuste y no captar bien relaciones complejas en los datos.

Por otro lado, el **Random Forest** mejora la robustez al utilizar múltiples árboles de decisión en conjunto, lo que reduce la varianza y mejora la generalización. En este caso, ha logrado un AUC de 0.91299 y una precisión del 79% para la clase negativa y 83% para la positiva. Aunque mejora respecto al árbol de decisión individual, sigue teniendo la desventaja de ser computacionalmente más costoso y menos interpretable debido a la naturaleza de su ensamblado de múltiples árboles.

El modelo **XGBoost** ha mostrado un mejor rendimiento con un AUC de 0.92085, logrando una precisión del 80% para la clase negativa y 83% para la positiva. Su éxito radica en el uso de boosting, que optimiza los errores de predicciones previas y mejora la capacidad del modelo para identificar patrones. Además, es eficiente en términos computacionales y maneja bien valores faltantes y ruido en los datos. Sin embargo, su principal debilidad es que requiere un ajuste de hiperparámetros cuidadoso para evitar el sobreajuste y maximizar su rendimiento.

De forma similar, el modelo **LightGBM** ha obtenido el mejor AUC individual con 0.92134, destacándose también en precisión y recall. LightGBM es una alternativa optimizada a XGBoost, diseñada para ser más rápida y eficiente en términos de memoria. Su arquitectura permite una mayor velocidad de entrenamiento, especialmente en datasets grandes. En este caso, ha ofrecido un rendimiento similar a XGBoost pero con una mejor precisión para la clase positiva (85%), lo que indica que tiene una mejor capacidad para identificar correctamente a los estudiantes con buen desempeño. No obstante, al ser un modelo más complejo, su interpretabilidad es menor en comparación con modelos más simples como los árboles de decisión.

Finalmente, se ha implementado un modelo de **stacking** que combina los anteriores con el objetivo de aprovechar sus fortalezas y mitigar sus debilidades. Esta técnica busca mejorar la capacidad predictiva al integrar los outputs de múltiples modelos en una segunda capa de aprendizaje. En este caso, el modelo de stacking ha alcanzado un AUC de 0.92143, prácticamente igual al de LightGBM, pero con una ligera mejora en la clasificación de la clase positiva. Su principal ventaja es la capacidad de combinar modelos complementarios para mejorar la robustez y precisión. Sin embargo, su desventaja radica en la complejidad computacional y la dificultad de interpretación, ya que los resultados dependen de múltiples modelos y sus interacciones.

La elección de estos modelos ha permitido analizar diferentes estrategias para la predicción del rendimiento académico de los estudiantes, desde enfoques más interpretables hasta técnicas más avanzadas con un mejor rendimiento predictivo.

Cabe destacar que para seleccionar el mejor modelo en cada técnica de *Machine Learning*, hemos utilizado **RandomizedSearchCV** para realizar una búsqueda optimizada de hiperparámetros. Posteriormente, hemos comparado los cinco mejores modelos obtenidos en términos de AUC. Además del AUC, se han considerado otros criterios para la selección final, como la desviación estándar del AUC y métricas clave como son **accuracy, precision, recall** y **F1-score**, asegurando así un equilibrio entre rendimiento y estabilidad del modelo.

Introducción al Knowledge Tracing con Deep Learning

El ***Knowledge Tracing*** es la tarea de modelizar los conocimientos de los alumnos a lo largo del tiempo para poder predecir con exactitud su rendimiento en futuras interacciones. Mejorar en esta tarea significa que se pueden sugerir recursos a los alumnos en función de sus necesidades individuales, y que se pueden omitir o retrasar los contenidos que se prevean demasiado fáciles o difíciles. Existen modelos avanzados como **Deep Knowledge Tracing (DKT)** o **Self-Attentive Knowledge Tracing (SAINT+)**, que han demostrado un alto rendimiento en este campo, pero su implementación suele ser compleja debido a la necesidad de arquitecturas sofisticadas basadas en redes recurrentes o transformadores. El lector puede consultar más información sobre estos dos modelos en la bibliografía del informe.

En este trabajo, ante la dificultad de implementar modelos específicos de *Knowledge Tracing*, hemos explicado diversas arquitecturas de *Deep Learning* con el propósito de capturar patrones en los datos de los estudiantes. Hemos evaluado modelos de regresión logística con redes neuronales densas (fully connected), así como arquitecturas más avanzadas como LSTM y CNN, las cuales pueden ser útiles para modelar secuencias temporales y detectar patrones en los datos educativos.

El modelo de **Logistic Regression con Keras** sirvió como una base inicial para la clasificación, utilizando capas densas con técnicas de regularización como Dropout, Batch Normalization y Gaussian Noise para mejorar la generalización. Su desempeño en términos de AUC (0.9195) sugiere que logra capturar relaciones relevantes entre las variables. Posteriormente, utilizamos **Keras Tuner** para optimizar los hiperparámetros, logrando una ligera mejora en el AUC (0.9197), lo que indica que la estructura del modelo base ya era bastante sólida.

Por otro lado, utilizamos redes **LSTM** debido a su capacidad para modelar dependencias temporales en secuencias de datos. En el contexto de *Knowledge Tracing*, estas redes son especialmente útiles porque pueden capturar la evolución del conocimiento de los estudiantes a lo largo del tiempo. A pesar de que el modelo LSTM implementado no alcanzó un AUC superior a los modelos anteriores (0.9184), su potencial en el seguimiento del conocimiento es considerable si se ajustan mejor los hiperparámetros o se entrena con una arquitectura más profunda.

Finalmente, probamos una arquitectura **CNN**, que aunque no está tradicionalmente relacionada con *Knowledge Tracing*, puede ser útil en la identificación de patrones en los datos educativos. En este caso, la CNN logró el mejor AUC (0.9200) entre los modelos evaluados de *Deep Learning*, lo que sugiere que puede estar capturando estructuras relevantes en los datos sin necesidad de modelar explícitamente la secuencialidad.

Si bien los modelos utilizados no son directamente comparables con enfoques avanzados como DKT o SAINT+, su desempeño muestra que técnicas de *Deep Learning* pueden ser muy efectivas para predecir el rendimiento académico. Con mayor refinamiento en la modelización de la secuencialidad de las interacciones del estudiante, podríamos lograr un sistema más robusto de *Knowledge Tracing*, acercándonos a los modelos de vanguardia en este campo.

Por último, cabe mencionar que para los modelos de *Deep Learning* entrenados en este apartado hemos seguido un proceso de transformación de datos similar al realizado en el apartado anterior, teniendo en cuenta que ha sido necesario un reshape de los datos para los modelos de LSTM y CNN. La arquitectura con la que hemos entrenado cada uno de estos modelos se encuentra en el script de python **dl.py** de la carpeta **utils**.

Comparación y Discusión de Resultados

En este trabajo hemos evaluado distintos enfoques para la predicción del rendimiento académico de los estudiantes, desde modelos más tradicionales de *Machine Learning* hasta técnicas de *Deep Learning*. Cada uno de estos enfoques presenta sus ventajas y desventajas en términos de precisión, capacidad de generalización e interpretabilidad, factores que son claves en el ámbito académico donde no solo importa la predicción, sino también la capacidad de explicar y justificar los resultados.

Por un lado, si analizamos los modelos basandonos en su resultado en la métrica AUC del conjunto de datos de test, encontramos resultados muy parejos. A continuación, podemos ver una tabla con el resumen de los resultados obtenidos en cada modelo:

Modelo	Test AUC
Decision Tree	0.91117
Random Forest	0.91299
XGBoost	0.92085
LightGBM	0.92134
Stacking	0.92143
Keras Logistic Regression	0.91952
Keras Tuner Logistic Regression	0.91975
LSTM	0.91844
CNN	0.92007

El modelo de Stacking ha obtenido el mejor AUC (0.92143), aunque la diferencia con LightGBM (0.92134) y XGBoost (0.92085) es mínima. En el caso de los modelos de *Deep Learning*, la CNN ha sido la mejor con un AUC de 0.92007, superando al modelo basado en LSTM (0.91844) y a la Regresión Logística en redes neuronales (0.9195).

Estos resultados indican que los modelos basados en boosting, como LightGBM y XGBoost, han demostrado ser altamente competitivos frente a las redes neuronales profundas, con la ventaja adicional de ser más eficientes computacionalmente y menos exigentes en términos de datos de entrenamiento.

Por otro lado, la capacidad de interpretación de un modelo es un aspecto clave para la selección del modelo óptimo. En el ámbito educativo, como ya hemos comentado, este aspecto es fundamental para comprender por qué un estudiante puede estar teniendo dificultades o qué factores influyen en su desempeño.

En primer lugar, tenemos los modelos de *Machine Learning* que son aquellos que ofrecen una mejor interpretabilidad. El Decision Tree es el más explicativo de todos, ya que es un árbol de decision fácil de interpretar, sin embargo, es el que ofrece un peor rendimiento. Los modelos de Random Forest, XGBoost y LightGBM permiten analizar la importancia de las variables, aunque con mayor complejidad en la interpretación de las predicciones. Finalmente, el Stacking presenta claras dificultades en cuanto a la interpretación de las predicciones, ya que combina varios modelos y sus interacciones pueden ser difíciles de descomponer.

De otro lado, encontramos los modelos de *Deep Learning*, con una dificultad mucho mayor de interpretación que los modelos anteriores. La Regresión Logística con redes neuronales es más explicativa que los modelos de LSTM y CNN. El modelo LSTM, aunque está diseñado para capturar relaciones temporales en los datos de los estudiantes, es, en esencia, una "caja negra" difícil de interpretar. Por último, el modelo CNN ha demostrado tener el mejor rendimiento entre los modelos de *Deep Learning*, pero es el modelo menos explicable de todos, ya que su arquitectura basada en convoluciones detecta patrones en los datos sin ofrecer una representación clara de cómo cada variable individual contribuye a la predicción.

Por tanto, habiendo visto los resultados de cada modelo, así como su interpretabilidad, vamos a elegir como modelo ganador para este trabajo el modelo LightGBM por los siguientes motivos:

- Este modelo es el segundo en términos de AUC, muy cerca del primero (Stacking).
- Ofrece una mayor interpretabilidad que los modelos de *Deep Learning* y Stacking, solo por detrás del modelo de Decision Tree en términos de explicatividad
- Tanto el modelo XGBoost como el LightGBM son modelos basados en boosting basados en árboles de decisión y permiten analizar la importancia de las características mediante métodos como *feature importance* y *SHAP values*. Ambos tienen por tanto una interpetavilidad muy similar y unos resultados practicamente idénticos. Sin embargo, el modelo LightGBM presenta algunas ventajas frente al XGBoost, como podrían ser: mayor velocidad de entrenamiento, menor consumo de memoria, mejor rendimiento en datasets con muchas características y escalabilidad. Este último punto es especialmente importante teniendo en cuenta que el conjunto de datos de entrenamiento es solamente una pequeña muestra del dataset original.

Conclusiones

En este trabajo, hemos analizado diversas técnicas de *Machine Learning* y *Deep Learning* para predecir el rendimiento académico de estudiantes. Tras examinar diversos modelos con relación a su precisión e interpretación, hemos seleccionado LightGBM como el modelo más adecuado para nosotros. Su elección se basa en su alta precisión, capacidad de interpretación de los resultados y eficacia computacional, lo que lo hace escalable para grandes datasets.

Aunque se han logrado excelentes resultados, los modelos de *Knowledge Tracing* más avanzados, como DKT o SAINT+, han demostrado en la literatura un desempeño superior en problemas similares. Su implementación en este estudio no pudo realizarse debido a la complejidad técnica y el alto costo computacional que implican, sin embargo, representan una dirección prometedora para futuros estudios.

Además, es fundamental destacar que, debido a limitaciones tecnológicas, hemos realizado este trabajo con una muestra limitada del conjunto de datos EdNet, lo que podría haber influido en el desempeño de los modelos, particularmente en los de *Deep Learning*. Estos modelos habitualmente se benefician de grandes volúmenes de datos, ya que necesitan muchos registros para capturar de forma eficiente patrones complejos. Por tanto, si se utiliza el conjunto de datos completo, es probable que las redes neuronales (en especial LSTM) incrementen su desempeño, ya que podrían interpretar de manera más eficaz la evolución del conocimiento del estudiante a lo largo del tiempo.

En futuros estudios, resultaría fascinante explorar estos modelos avanzados de *Knowledge Tracing* y valorar cómo cambia el desempeño de los modelos de *Deep Learning* al emplear el conjunto de datos completo, contrastándolos de nuevo con LightGBM en un entorno con mayor capacidad computacional.

Bibliografía

1. Piech, C., Bassen, J., Huang, J., Ganguli, S., Sahami, M., Guibas, L. J., & Sohl-Dickstein, J. (2015). *Deep Knowledge Tracing*. En *Advances in Neural Information Processing Systems* (Vol. 28). Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/bac9162b47c56fc8a4d2a519803d51b3-Paper.pdf
2. Riiid. (2020). *EdNet: A large-scale dataset for knowledge tracing and educational applications*. GitHub. https://github.com/riiid/ednet
3. Shin, D., Shim, Y., Yu, H., Lee, S., Kim, B., & Choi, Y. (2021). *SAINT+: Integrating temporal features for EdNet correctness prediction*. En *LAK21: 11th International Learning Analytics and Knowledge Conference* (pp. 490–496). Association for Computing Machinery. https://doi.org/10.1145/3448139.3448188