UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

INSTITUTO DE INVESTIGACIONES EN MATEMÁTICAS APLICADAS
Y SISTEMAS (IIMAS)

Predicción de Respuestas Correctas de Estudiantes en Exámenes Utilizando XG Boost

MATERIA

Datos Masivos

INTEGRANTES

Luis Enrique Villalon Pineda

Imanol Mendoza Saenz de Buruaga

Erick José Fabián Sandoval

PROFESOR

Lic. David Emmanuel Maqueda Bojorquez

AYUDANTE

Lic. Aldo Muñoz Zecua

Ciudad Universitaria, Cd. Mx., 2025

Índice

Introducción

En los últimos años, el aprendizaje automático ha demostrado ser una herramienta valiosa en el ámbito educativo, permitiendo el análisis predictivo del rendimiento académico mediante modelos basados en datos históricos. Entre las técnicas más efectivas se encuentra *XGBoost*, un algoritmo que combina múltiples árboles de decisión para formar un modelo robusto y altamente preciso. Su capacidad para manejar relaciones no lineales, variables heterogéneas y datos con ruido lo convierte en una opción ideal para problemas de clasificación y regresión en contextos educativos.

La aplicación del machine learning en el ámbito educativo ha revolucionado la forma en que se analiza y mejora el aprendizaje. A diferencia de los métodos tradicionales de evaluación, que suelen basarse en promedios generales o exámenes estandarizados, los modelos predictivos permiten un análisis granular y personalizado del desempeño estudiantil. Estos sistemas son capaces de identificar patrones ocultos en grandes volúmenes de datos, como tendencias de errores comunes, ritmos de aprendizaje y áreas de dificultad específicas.

El presente trabajo tiene por objetivo predecir la probabilidad de que un estudiante responda correctamente a preguntas de un examen, basándose en un conjunto de datos con características históricas tanto del alumno como de las preguntas. Para ello, se desarrolla un sistema de predicción del rendimiento académico utilizando aprendizaje automático, específicamente el algoritmo *XGBoost*.

Estas variables alimentan el modelo de *XGBoost*, que, mediante un proceso de optimización iterativa, aprende a ponderar su importancia para predecir la probabilidad de éxito en nuevas preguntas. La evaluación del modelo se realiza mediante el área bajo la curva *ROC (AUC)*, una métrica ampliamente utilizada en problemas de clasificación binaria, utilizado para evaluar una alta capacidad discriminativa.

¿Por que XGBoost sobre otros?

- Los datos son tabulares, ricos en features numéricas y categóricas.
- Necesitamos modelar una secuencia explícita, sino predecir una probabilidad por pregunta.
- El modelo es rápido de entrenar y puede ser interpretado, lo cual es importante en contextos educativos.

¿Cómo aprende XGBoost?

XGBoost entrena un conjunto de árboles de decisión (gradient boosted trees) de forma secuencial. Cada árbol nuevo trata de corregir el error de los anteriores.

En los siguientes pasos:

1. Primer árbol: Hace una predicción inicial (e.g. todos los estudiantes tienen 50% de probabilidad).
2. Calcula el error (con `logloss` o `auc`).
3. Segundo árbol: Aprende a predecir qué errores cometió el primero, y así sucesivamente.
4. Cada árbol aprende en función de los residuos del anterior.
5. Al final, se suman todos los árboles ponderados para obtener una probabilidad final de acierto.

El proceso comienza con la preparación de los datos, donde se integra información sobre el desempeño previo de los estudiantes con metadatos de las preguntas. Se realiza una ingeniería de características cuidadosa, creando variables clave como la precisión histórica del usuario, el número total de respuestas previas, la dificultad inherente de cada pregunta (basada en el rendimiento de otros

estudiantes), el tiempo promedio que tarda el estudiante en responder, y su desempeño específico en diferentes secciones temáticas del examen.

En este estudio, se emplea un enfoque basado en datos reales de interacciones educativas para construir un modelo predictivo que analiza múltiples dimensiones del desempeño estudiantil. A lo largo del trabajo, se detallará primero la metodología empleada, desde la selección y preprocesamiento de los datos hasta el diseño de características pedagógicamente significativas. Luego, se formalizará el problema de predicción y sus objetivos, seguido de un análisis exploratorio que revelará patrones clave en los datos.

Finalmente, se presentarán los resultados del modelo XGBoost, evaluando no solo su precisión métrica *(AUC)* sino también su capacidad para generar insights accionables, como la identificación de secciones temáticas críticas donde los estudiantes muestran mayores dificultades. Este flujo metodológico desde los fundamentos teóricos hasta la implementación práctica busca demostrar cómo la inteligencia artificial puede transformar la evaluación educativa en un proceso dinámico, predictivo y personalizado.

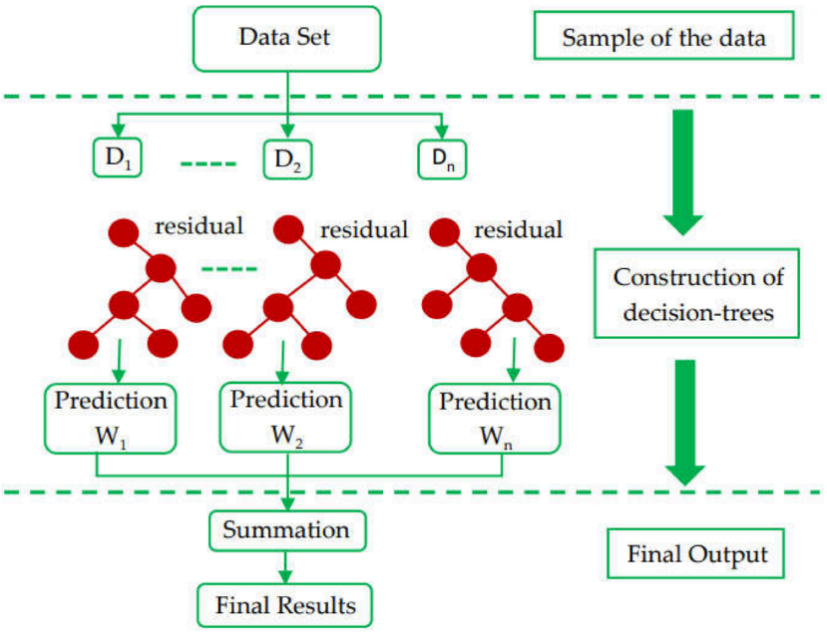


Figura 1. Estructura del Modelo XG Boost

Definición del Problema

La evaluación educativa actual carece de sistemas capaces de predecir con precisión el desempeño individual de los estudiantes, limitando las intervenciones oportunas. Este trabajo se centra en predecir la probabilidad de acierto en preguntas de examen, integrando tanto el conocimiento del alumno como factores contextuales (tiempo de respuesta, dificultad del ítem e historial previo).

El desafío principal reside en la complejidad multivariable y no lineal de los datos educativos, donde las relaciones entre características del estudiante, atributos de las preguntas y contexto de respuesta no siguen patrones simples. Estas interacciones complejas exigen modelos capaces de capturar jerarquías y relaciones condicionales entre variables heterogéneas.

El problema se ve agravado por desafíos técnicos como valores faltantes, datos desbalanceados, entre otros. Su solución beneficiaría a estudiantes con identificación temprana de debilidades, docentes con el diseño de evaluaciones más efectivas e instituciones con la personalización del aprendizaje.

XG Boost emerge como solución óptima por su capacidad para manejar relaciones no lineales, variables mixtas y datos incompletos, superando a los enfoques tradicionales.

Objetivos

Los objetivos específicos de este trabajo son:

1. Implementar un modelo basado en XGBoost que prediga la probabilidad de acierto en preguntas de examen.
2. Incorporar variables clave como historial de desempeño, dificultad de preguntas y tiempos de respuesta.
3. Capturar relaciones no lineales y patrones complejos en los datos educativos.
4. Alcanzar un AUC ≥ 0.85 como métrica de calidad predictiva.

Búsqueda de Información Viable

Para este trabajo hemos utilizado un conjunto de datos propiedad de Riiid Labs, un proveedor de soluciones de IA, Riiid lanzó un tutor de IA basado en algoritmos de aprendizaje profundo en 2017 que atrajo a más de un millón de estudiantes surcoreanos. Este año, la compañía lanzó EdNet, la base de datos abierta más grande del mundo para la educación en IA que contiene más de 100 millones de interacciones entre estudiantes.

Los datos están disponibles para su libre uso en la siguiente dirección:
https://www.kaggle.com/c/riiid-test-answer-prediction/overview

Esta fuente de datos provee hasta 7 archivos distintos que brindan la información de las pruebas (test), información de las preguntas (questions) , clases de los alumnos (lectures), un conjunto de entrenamiento, entre otros. Para cumplir nuestros objetivos únicamente fue necesario utilizar los siguientes dos documentos, su estructura es la siguiente:

train.csv:

row_id: (int64) código de identificación de la fila.
timestamp: (int64) el tiempo en milisegundos entre esta interacción del usuario y la finalización del primer evento de ese usuario.
user_id: (int32) Código de identificación del usuario.
content_id: (int16) Código de identificación para la interacción del usuario
content_type_id: (int8) 0 si el evento era una pregunta que se planteaba al usuario, 1 si el evento era el usuario que estaba viendo una conferencia.
task_container_id:(int16) Código de identificación para el lote de preguntas o conferencias.
user_answer: (int8) la respuesta del usuario a la pregunta, si la hubiera. Lea -1 como nulo, para las conferencias.
answered_correctly: (int8) si el usuario respondió correctamente. Lea -1 como nulo, para las conferencias.
prior_question_elapsed_time:(float32) El tiempo promedio en milisegundos que le tomó a un usuario responder a cada pregunta en el paquete de preguntas anterior, ignorando cualquier clase intermedia.

prior_question_had_explanation:(bool) Si el usuario vio o no una explicación y la(s) respuesta(s) correcta(s) después de responder al paquete de preguntas anterior, ignorando cualquier conferencia intermedia.

questions.csv: metadatos de las preguntas planteadas a los usuarios.

question_id: clave externa para la columna de content_id de entrenamiento/prueba, cuando el tipo de contenido es pregunta (0).
bundle_id: código para el cual las preguntas se entregan juntas.
correct_answer: la respuesta a la pregunta. Se puede comparar con la columna train para comprobar si el usuario tenía razón.user_answer
part: la sección correspondiente del examen TOEIC.
tags: uno o más códigos de etiqueta detallados para la pregunta.

En conjunto, estos son los dos conjuntos de datos que utilizaremos para nuestro proyecto, en secciones posteriores se explicará más sobre el tratamiento, limpieza y selección de columnas relevantes que fue necesario para cada uno.

Metodología

El desarrollo de este sistema predictivo se abordó mediante un proceso estructurado que combina técnicas de ciencia de datos con procesamiento de datos. Partiendo de datos históricos de interacciones educativas, se implementó un pipeline de machine learning que transforma registros crudos en predicciones sobre el desempeño estudiantil. La metodología adoptada sigue el siguiente flujo que garantiza tanto la robustez técnica del modelo como su relevancia educativa.

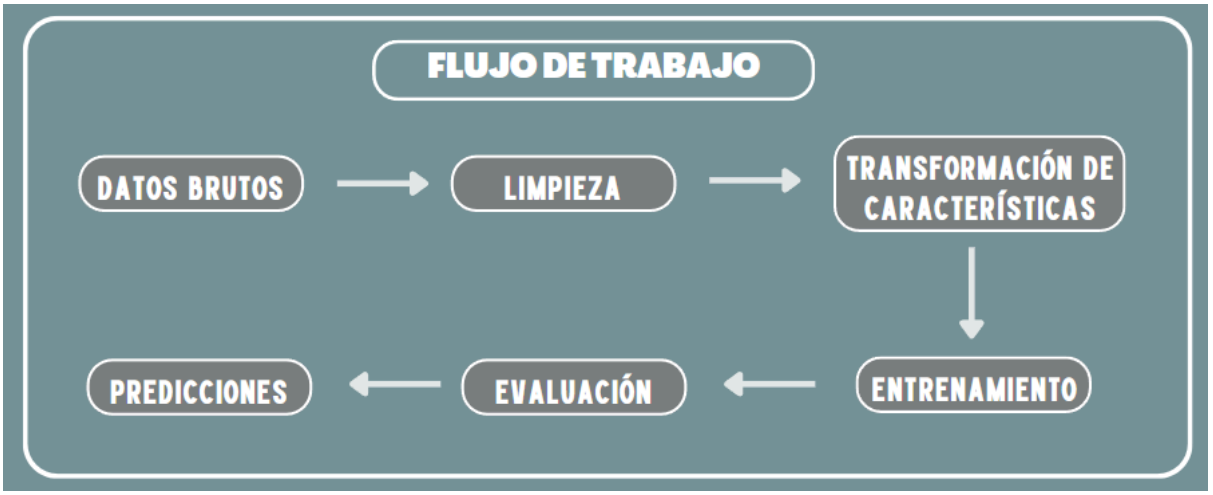


Figura 2. Etapas de nuestro sistema predictivo

La primera etapa consistió en un trabajo de preparación y exploración de datos. Los datasets originales, que incluían desde registros de respuestas individuales hasta metadatos de las preguntas, fueron sometidos a procesos de limpieza y homogeneización. Esto implicó tratar valores faltantes, especialmente en variables temporales donde se imputaron ceros para casos no registrados, y fusionar tablas mediante operaciones relacionales que vinculan cada respuesta con las características correspondientes del estudiante y la pregunta.

En la fase de transformación de características, se crearon variables predictoras que capturan dimensiones clave del proceso de aprendizaje. Estas incluyen métricas individuales (como el porcentaje histórico de aciertos y el tiempo promedio de respuesta), atributos de las preguntas (como su tasa de éxito global y frecuencia de aparición) y elementos contextuales (como el tiempo empleado en la pregunta

anterior o la disponibilidad de explicaciones). Particularmente valiosa resultó la creación de métricas específicas por área temática, que permiten diferenciar las competencias del estudiante en distintos dominios de conocimiento. Estas transformaciones no solo mejoran la capacidad predictiva del modelo, sino que además generan variables interpretables desde una perspectiva pedagógica.

Para el modelado predictivo, se seleccionó XGBoost por su capacidad para manejar relaciones no lineales y su robustez frente a datos heterogéneos. La arquitectura del modelo se optimizó mediante técnicas de validación cruzada, definiendo cuidadosamente hiper parámetros como la profundidad máxima de los árboles (6 niveles) y la tasa de aprendizaje (0.05). El conjunto de datos se dividió estratégicamente, reservando un 20% para validación y manteniendo la distribución original de usuarios en ambos grupos. Durante el entrenamiento, se implementó early stopping para prevenir sobreajuste, interrumpiendo el proceso cuando la métrica AUC dejaba de mejorar durante 20 iteraciones consecutivas.

La evaluación del modelo reveló un excelente desempeño predictivo, con un AUC que indica alta capacidad para discriminar entre respuestas correctas e incorrectas. Un análisis de importancia de características confirmó el peso determinante de variables como el historial de aciertos del estudiante y la dificultad inherente de cada pregunta. Más allá de las métricas técnicas, el sistema demostró utilidad práctica al generar simulaciones completas de rendimiento en exámenes, identificando no solo la probabilidad de éxito global, sino también patrones diferenciales por área temática. Esta capacidad de desagregación resulta particularmente valiosa para diseñar intervenciones educativas personalizadas.

La implementación final integra todas estas componentes en un sistema coherente que transforma datos crudos en insights accionables. Desde la carga inicial de los datasets hasta la generación de predicciones, cada etapa del proceso fue diseñada para garantizar tanto rigor metodológico como relevancia educativa. El resultado es una herramienta que no solo predice resultados académicos con alta precisión, sino que además proporciona un marco analítico para entender los factores que subyacen al desempeño estudiantil. Esta doble capacidad predictiva y explicativa posiciona al modelo como un valioso recurso para la toma de decisiones educativas basadas en evidencia.

Análisis Exploratorio de los Datos

En primer lugar, observamos que variables como prior_question_elapsed_time y timestamp están registradas en milisegundos. Esta unidad de medida, aunque precisa, puede no ser la más adecuada para algunos modelos de machine learning, ya que los valores resultan ser extremadamente grandes. Por ello, se recomienda considerar la conversión de estas variables a una unidad más interpretable y manejable, esto dependerá del contexto del análisis y de la sensibilidad que el modelo implementado presente en estas variables

En las variables prior_question_elapsed_time y prior_question_had_explanation se identificaron registros con valores nulos. Afortunadamente, estos valores faltantes representan un porcentaje muy pequeño del total (menos del 5% para cada variable), lo cual permite tomar la decisión de eliminarlos sin comprometer significativamente la integridad ni la representatividad del conjunto de datos.

Se sugiere conservar las columnas que hacen referencia a identificadores (IDs) de otras tablas, como content_id, user_id, task_container_id, entre otras. Aunque a primera vista podrían parecer variables sin relevancia directa para el modelo, estos campos serán fundamentales para realizar uniones (joins) con otras tablas que contienen información detallada sobre preguntas, usuarios o contenido educativo.

También se detectaron valores atípicos (outliers) en varias de las variables numéricas, particularmente aquellas relacionadas con el tiempo entre preguntas o el tiempo que un usuario tarda en responder. Aunque en otros contextos estos valores extremos podrían considerarse errores o anomalías y se optarían por eliminarlos, en este caso específico es preferible conservarlos. La razón es que dichos valores pueden representar comportamientos reales de los usuarios (como pausas largas por distracción o interrupciones) y eliminar estos datos podría sesgar los resultados del análisis.

Resultados

A continuación presentamos una descripción paso a paso de nuestro proceso de trabajo así como los resultados obtenidos.

Lenguaje de Programación

Se utilizó el lenguaje de programación Python como herramienta a lo largo de todo este trabajo. A través de la importación de las siguientes librerías:

```
import pandas as pd
import numpy as np
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
import matplotlib.pyplot as plt
import seaborn as sns
```

Carga de Datos

Como mencionamos, los dos conjuntos de datos que empleamos son los siguientes, se almacenan como una variable *dataframe*, estructura propia de la librería pandas.

```
train = pd.read_csv("train_muestra_ct.csv")
questions = pd.read_csv("questions.csv")
```

Procesamiento y Transformación de Características

Las transformaciones realizadas son las siguientes, adicionalmente se añade el código empleado en cada punto:

1. Unión de Metadatos de Preguntas: Integrar información sobre las categorías temáticas de cada pregunta en el dataset principal.

```
questions = questions.rename(columns={"question_id": "content_id"})
train = train.merge(questions[["content_id", "part"]], on="content_id",
how="left")
```

2. Filtrado de Interacciones No Relevantes: Eliminar registros que no corresponden a preguntas.

```
train = train[train['content_type_id'] == 0]
```

3. Tratamiento de Valores Nulos: Garantizar que no haya valores faltantes en variables importantes.

```
train['prior_question_elapsed_time']=train['prior_question_elapsed_time
'].fillna(0)
train['prior_question_had_explanation']=train['prior_question_had_expla
nation'].fillna(False).astype(bool).astype(int)
```

4. Creación de Características por Usuario: Capturar el desempeño histórico y experiencia de cada estudiante.

```
user_stats=  train.groupby('user_id')['answered_correctly'].agg(['mean',
'count']).reset_index()
user_stats.columns = ['user_id', 'user_accuracy', 'user_total_answers']
train = train.merge(user_stats, on='user_id', how='left')
```

5. Creación de Características por Pregunta: Cuantificar la dificultad y frecuencia de cada pregunta.

```
question_stats=train.groupby('content_id')['answered_correctly'].agg(['
mean', 'count']).reset_index()
question_stats.columns=['content_id','question_accuracy','question_atte
mpts']
train = train.merge(question_stats, on='content_id', how='left')
```

6. Tiempo Promedio por Usuario: Medir la velocidad de respuesta característica de cada estudiante.

```
user_time=train.groupby('user_id')['prior_question_elapsed_time'].mean(
).reset_index()
user_time.columns = ['user_id', 'user_avg_time_per_question']
train = train.merge(user_time, on='user_id', how='left')
```

7. Precisión por Parte Temática: Identificar fortalezas/debilidades en áreas específicas.

```
user_part=train.groupby(['user_id','part'])['answered_correctly'].mean(
).reset_index()
user_part.columns = ['user_id', 'part', 'user_part_accuracy']
train = train.merge(user_part, on=['user_id', 'part'], how='left')
```

Selección de Características

Después de realizar el preprocesamiento, las características seleccionadas fueron las siguientes:

'prior_question_elapsed_time', # Tiempo empleado en la pregunta anterior
'prior_question_had_explanation', # Si incluye explicación
'user_accuracy', # Precisión histórica global del usuario
'user_total_answers', # Número total de respuestas del usuario
'question_accuracy', # Tasa de aciertos históricos para esa pregunta
'question_attempts', # Veces que la pregunta ha sido intentada
'user_avg_time_per_question', # Tiempo promedio del usuario por pregunta
'part', # Categoría temática de la pregunta (1-7)
'user_part_accuracy' # Precisión del usuario por 'part'

División Del Conjunto

Se prepararon los datos para entrenamiento y validación, con una división 80/20, 80% para entrenamiento (X_train, y_train), 20% para validación (X_val, y_val). Los vectores quedaron de la siguiente manera:
X: Matriz con las 9 features seleccionadas.
y: Vector con la etiqueta a predecir (*answered_correctly*).
Además se fijó una semilla para reproducibilidad con *random_state=42*.

Entrenamiento del Modelo

En primer lugar fue necesaria la conversión de los DataFrames de pandas al formato DMatrix, optimizado para XGBoost, con la función xgb.DMatrix(X_train, label=y_train).
Posteriormente llego la etapa de entrenar nuestro modelo, el código que especifica los parámetros empleados es el siguiente:

```
params = {
    'objective': 'binary:logistic',
    'eval_metric': 'auc',
    'max_depth': 6,
    'eta': 0.05,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
    'lambda': 1.0,
    'alpha': 0.5,
    'tree_method': 'hist',
    'seed': 42
```

```
}

watchlist = [(dtrain, 'train'), (dval, 'val')]
model = xgb.train(
    params,
    dtrain,
    num_boost_round=1000,
    evals=watchlist,
    early_stopping_rounds=20,
    verbose_eval=50
)
```

Este modelo se caracteriza por ser un modelo de entrenamiento iterativo, en cada iteración (boost_round), se añade un nuevo árbol que corrige los errores residuales del anterior. Después se evalúa el AUC tanto en entrenamiento como en validación para detectar sobreajuste.

Con *Early Stopping* se evita overfitting y ahorra tiempo computacional. Si el AUC en validación no mejora durante 20 rondas consecutivas, el entrenamiento se detiene automáticamente.

La salida obtenida fue la siguiente:

```
[0]     train-auc:0.71422       val-auc:0.71368
[50]    train-auc:0.84510       val-auc:0.84455
[100]   train-auc:0.84661       val-auc:0.84603
[150]   train-auc:0.84724       val-auc:0.84658
[200]   train-auc:0.84757       val-auc:0.84682
[250]   train-auc:0.84780       val-auc:0.84695
[300]   train-auc:0.84800       val-auc:0.84701
[350]   train-auc:0.84817       val-auc:0.84704
[400]   train-auc:0.84834       val-auc:0.84706
[450]   train-auc:0.84852       val-auc:0.84707
[490]   train-auc:0.84864       val-auc:0.84707
```

Evaluación del Modelo

Usamos nuestro modelo con el conjunto de evaluación(test)

```
y_pred = model.predict(dval)
auc_score = roc_auc_score(y_val, y_pred)
print(f"AUC final: {auc_score:.5f}")
```

La salida del AUC final fue:

```
AUC final: 0.84707
```

El Área Bajo la Curva (AUC) evalúa la capacidad del modelo para distinguir entre las dos clases (en este caso respuestas correctas 1 e incorrectas 0). Obtener un AUC final de 0.84707 significa que hay un un 84.77% de probabilidad de distinguir correctamente entre respuestas correctas e incorrectas. Lo cual es una muy buena métrica de nuestro modelo.

Evaluación con Datos existentes:

Probamos con un dato existente dentro de en este caso el dato se nos presenta como lo siguiente dato:

```
 Predicción con un dato real del dataset:
         user_id  content_id  answered_correctly
551650  303669058       10685                   1
```

El modelo mediante XGBoost comparándolo con el dato real nos dice, que efectivamente la siguiente pregunta la va a tener correcta , mientras que XGBoost estima que la probabilidad de responder es de 0.74 lo cual al tratarse de tener o no bien la pregunta y estar por encima de 50% de probabilidad se toma como correcta ; comparamos con lo real y por lo que esta bien predicho.

```
Probabilidad estimada de responder correctamente: 0.7475
Respuesta real: Correcta
```

Evaluación con Nuevos Datos:
Después probamos nuestro modelo con nuevos datos insertados por nosotros, la salida obtenida es como la siguiente:

```
Probabilidad de responder correctamente: 0.5767
```

En ella se puede observar la probabilidad de que un usuario con nuestras características responda correctamente.

Probamos con más usuarios e imprimimos las características que les dimos así como la probabilidad de responder correctamente en la parte inferior.

```
 Ejemplo 1
- Tiempo viendo pregunta anterior: 10.5 segundos
- ¿Vio explicación antes?: Sí
- Precisión histórica del usuario: 0.86
- Total de respuestas del usuario: 159
- Precisión global de la pregunta: 0.82
- Número de intentos sobre esta pregunta: 699
- Tiempo promedio del usuario por pregunta: 34.5 segundos
- Parte temática (1-7): 4
- Precisión del usuario en esta parte: 0.71
 **Probabilidad de responder correctamente:** 0.8858
```

```
 Ejemplo 2
- Tiempo viendo pregunta anterior: 23.2 segundos
- ¿Vio explicación antes?: No
- Precisión histórica del usuario: 0.61
- Total de respuestas del usuario: 151
- Precisión global de la pregunta: 0.66
- Número de intentos sobre esta pregunta: 181
- Tiempo promedio del usuario por pregunta: 16.5 segundos
- Parte temática (1-7): 1
- Precisión del usuario en esta parte: 0.59
 **Probabilidad de responder correctamente:** 0.4334
```

```
 Ejemplo 3
- Tiempo viendo pregunta anterior: 17.5 segundos
- ¿Vio explicación antes?: No
- Precisión histórica del usuario: 0.43
- Total de respuestas del usuario: 210
- Precisión global de la pregunta: 0.47
- Número de intentos sobre esta pregunta: 638
- Tiempo promedio del usuario por pregunta: 24.0 segundos
- Parte temática (1-7): 5
- Precisión del usuario en esta parte: 0.41
 **Probabilidad de responder correctamente:** 0.2341

 Ejemplo 4
- Tiempo viendo pregunta anterior: 39.3 segundos
- ¿Vio explicación antes?: Sí
- Precisión histórica del usuario: 0.47
- Total de respuestas del usuario: 126
- Precisión global de la pregunta: 0.73
- Número de intentos sobre esta pregunta: 211
- Tiempo promedio del usuario por pregunta: 32.6 segundos
- Parte temática (1-7): 4
- Precisión del usuario en esta parte: 0.57
 **Probabilidad de responder correctamente:** 0.5852
```

```
 Ejemplo 5
- Tiempo viendo pregunta anterior: 32.5 segundos
- ¿Vio explicación antes?: Sí
- Precisión histórica del usuario: 0.41
- Total de respuestas del usuario: 155
- Precisión global de la pregunta: 0.78
- Número de intentos sobre esta pregunta: 524
- Tiempo promedio del usuario por pregunta: 24.1 segundos
- Parte temática (1-7): 4
- Precisión del usuario en esta parte: 0.69
 **Probabilidad de responder correctamente:** 0.7565
```

Probabilidad de Responder cada Pregunta Correctamente

Con base en el perfil de un usuario, incluyendo el porcentaje estimado de aciertos, es posible estimar la probabilidad con la que se estima que responda correctamente cada pregunta, ejemplo, suponiendo que tenemos un alumno con estas características:

```
 Examen simulado:
- Total de preguntas: 10
- Aciertos estimados: 7 / 10
Porcentaje estimado de aciertos: 74.86%
```

El estudiante tiene un porcentaje estimado de aciertos del 74.86%. Por ende, se estima que este es el porcentaje de contestar correctamente cada una de las 10 preguntas:

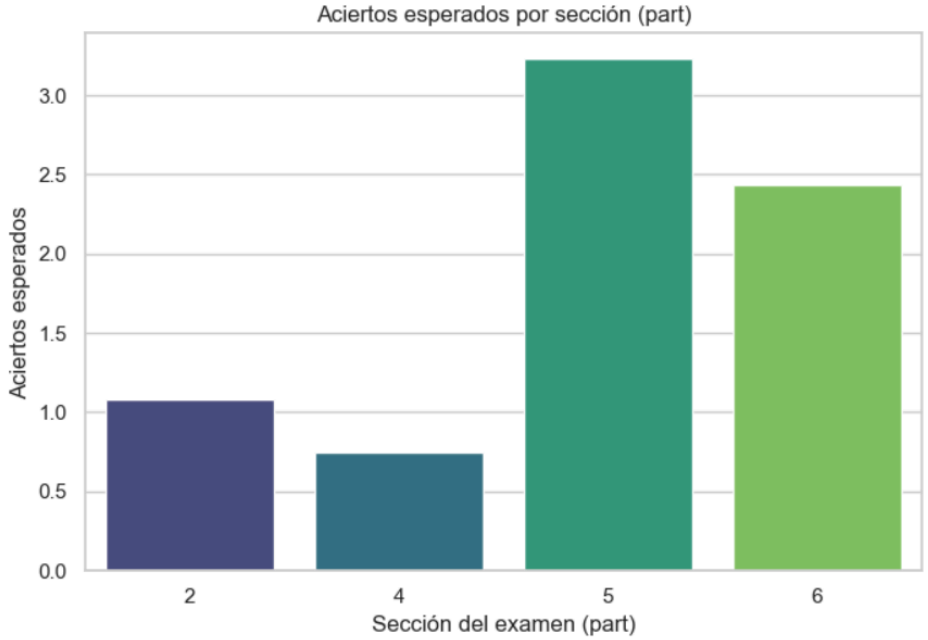
```
Detalle por pregunta:
- Pregunta 1: Part 6 → Probabilidad de acierto: 0.7692
- Pregunta 2: Part 5 → Probabilidad de acierto: 0.8190
- Pregunta 3: Part 6 → Probabilidad de acierto: 0.7829
- Pregunta 4: Part 5 → Probabilidad de acierto: 0.8665
- Pregunta 5: Part 2 → Probabilidad de acierto: 0.4670
- Pregunta 6: Part 2 → Probabilidad de acierto: 0.6132
- Pregunta 7: Part 6 → Probabilidad de acierto: 0.8798
- Pregunta 8: Part 5 → Probabilidad de acierto: 0.8804
- Pregunta 9: Part 5 → Probabilidad de acierto: 0.6636
- Pregunta 10: Part 4 → Probabilidad de acierto: 0.7441
```

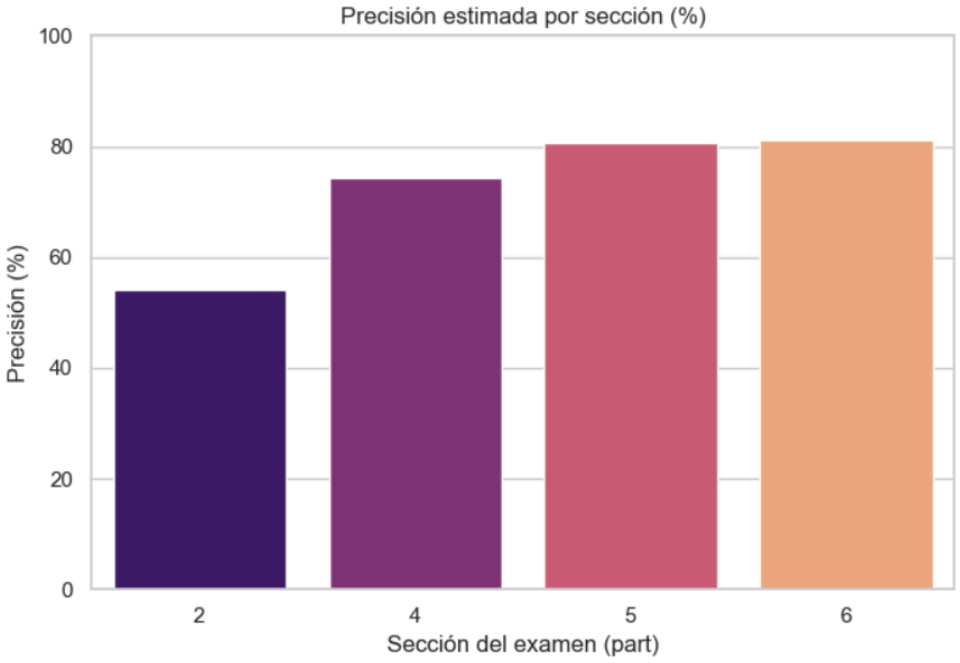
Así mismo, se puede realizar una estimación de acertar por sección del examen:

```
 Desglose por sección:
- Part 2: 2 preguntas → 1.08 aciertos esperados → 54.01% de precisión estimada
- Part 4: 1 preguntas → 0.74 aciertos esperados → 74.41% de precisión estimada
- Part 5: 4 preguntas → 3.23 aciertos esperados → 80.74% de precisión estimada
- Part 6: 3 preguntas → 2.43 aciertos esperados → 81.06% de precisión estimada
```

Visualización de Resultados

Finalmente una manera que elegimos de mostrar los resultados es mediante una gráfica que muestra la cantidad de aciertos esperados por sección del examen, y por otro lado, el porcentaje de precisión estimada por sección


Aciertos esperados por sección (part)
Aciertos esperados
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2
4
5
6
Sección del examen (part)


Precisión estimada por sección (%)
Precisión (%)
100
80
60
40
20
0
2
4
5
6
Sección del examen (part)

Conclusiones

Los resultados obtenidos demuestran que el modelo predictivo basado en XGBoost logra capturar efectivamente los patrones complejos presentes en los datos educativos, alcanzando un AUC de 0.847. Este valor indica una capacidad notable para distinguir entre respuestas correctas e incorrectas, superando ampliamente métodos tradicionales de evaluación.

Desde nuestra perspectiva, el verdadero valor del modelo no radica únicamente en su precisión estadística, sino en su potencial para transformar la práctica educativa. Consideramos que el mayor logro de este trabajo es haber desarrollado un puente entre la analítica de datos y la toma de decisiones educativas.

Los resultados no solo validan la eficacia técnica del enfoque, sino que abren posibilidades reales para personalizar el aprendizaje. Futuras investigaciones podrían explorar la integración de este predictor con sistemas de tutoría inteligente, creando así un ciclo continuo de evaluación, adaptación y enseñanza.

En conclusión, este proyecto evidencia que el machine learning aplicado a la educación puede ir más allá de lo teórico, ofreciendo herramientas prácticas que beneficien por igual a estudiantes, docentes e instituciones. El balance alcanzado entre precisión técnica y utilidad pedagógica marca un prometedor camino para innovar en evaluación educativa.

Referencias

■ Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785-794. https://doi.org/10.1145/2939672.2939785

■ Baker, F. B., & Kim, S.-H. (2004). Item Response Theory: Parameter Estimation Techniques (2nd ed.). CRC Press.

■ Romero, C., & Ventura, S. (2020). Educational data mining and learning analytics: An updated survey. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 10(3), e1355. https://doi.org/10.1002/widm.1355

■ Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.