INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

MNA - Maestría Inteligencia Artificial Aplicada

Proyecto.

A01796362 Eduardo Perez Carrillo

A01365006 Oscar Gerardo Álvarez Mejía

A01797139 Deménard Gardy Armand

A01796858 Luis Esteban Barranco Guida

A01796859 Fernando Abel Leal Villavicencio

Docente Titular:

Dr. Gerardo Rodríguez Hernández

Maestro Ricardo Valdez Hernández

Docente Asistente:

Maestra María Mylen Treviño Elizondo

Docente Tutor:

Livan Israel Valdes Esquivel

Operaciones de aprendizaje automático

02 Noviembre 2025

Índice

Índice de Ilustraciones

Índice de Tablas

Capítulo 1. Introducción

La evaluación escolar constituye uno de los pilares fundamentales del proceso educativo, ya que permite medir no solo el nivel de conocimiento adquirido por los estudiantes, sino también la efectividad de las estrategias pedagógicas implementadas por las instituciones [1]. A través de las calificaciones, los docentes y autoridades académicas obtienen información clave para identificar avances, rezagos y necesidades de apoyo específicas [2]. Sin embargo, el rendimiento académico no depende únicamente de las capacidades cognitivas de los estudiantes, sino que está influido por una amplia gama de factores sociales, económicos y personales que determinan su desempeño dentro y fuera del aula [3].

En este sentido, mantener un registro sistemático y analítico de estos factores se vuelve esencial para comprender la complejidad del aprendizaje [4]. Variables como el nivel educativo de los padres, el tipo de institución de procedencia, el idioma de instrucción o incluso el entorno socioeconómico pueden impactar de manera significativa en las calificaciones finales [4]. Analizar dichos elementos desde un enfoque cuantitativo y con herramientas basadas en datos permite no solo observar patrones históricos, sino también anticipar comportamientos futuros. Así, la adopción de medidas orientadas por evidencia, basadas en criterios objetivos y en el análisis de datos académicos, contribuye a diseñar estrategias de intervención más efectivas, equitativas y personalizadas para cada grupo de estudiantes [5].

En los últimos años, la integración de técnicas de *machine learning* en el ámbito educativo ha abierto nuevas oportunidades para el análisis predictivo del rendimiento estudiantil [6]. Sin embargo, el desarrollo de modelos de predicción efectivos requiere más que un entrenamiento algorítmico [7]: demanda procesos estructurados de gestión, monitoreo y mejora continua. En este contexto, el enfoque MLOps (*Machine Learning Operations*) se presenta como una metodología esencial para garantizar la trazabilidad, la reproducibilidad y la actualización constante de los modelos predictivos [8]. Al incorporar prácticas de ingeniería de software en el ciclo de vida del aprendizaje automático, MLOps permite automatizar la recolección de datos, el entrenamiento y la validación de modelos, asegurando que los resultados sean consistentes y útiles para la toma de decisiones educativas [9].

De esta manera, la combinación entre la analítica de datos educativos y la implementación de flujos MLOps ofrece una base sólida para transformar la gestión académica tradicional en un proceso inteligente y proactivo [10]. Este tipo de sistemas no solo predicen el rendimiento académico de los estudiantes, sino que además facilitan la identificación temprana de casos de riesgo, la optimización de recursos institucionales y la creación de políticas de apoyo fundamentadas en evidencia [11].

Finalmente, el presente documento se estructura de la siguiente manera: en el Capítulo dos se presenta el planteamiento del problema, donde se define la necesidad de implementar un modelo predictivo y se delimitan sus objetivos específicos. En el Capítulo tres se expone el marco teórico que sustenta la investigación, abarcando los conceptos relacionados con el aprendizaje automático, la analítica educativa y la metodología MLOps. El Capítulo cuatro describe la metodología empleada, incluyendo la definición del *dataset*, las variables analizadas y el flujo de trabajo propuesto. En el Capítulo cinco se presentan los resultados obtenidos durante la fase de análisis y modelado, mientras que el Capítulo seis expone las conclusiones generales y las recomendaciones derivadas del estudio.

Capítulo 2. Planteamiento del problema

Durante las últimas décadas, los sistemas educativos han experimentado una transformación significativa impulsada por la globalización, la digitalización y la creciente demanda de capital humano altamente calificado [12]. Este cambio ha puesto de manifiesto la necesidad de contar con mecanismos más precisos y objetivos para evaluar el rendimiento de los estudiantes y anticipar los factores que inciden en su éxito académico. En el contexto de la educación superior, donde los procesos de admisión y permanencia resultan determinantes para la trayectoria profesional, disponer de herramientas analíticas que permitan identificar patrones de desempeño se ha convertido en un componente esencial de la gestión educativa moderna [13].

Tradicionalmente, la evaluación del rendimiento estudiantil se ha limitado al análisis descriptivo de las calificaciones obtenidas, sin considerar de forma integral las variables que las condicionan. Sin embargo, la evidencia acumulada en diversos estudios ha demostrado que el desempeño académico está estrechamente relacionado con factores sociales, económicos y familiares, tales como el nivel educativo de los padres, la ocupación del hogar, el tipo de escuela de procedencia o el idioma de instrucción. Estas variables, aunque registradas en los sistemas institucionales, rara vez se aprovechan con fines analíticos o predictivos. El resultado es una pérdida de información valiosa que podría contribuir a mejorar la toma de decisiones en materia de becas, tutorías y programas de apoyo académico [14].

En particular, en los procesos de admisión universitaria, los responsables de las políticas educativas enfrentan el reto de evaluar grandes volúmenes de datos provenientes de aspirantes con contextos muy diversos. Identificar a los estudiantes con mayor probabilidad de éxito o de riesgo académico de forma temprana permitiría diseñar estrategias de acompañamiento más efectivas y equitativas. Sin embargo, esta tarea resulta compleja cuando se realiza de manera manual o utilizando herramientas

estadísticas tradicionales, pues éstas no siempre capturan las relaciones no lineales ni las interacciones entre las variables involucradas.

En este escenario, el aprendizaje automático (*machine learning*) surge como una alternativa sólida para el análisis y la predicción del rendimiento académico, al ofrecer modelos capaces de aprender de los datos históricos y generar estimaciones con alto nivel de precisión. No obstante, la implementación de estos modelos plantea desafíos relacionados con la gestión de datos, la reproducibilidad de resultados y la actualización continua del sistema.

Es aquí donde la metodología MLOps cobra relevancia, al proporcionar un marco estructurado que integra prácticas de ingeniería de software con los flujos de trabajo del aprendizaje automático. Gracias a MLOps, es posible establecer procesos estandarizados para la recolección, entrenamiento, validación, despliegue y monitoreo de modelos predictivos de forma automatizada y sostenible en el tiempo.

El problema que motiva esta investigación radica en la ausencia de un sistema predictivo que permita estimar, con base en información demográfica, académica y socioeconómica, el nivel de rendimiento esperado de los aspirantes a instituciones de educación superior. Actualmente, las decisiones en torno a la admisión o el apoyo académico se sustentan principalmente en criterios cuantitativos simples como el promedio de calificaciones o el resultado de exámenes sin aprovechar el potencial analítico de los datos disponibles. Esta limitación reduce la capacidad de las instituciones para detectar oportunamente a los estudiantes en riesgo de bajo desempeño y para diseñar políticas educativas más inclusivas y basadas en evidencia.

Para abordar esta problemática, se dispone de un conjunto de datos que integra información demográfica, académica y socioeconómica de aspirantes a instituciones de educación superior, el cual permitirá posteriormente analizar los factores asociados al desempeño académico mediante técnicas de clasificación supervisada.

Por tanto, la presente investigación propone el desarrollo de un modelo de clasificación supervisada que prediga el nivel de rendimiento académico de los estudiantes (*Excellent, Good, Average, Very Good*) a partir de variables demográficas, educativas y familiares.

Este planteamiento no busca únicamente desarrollar un modelo de predicción, sino también sentar las bases para un sistema sostenible, escalable y ético de analítica educativa, capaz de contribuir a la mejora de la calidad y la equidad en los procesos de enseñanza superior.

Capítulo 3. Marco teórico

3.1 Educación superior y evaluación de aspirantes

Las instituciones de educación superior constituyen un eje estratégico para el desarrollo social, científico y económico de cualquier país [15]. En ellas convergen los esfuerzos formativos orientados a la creación de capital humano capaz de responder a los desafíos tecnológicos y productivos contemporáneos [16]. Sin embargo, la creciente demanda de acceso a la universidad ha exigido la implementación de mecanismos de selección más precisos, transparentes y basados en criterios objetivos [17].

Los exámenes de admisión se han consolidado como una de las herramientas más utilizadas para valorar las competencias cognitivas y el potencial académico de los aspirantes [18]. Estos instrumentos buscan determinar, de manera estandarizada, el nivel de conocimientos y habilidades requeridas para cursar estudios universitarios. No obstante, la interpretación de sus resultados no siempre considera la diversidad de factores que influyen en el desempeño: las diferencias socioeconómicas, el tipo de formación previa o la disponibilidad de recursos educativos pueden impactar significativamente en los puntajes obtenidos [19].

En este contexto, la analítica educativa surge como un enfoque innovador que permite aprovechar los datos institucionales para comprender mejor los patrones de aprendizaje y desempeño de los estudiantes. Mediante la aplicación de técnicas estadísticas y de inteligencia artificial, es posible transformar la información acumulada en conocimiento útil para la gestión académica, la mejora de los procesos de admisión y la formulación de políticas de equidad educativa [20].

3.2 Aprendizaje automático

El aprendizaje automático (*Machine Learning*) se define como una rama de la inteligencia artificial que permite a los sistemas aprender patrones a partir de los datos sin ser explícitamente programados [21]. En lugar de depender de reglas fijas, los algoritmos de *machine learning* construyen modelos capaces de generalizar comportamientos y realizar predicciones basadas en ejemplos previos [22].

En el ámbito educativo, esta disciplina ha adquirido una relevancia creciente debido a su capacidad para identificar tendencias ocultas en los datos de los estudiantes, predecir su rendimiento y proponer estrategias personalizadas de apoyo académico [23]. Los modelos de *machine learning* pueden procesar grandes volúmenes de información y encontrar correlaciones entre variables que, de otra forma, pasarían desapercibidas [24].

Entre las aplicaciones más comunes destacan la predicción del desempeño académico, la detección de abandono escolar, la evaluación del aprendizaje adaptativo y la recomendación de cursos o tutorías personalizadas. Estas herramientas apoyan la toma

de decisiones institucionales y contribuyen a la mejora continua de los procesos formativos [25].

3.3 Aprendizaje supervisado

Dentro del campo del aprendizaje automático, el aprendizaje supervisado representa la categoría más utilizada para tareas predictivas. Este enfoque se basa en el uso de un conjunto de datos etiquetados, es decir, donde se conoce la variable de salida o *target*, para entrenar un modelo que aprenda la relación entre las entradas y los resultados esperados [26].

Los algoritmos supervisados se dividen principalmente en dos tipos: clasificación y regresión. En la clasificación, el objetivo es asignar una etiqueta a cada observación. En la regresión, en cambio, se busca estimar un valor continuo [27].

Algunos de los modelos más utilizados en clasificación incluyen los árboles de decisión, los bosques aleatorios (*Random Forest*), los métodos de ensamble como *Gradient Boosting (XGBoost o LightGBM*) y la regresión logística, cada uno con ventajas particulares en términos de interpretabilidad, velocidad de entrenamiento y precisión [28].

3.4 Datasets y analítica educativa

El concepto de *dataset* se refiere a un conjunto estructurado de datos que almacena información sobre entidades o eventos específicos, organizada en filas (instancias) y columnas (atributos) [29]. En el contexto del aprendizaje automático, la calidad, representatividad y consistencia del *dataset* son factores determinantes para la efectividad del modelo predictivo [30].

Los datasets educativos suelen contener información de tipo demográfico (género, edad, procedencia), académico (promedios, resultados de exámenes, tipo de institución) y socioeconómico (ocupación de los padres, nivel de ingreso, tipo de vivienda, etc.). La preparación del *dataset* implica diversas etapas: limpieza de datos, manejo de valores faltantes, normalización de variables y codificación de categorías [29][30].

Además, es fundamental garantizar la ética en el tratamiento de los datos, preservando la privacidad de los estudiantes y evitando sesgos en la interpretación de los resultados. Un modelo de predicción educativo no debe replicar desigualdades sociales, sino contribuir a mitigarlas mediante un análisis objetivo y transparente [31].

3.5 MLOps: Integración y operación de modelos de aprendizaje automático

El término MLOps combina las prácticas de *Machine Learning* y *DevOps* para gestionar el ciclo de vida completo de los modelos predictivos de forma automatizada, reproducible y escalable [32]. Su objetivo es facilitar la colaboración entre científicos de datos, ingenieros y analistas, garantizando que los modelos no solo se desarrollen

correctamente, sino que también se desplieguen, supervisen y actualicen de manera continua [33].

El flujo de trabajo típico de MLOps se compone de las siguientes etapas [34][35][36]:

1. Definición de requerimientos y recolección de datos: identificación de las fuentes, formatos y variables relevantes.
2. Preparación y limpieza de datos: normalización, imputación de valores faltantes y codificación de atributos categóricos.
3. Entrenamiento y validación del modelo: selección de algoritmos, ajuste de hiperparámetros y evaluación mediante métricas como *Accuracy, Precision, Recall y F1-Score.*
4. Registro y versionamiento: almacenamiento controlado de modelos, *datasets* y configuraciones mediante herramientas como MLflow o DVC.
5. Despliegue: implementación del modelo en un entorno de producción, ya sea en servidores locales o en la nube.
6. Automatización con contenedores: uso de tecnologías como Docker o Kubernetes para asegurar portabilidad, consistencia y escalabilidad en los entornos de ejecución.
7. Monitoreo y mantenimiento: seguimiento de métricas en tiempo real, detección de *data drift* y reentrenamiento periódico.
8. Gobernanza y auditoría: trazabilidad de los resultados, control de versiones y cumplimiento de criterios éticos y de privacidad.

En conjunto, el enfoque MLOps transforma el desarrollo de modelos en un proceso continuo, colaborativo y automatizado, garantizando que las predicciones se mantengan vigentes y confiables a lo largo del tiempo.

3.6 Evaluación de métricas de desempeño

La evaluación del desempeño de un modelo de aprendizaje automático constituye una etapa fundamental dentro del proceso de desarrollo, ya que permite determinar su capacidad para realizar predicciones correctas y generalizar el conocimiento aprendido a nuevos datos. En el caso de los modelos de clasificación supervisada las métricas más utilizadas para medir la calidad de las predicciones son *Accuracy, Precision, Recall y F1-Score*, además del uso de la matriz de confusión como herramienta de diagnóstico visual y cuantitativo [37][38].

La matriz de confusión es una representación tabular que muestra el número de aciertos y errores cometidos por el modelo en cada categoría. Cada fila de la matriz representa

las instancias reales, mientras que cada columna representa las predicciones realizadas. A partir de esta matriz es posible identificar con precisión en qué clases el modelo tiende a confundirse, proporcionando una visión más detallada que la que ofrecen las métricas agregadas [39].

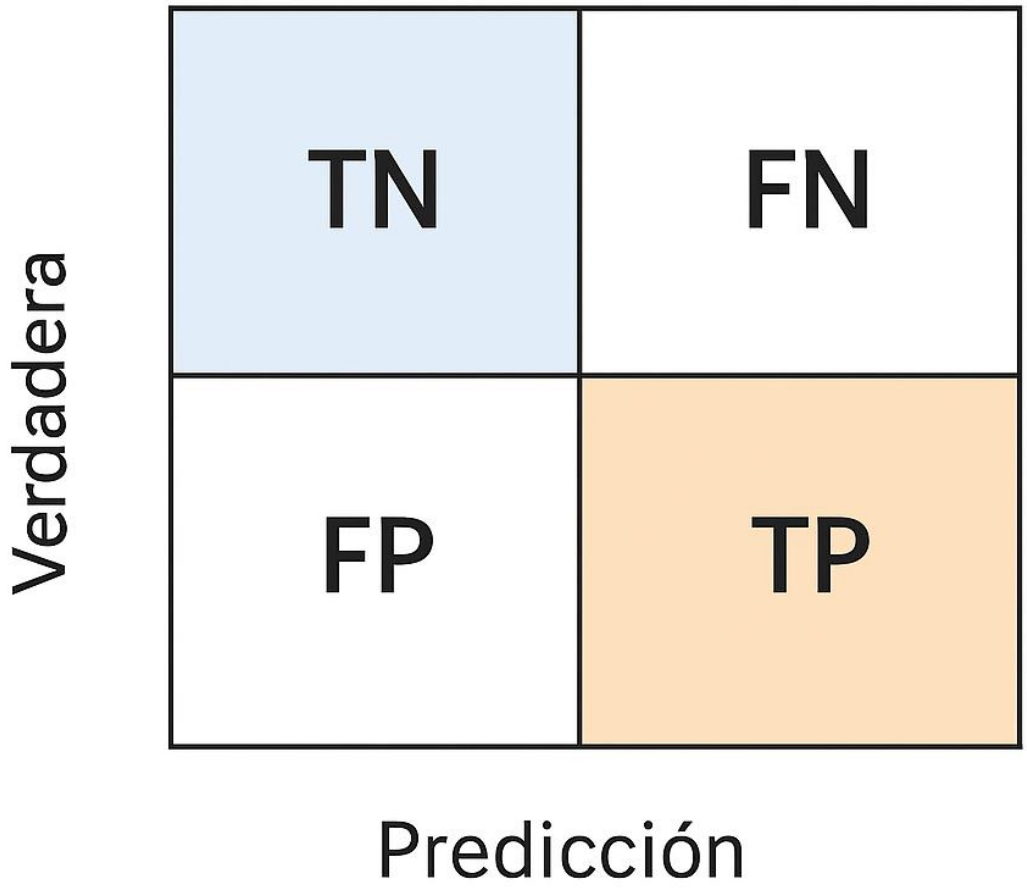

Ilustración 1 Ejemplo de Matriz de Confusión

El cuadrante TN (*True Negative*) corresponde a los casos en los que el modelo predijo correctamente la ausencia de la característica o clase de interés; es decir, identificó como “negativos” aquellos casos que realmente lo eran. Por su parte, el FN (*False Negative*) refleja los errores en los que el modelo no detectó una instancia positiva, clasificándola incorrectamente como negativa. En contraste, el FP (*False Positive*) muestra los casos en los que el modelo predijo erróneamente la presencia de la clase cuando en realidad no existía, mientras que el TP (*True Positive*) representa las predicciones acertadas en las que el modelo identificó correctamente una instancia positiva. En conjunto, estos cuatro valores permiten evaluar con precisión el equilibrio entre aciertos y errores, ofreciendo una visión integral del comportamiento del clasificador y de su capacidad para distinguir entre las distintas categorías analizadas [39].

La *accuracy* representa la proporción de predicciones correctas sobre el total de observaciones analizadas. Es una métrica global que indica, en términos generales, qué tan bien clasifica el modelo. Matemáticamente se expresa como [40]:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Aunque es una medida intuitiva, su interpretación debe realizarse con precaución en *datasets* desbalanceados, ya que puede ocultar deficiencias en clases minoritarias.

La *precision* mide la proporción de predicciones positivas que fueron realmente correctas. Indica la confiabilidad del modelo cuando clasifica una instancia en una categoría específica. Se define como [40]:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Un valor alto de precisión implica que el modelo comete pocos falsos positivos. En contextos educativos, una alta precisión significa que la mayoría de los estudiantes identificados como de “bajo rendimiento”, efectivamente lo son.

El *recall* evalúa la capacidad del modelo para detectar correctamente todos los casos pertenecientes a una clase. Es decir, mide cuántos elementos relevantes fueron realmente recuperados. Un valor alto de *recall* indica que el modelo logra identificar la mayoría de los casos de interés [41]:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

El *F1-Score* combina *precision* y *recall* en una sola métrica, equilibrando ambos indicadores mediante su media armónica [41]:

$$F1 = 2\,\frac{Precision * Recall}{Precision + recall} \quad (4)$$

Esta métrica resulta especialmente útil cuando el *dataset* está desbalanceado o cuando tanto los falsos positivos como los falsos negativos son relevantes. Un valor de *F1-Score* cercano a 1 indica un modelo equilibrado en su capacidad de identificar correctamente las clases.

3.7 Estructuración estandarizada del proyecto con Cookiecutter

La adopción de una estructura modular basada en *Cookiecutter Data Science* permite establecer un entorno ordenado, coherente y fácilmente extensible para el ciclo de vida

del modelo. Este enfoque parte del principio de que un proyecto de aprendizaje automático debe organizar sus componentes, datos, código, modelos y reportes, en una jerarquía clara que favorezca la colaboración y la trazabilidad. La plantilla de *Cookiecutter* ofrece un esquema estandarizado que separa la lógica de procesamiento, el almacenamiento de artefactos y la documentación, de modo que cada elemento pueda versionarse y actualizarse de manera independiente.

En términos de ingeniería de software, esta organización favorece la mantenibilidad del código y la reproducibilidad de los experimentos, al mismo tiempo que reduce la dependencia entre módulos. Gracias a esta estructura, el proyecto mantiene una identidad uniforme en todas sus fases, alineándose con los lineamientos de reproducibilidad propuestos por *DrivenData* [43], quienes señalan que un proyecto de ciencia de datos bien estructurado debe ser legible, modular y fácilmente automatizable.

3.8 Refactorización modular del código

La refactorización del código se fundamenta en los principios de ingeniería limpia y programación modular, con el objetivo de mejorar la eficiencia, la legibilidad y la mantenibilidad del sistema. Este proceso consiste en reorganizar las funciones escritas en notebooks hacia scripts independientes con una responsabilidad específica, siguiendo los lineamientos del Single *Responsibility Principle* y las buenas prácticas descritas [44]. A través de esta reestructuración, cada etapa del pipeline es encapsulada en un módulo autónomo, lo que favoreció la depuración y el control de versiones.

La modularidad permite además incorporar estrategias de programación orientada a objetos cuando fue pertinente, lo que incrementó la reutilización del código y la claridad de las dependencias. Esta práctica se considera esencial dentro de MLOps, ya que reduce la deuda técnica y asegura que los cambios futuros no afecten el comportamiento global del sistema, manteniendo la estabilidad y coherencia del proyecto.

3.9 Automatización del *pipeline* de modelado

El desarrollo del pipeline de modelado se basa en la filosofía de automatizar cada etapa del ciclo de vida del aprendizaje automático, desde la adquisición y limpieza de los datos hasta la evaluación final del modelo. La automatización mediante *pipelines* de Scikit-Learn garantiza que el flujo de procesamiento se ejecute siempre bajo las mismas condiciones, evitando errores manuales y asegurando la reproducibilidad de los resultados.

El uso de pipelines permite concatenar procesos de preprocesamiento y modelado en una única estructura lógica que facilita tanto el entrenamiento como la validación cruzada[45]. Este enfoque, ampliamente adoptado en entornos de producción, también responde a los lineamientos del marco de referencia propuesto por Google Cloud[46], en

el cual la automatización y el versionamiento son elementos clave para alcanzar los niveles 1 y 2 de madurez en MLOps.

3.10 Seguimiento, versionamiento y gestión de experimentos

Una parte esencial es implementar mecanismos de seguimiento de experimentos y versionamiento de modelos mediante herramientas especializadas como Weights & Biases (W&B) y MLflow. Estas plataformas permiten registrar de manera sistemática los parámetros, métricas y artefactos generados durante el entrenamiento, lo que garantiza la trazabilidad completa del flujo de trabajo.

MLflow facilita la gestión del ciclo de vida del modelo mediante el registro de configuraciones, la exportación del pipeline y la integración con sistemas de despliegue. Complementariamente [47], Biewald destaca que W&B refuerza la reproducibilidad y la colaboración entre equipos al centralizar el seguimiento de métricas y artefactos experimentales [48]. Al combinar ambas herramientas, el proyecto logró un control integral de los experimentos, integrando prácticas de versionamiento propias de DevOps con las necesidades específicas del aprendizaje automático.

3.11 Reproducibilidad y trazabilidad en aprendizaje automático

La reproducibilidad constituye uno de los pilares fundamentales de la investigación científica y de la ingeniería de modelos de aprendizaje automático. Se define como la capacidad de replicar un experimento y obtener resultados consistentes bajo las mismas condiciones, incluyendo los mismos datos, configuraciones y versiones de dependencias. La trazabilidad, por su parte, se refiere al registro histórico de cada etapa del proceso, desde la adquisición de los datos hasta la generación de los resultados, lo que permite auditar y validar la integridad del flujo de trabajo.

En el contexto del aprendizaje automático, ambos conceptos garantizan la transparencia y confiabilidad de los modelos, evitando la pérdida de información asociada a versiones o configuraciones no documentadas. Pineau et al. Algunos autores destacan que la reproducibilidad se convierte en un criterio esencial para evaluar la calidad de la investigación en IA [49], mientras que otros enfatizan que la trazabilidad refuerza la responsabilidad científica al permitir reconstruir el proceso completo de toma de decisiones del modelo [50].

3.12 Versionamiento de datos y modelos

El versionamiento de datos y el registro de modelos representan pilares fundamentales dentro de los entornos MLOps, ya que permiten mantener la coherencia y trazabilidad de todos los artefactos involucrados en el ciclo de vida del aprendizaje automático. El control de versiones sobre los conjuntos de entrenamiento y validación asegura la preservación

del historial de cambios, lo que facilita su recuperación, auditoría y comparación a lo largo de las distintas etapas del desarrollo [51].

Estas prácticas fortalecen la reproducibilidad de los experimentos y previenen discrepancias entre las ejecuciones, garantizando que cada resultado pueda ser rastreado hasta su configuración original. El versionamiento constituye un componente estructural dentro de los flujos de trabajo en aprendizaje automático destacando el papel de MLflow como una herramienta que centraliza el registro, comparación y gestión de modelos, consolidando un enfoque sistemático para el seguimiento de experimentos en entornos MLOps [47].

3.13 Explicabilidad y transparencia de los modelos

La explicabilidad de los modelos, también conocida como *Explainable Artificial Intelligence (XAI)*, se consolida como un campo fundamental dentro de la inteligencia artificial moderna, ya que busca hacer comprensibles los procesos de decisión de los sistemas de aprendizaje automático. En ámbitos donde las predicciones influyen directamente en personas o instituciones, como el sector educativo o financiero, la interpretabilidad de los resultados se convierte en un elemento clave para generar confianza y favorecer la adopción de soluciones basadas en IA. Ribeiro, Singh y Guestrin [53] presentan el método LIME (*Local Interpretable Model-Agnostic Explanations*), una propuesta que ofrece explicaciones locales sobre la influencia de cada variable en una predicción específica, sin depender del tipo de modelo utilizado.

Incorporar la explicabilidad en los modelos de aprendizaje automático permite comprender no sólo su nivel de precisión, sino también la lógica que guía sus decisiones. Este enfoque aporta transparencia al proceso, facilita la detección de posibles sesgos y contribuye a una validación ética más sólida, lo que incrementa la confianza en el sistema y refuerza su credibilidad en contextos reales de aplicación.

Capítulo 4. Metodología (Parcial)

La presente investigación se enmarca como aplicada, cuantitativa, no experimental y de alcance exploratorio–descriptivo. El objetivo específico es: desarrollar un modelo de clasificación supervisada que prediga el nivel de rendimiento académico de los estudiantes (*Excellent, Good, Average, Very Good*) a partir de variables demográficas, educativas y familiares.

El *dataset* utilizado en esta investigación corresponde a una versión modificada del dataset “Student Performance on an Entrance Examination” [42] recopilado por el Prof. *Jiten Hazarika*. Este dataset contiene información de los aspirantes que participaron en el examen de admisión a instituciones de educación médica en el estado de Assam, India. La base de datos integra variables demográficas, académicas y socioeconómicas, entre las que destacan el género, el tipo de institución educativa de procedencia, el idioma de instrucción, las calificaciones obtenidas en los niveles de educación secundaria y media superior, así como la ocupación de los padres. La versión utilizada en este estudio fue adaptada y normalizada para su análisis mediante técnicas de aprendizaje supervisado, garantizando la consistencia en las etiquetas de clasificación y la integridad de las variables relevantes para la predicción del rendimiento académico.

El *dataset* empleado para el entrenamiento y análisis del modelo está conformado por un total de 679 registros y 13 columnas. Cada registro corresponde a un aspirante que participó en el examen de admisión a instituciones de educación médica del estado de Assam, India.

Está compuesto por trece variables, distribuidas entre atributos demográficos, académicos y socioeconómicos. Las variables incluidas son: Performance, que representa el nivel de rendimiento académico del estudiante; *Gender*, que indica el sexo del aspirante; *Caste*, asociada al grupo social de pertenencia; *coaching*, que especifica si el estudiante recibió preparación adicional y en qué modalidad; time, que refleja la duración o periodo de dicho apoyo; *Class_ten_education* y *twelve_education*, que describen el tipo de institución educativa cursada en los niveles de secundaria y preparatoria, respectivamente; *medium*, que señala el idioma de instrucción; *Class_X_Percentage* y *Class_XII_Percentage*, que registran los porcentajes de calificación obtenidos en cada nivel; *Father_occupation* y *Mother_occupation*, que detallan la ocupación de los padres; y finalmente *mixed_type_col*, una columna auxiliar que contiene valores heterogéneos derivados del proceso de recopilación original.

A continuación, se presenta la descripción detallada del proceso metodológico correspondiente a la Fase 1, en la cual se documentan los requerimientos del proyecto, se realiza la preparación y exploración del conjunto de datos, y se construye el modelo predictivo inicial que servirá como base para las siguientes etapas del ciclo de aprendizaje automático.

4.1 Fase 1: Análisis y diseño del modelo predictivo.

En esta etapa, el *Subject Matter Expert* (SME) asume un rol central al coordinar las actividades del equipo y distribuir la carga de trabajo conforme a las competencias técnicas y responsabilidades de cada integrante. Su participación asegura la correcta alineación entre los requerimientos del dominio educativo y el desarrollo técnico del modelo, promoviendo una ejecución ordenada, colaborativa y orientada a resultados verificables. A continuación, se presenta la tabla que detalla la asignación de funciones y tareas específicas dentro de esta fase.

Tabla *1. Roles y Tareas Fase 1*

Rol	Nombre	Tareas
Subject Matter Expert	Eduardo Perez	Comprender el problema a resolver con el *dataset*. Colaborar en la creación y documentación de los requerimientos (ML Canvas). Definir junto al equipo cuál será el objetivo del modelo Proponer cómo una solución basada en ML puede aportar valor a estudiantes. Análisis de Resultados de Fase 1.
Data Scientist	Fernando Leal	Analizar los datos con técnicas de *Exploratory Data Analysis* (EDA). Detectar patrones, relaciones y valores atípicos. Preparar los datos aplicando transformaciones como normalización o codificación de categorías. Probar diferentes algoritmos de *Machine Learning*. Evaluar los modelos con métricas adecuadas.
Data Engineer	Oscar Álvarez	Encargarse de la limpieza de datos (eliminar valores nulos, duplicados o inconsistentes). Organizar y preparar los datos para que puedan usarse en los modelos. Documentar los cambios realizados para asegurar la trazabilidad.
DevOps	Deménard Gardy Armand	Apoyar en la organización del repositorio (estructura de carpetas, ramas en Git). Revisar que todo lo que se haga pueda repetirse de forma clara y ordenada. Colaborar en la documentación del proceso.
Machine Learning Architect	Esteban Barranco	Diseñar la ruta de trabajo del proyecto (desde los datos iniciales hasta la evaluación del modelo). Coordinar la colaboración entre el *Data Engineer* y el *Data Scientist*. Garantizar que el trabajo realizado en esta fase pueda crecer y adaptarse a etapas posteriores.

4.1.1 Análisis de Requerimientos

Una vez distribuida la carga de trabajo, el SME procede a la documentación de los requerimientos del proyecto utilizando el formato *Machine Learning Canvas* (ML Canvas). Esta herramienta permite estructurar de manera clara los elementos clave del sistema de aprendizaje automático, incluyendo la definición del problema, las fuentes de datos, las variables relevantes, los criterios de evaluación y la propuesta de valor. Su aplicación facilita una visión integral y coherente del proyecto, asegurando que cada decisión metodológica esté alineada con los objetivos del modelo predictivo y con las necesidades institucionales identificadas.

4.1.2 Manipulación y preparación de datos

Durante esta etapa se llevarán a cabo las tareas de limpieza, estandarización y control de versiones del conjunto de datos, con el propósito de preparar la información para su posterior análisis y modelado. En primer lugar, se implementará un proceso de depuración y validación de los registros, orientado a identificar y corregir inconsistencias en las variables, garantizando la uniformidad de los valores categóricos mediante operaciones de normalización textual y estandarización de etiquetas. Asimismo, se atenderá el tratamiento de valores nulos o ausentes, aplicando estrategias estadísticas que permitan sustituirlos por valores representativos de acuerdo con la naturaleza de cada variable, manteniendo la coherencia interna del conjunto de datos.

Adicionalmente, se contempla la eliminación de columnas o atributos no relevantes para los objetivos de la investigación, optimizando así la estructura del *dataset* y reduciendo el ruido que pudiera afectar el desempeño del modelo. Todo este proceso será documentado de forma sistemática, asegurando que las transformaciones realizadas puedan ser replicadas y auditadas en fases posteriores del proyecto. Finalmente, se gestionará el versionado de los artefactos de datos a través de herramientas especializadas, con el fin de garantizar la trazabilidad, transparencia y reproducibilidad dentro del ciclo de vida del proyecto MLOps.

Para el desarrollo de esta tarea se utilizará el lenguaje de programación Python, dado su amplio uso en proyectos de análisis de datos y su compatibilidad con herramientas de ciencia computacional.

En esta etapa se emplearán principalmente las librerías Pandas y NumPy. La primera permitirá estructurar y manipular los datos a través de marcos tipo *DataFrame*, facilitando la organización, depuración y normalización de las variables. Por su parte, NumPy se utilizará para realizar operaciones numéricas y manejar arreglos multidimensionales, optimizando el procesamiento interno de los datos.

4.1.3 Exploración y preprocesamiento de datos

Para esta fase se empleará el lenguaje de programación Python, apoyado principalmente en las bibliotecas Pandas y *Matplotlib*, que permiten realizar la exploración, descripción y visualización del conjunto de datos. A través de Pandas, se cargará la base de datos y se ejecutarán operaciones destinadas a examinar la estructura del *dataset*, incluyendo la cantidad de registros y variables, los nombres de las columnas, los tipos de datos y una vista preliminar de los primeros y últimos registros. Esto permitirá verificar la integridad de la información y detectar posibles inconsistencias o valores atípicos antes de proceder con el preprocesamiento.

Por su parte, mediante *Matplotlib* se generarán representaciones gráficas descriptivas que facilitarán la interpretación visual de las distribuciones y patrones presentes en las variables. Además, las visualizaciones permitirán analizar la distribución de la variable objetivo (*Performance*), así como la relación entre las características académicas y demográficas de los estudiantes.

4.1.4 Versionado de Datos.

Para el versionado de datos se empleará la herramienta *Weights & Biases*, debido a su capacidad para registrar, rastrear y gestionar de manera centralizada las distintas versiones de los conjuntos de datos utilizados durante el desarrollo del modelo. Su integración permitirá documentar cada modificación realizada en las etapas de limpieza, transformación y preparación, garantizando así la trazabilidad, reproducibilidad y transparencia del proceso metodológico. De esta forma, se asegurará que los resultados obtenidos puedan ser verificados y replicados en fases posteriores del ciclo MLOps.

4.1.5 Construcción, ajuste y evaluación de Modelos de Machine Learning.

En esta etapa se llevará a cabo la construcción, ajuste y evaluación del modelo de aprendizaje supervisado, con el propósito de predecir el nivel de rendimiento académico de los estudiantes a partir de sus características demográficas, educativas y familiares. Para ello, se empleará el lenguaje de programación Python, junto con las bibliotecas Pandas, NumPy, Scikit-learn, XGBoost, MLflow y *Weights & Biases* (W&B), las cuales conforman un entorno robusto para el procesamiento de datos, la creación de modelos predictivos y el control de versiones.

El proceso metodológico iniciará con la ingeniería de características (*Feature Engineering*), en la cual se aplicarán transformaciones destinadas a optimizar la representación de los datos antes del entrenamiento del modelo. Se utilizarán estructuras como *Pipeline* y *ColumnTransformer* para organizar las etapas de preprocesamiento y

aplicar operaciones específicas según el tipo de variable. Las variables numéricas serán tratadas mediante técnicas de imputación estadística (*SimpleImputer*) utilizando la mediana como valor representativo, mientras que las variables categóricas se codificarán mediante *OrdinalEncoder* y *OneHotEncoder*, con el fin de traducir la información cualitativa en formatos adecuados para el modelo. Este flujo permitirá estandarizar los datos y garantizar su compatibilidad con el algoritmo seleccionado.

Una vez finalizada la fase de preprocesamiento, se procederá a la construcción, entrenamiento y evaluación comparativa de cuatro modelos de clasificación supervisada, con el propósito de determinar cuál ofrece el mejor desempeño en la predicción del nivel de rendimiento académico de los estudiantes. Los modelos seleccionados para esta etapa serán: XGBoost, Regresión Logística (Logistic Regression), Máquinas de Vectores de Soporte (SVC) y un modelo de Regresión Logística Ordinal, elegidos por su efectividad en tareas de clasificación multiclase y su capacidad para manejar datos categóricos y ordinales.

El conjunto de datos será dividido en 80% para entrenamiento y 20% para validación, garantizando un proceso de evaluación equilibrado que reduzca el riesgo de sobreajuste. Para medir el rendimiento de cada modelo se emplearán las métricas Accuracy, Precision, Recall y F1-Score, con el fin de valorar tanto la exactitud global como el desempeño individual por categoría.

Adicionalmente, se generarán las curvas de aprendizaje para cada uno de los modelos con el objetivo de analizar el comportamiento del error de entrenamiento y validación a medida que aumenta el tamaño del conjunto de datos. Estas gráficas permitirán evaluar la capacidad de generalización de los algoritmos y detectar posibles indicios de sobreajuste o subajuste, proporcionando una comprensión más profunda sobre la estabilidad y el rendimiento de cada enfoque antes de su integración en el pipeline MLOps.

4.1.6 Control de Versiones

Con el propósito de asegurar la organización, trazabilidad y control de versiones de todos los componentes desarrollados, se creará un repositorio del proyecto en la plataforma GitHub. Este espacio permitirá centralizar el código fuente, los notebooks de análisis, los scripts de procesamiento y los archivos de configuración, garantizando una gestión ordenada del flujo de trabajo.

4.2 Fase 2: Implementación y gestión operativa del pipeline de Machine Learning

La segunda fase del proyecto se centra en la implementación práctica y operativa del sistema de aprendizaje automático, con el propósito de consolidar un entorno de trabajo estructurado, reproducible y escalable. Tras haber definido los requerimientos, comprendido el conjunto de datos y seleccionado los modelos base en la Fase 1, esta etapa busca transformar esos elementos en un flujo de trabajo formal que integre las mejores prácticas de ingeniería en *Machine Learning*. En este sentido, la metodología MLOps cobra nuevamente relevancia, al proporcionar los mecanismos necesarios para garantizar la trazabilidad, la organización modular del código y la gestión eficiente de los experimentos, fortaleciendo así la solidez técnica y la sostenibilidad del proyecto a largo plazo.

El enfoque adoptado en esta fase prioriza la profesionalización del desarrollo mediante la estructuración del proyecto bajo el esquema *Cookiecutter*, la refactorización modular del código, la automatización del pipeline de modelado y la implementación de herramientas de seguimiento y versionamiento como *MLflow* y DVC. Estas acciones no solo permiten un control integral del ciclo de vida del modelo, sino que también aseguran la consistencia entre los resultados obtenidos y las decisiones metodológicas adoptadas. De esta manera, la Fase 2 constituye el puente entre la experimentación inicial y la consolidación de un sistema predictivo operativo, capaz de integrarse de forma eficiente dentro de un entorno colaborativo y alineado con las prácticas actuales de MLOps.

En esta segunda fase, los roles establecidos en la etapa anterior conservan su estructura funcional, pero amplían su participación hacia la dimensión operativa y técnica del proyecto. Cada integrante asume tareas orientadas a consolidar la implementación del pipeline de Machine Learning, asegurando que el flujo de trabajo mantenga coherencia con los principios de reproducibilidad, trazabilidad y automatización definidos por el enfoque MLOps. Esta etapa se distingue por un trabajo colaborativo más especializado, donde la integración entre código, datos y herramientas adquiere un papel central en la gestión eficiente del modelo. A continuación, se presenta la tabla que resume las funciones y responsabilidades asignadas a cada rol durante el desarrollo de la Fase 2.

Tabla 2 Roles y Tareas Fase 2

Rol	Nombre	Tareas
Subject Matter Expert	Eduardo Perez	Supervisar la correcta implementación del pipeline de *Machine Learning* conforme a los objetivos establecidos en la Fase 1. Documentar las decisiones técnicas y metodológicas que sustentan la trazabilidad del sistema.
Data Scientist	Fernando Leal	Desarrollar y refactorizar el código del pipeline utilizando estructuras modulares. Implementar el entrenamiento, evaluación y ajuste de modelos en Scikit-Learn. Registrar y comparar los experimentos mediante W&B
Data Engineer	Oscar Álvarez	Integrar el control de versiones de datos, asegurando la trazabilidad entre *datasets* procesados y modelos entrenados. Gestionar la dependencia de librerías con *Poetry*. Mantener la consistencia entre los archivos de configuración (YAML/ENV) y las rutas del proyecto. Garantizar la reproducibilidad del pipeline a través de scripts automatizados.
DevOps	Deménard Gardy Armand	Estructurar el repositorio bajo el estándar *Cookiecutter ML*. Configurar el versionamiento de código con Git y la integración de herramientas de seguimiento de experimentos (W&B). Automatizar la ejecución del pipeline en entornos reproducibles. Supervisar el registro de modelos y la documentación técnica asociada.
Machine Learning Architect	Esteban Barranco	Diseñar la arquitectura integral del pipeline y su integración con las herramientas de control. Coordinar la interacción entre los componentes del sistema Validar la escalabilidad del proyecto y la correcta implementación del flujo MLOps. Definir lineamientos para las siguientes fases del ciclo de vida del modelo.

4.2.1 Estructuración del proyecto con *Cookiecutter*.

La organización del repositorio se basa en los principios del esquema *Cookiecutter* para proyectos de *Machine Learning*, el cual propone una estructura modular, clara y reproducible que favorece la gestión completa del ciclo de vida del modelo. Este formato garantiza una distribución ordenada de los componentes del proyecto y facilita el trabajo colaborativo entre los distintos roles técnicos. En esta estructura, la carpeta src/ contiene el código fuente encargado de las fases de preprocesamiento, entrenamiento y evaluación de los modelos; mientras que data/ almacena los conjuntos de datos tanto crudos como procesados. De igual forma, la carpeta outputs/ reúne las métricas, visualizaciones y resultados generados por el pipeline, manteniendo una relación coherente entre los datos de entrada y los productos finales obtenidos.

Por otro lado, los archivos de configuración como config.yaml y .env.example especifican las rutas, variables de entorno y parámetros que permiten ejecutar el proyecto de manera controlada y adaptable. La inclusión de Dockerfile y docker-compose.yml garantiza la reproducibilidad del entorno al describir las dependencias y configuraciones necesarias para su despliegue. Asimismo, los archivos README.md, pyproject.toml y poetry.lock documentan la instalación y manejo de librerías a través de Poetry, herramienta que contribuye a la portabilidad y estabilidad del entorno de desarrollo. Finalmente, los archivos de control .gitignore, .dvcignore y .dockerignore, junto con el script principal main.py y las carpetas .dvc/ y wandb_cache/, aseguran la trazabilidad del código, los datos y los experimentos, consolidando un entorno alineado con las prácticas recomendadas en MLOps y con los lineamientos profesionales establecidos por *Cookiecutter* ML Project.

4.2.2 Refactorización modular del código

En esta etapa, el proceso de desarrollo evoluciona desde un enfoque exploratorio basado en notebooks hacia una estructura modular ejecutada mediante scripts independientes. El código, que anteriormente se encontraba distribuido en *Jupyter Notebooks*, se organiza ahora en componentes autónomos dentro de la carpeta src/, lo que permite una mejor gestión del flujo de trabajo, la depuración y la trazabilidad del pipeline. Cada módulo cumple una función específica dentro del ciclo de vida del modelo, y su ejecución se controla a través del archivo run.py, el cual integra la lógica de limpieza, entrenamiento y evaluación de manera reproducible y automatizada. Este cambio responde a la necesidad de consolidar un entorno de desarrollo más robusto, mantenible y compatible con las buenas prácticas de MLOps.

El proceso de refactorización se fundamenta en la definición de funciones parametrizadas y controladas por línea de comandos mediante la biblioteca *argparse*, con un sistema de registro estructurado (*logging*) que permite monitorear la ejecución de cada etapa. Asimismo, se incorpora la herramienta W&B para el registro y versionamiento de artefactos, asegurando la trazabilidad de los datos procesados. La limpieza del *dataset* se gestiona a través de un mapa centralizado de reemplazos por columna, lo que estandariza los valores y reduce la redundancia de código. Además, se implementa un esquema de imputación de valores nulos basado en la moda de cada variable, optimizando el rendimiento y evitando la pérdida de información. El resultado final se registra como un nuevo artefacto versionado, garantizando la consistencia del flujo de datos entre etapas. En conjunto, esta refactorización fortalece el modularidad del sistema y facilita la integración futura de los componentes dentro de un pipeline automatizado en *Scikit-Learn*, cumpliendo así con los principios de eficiencia, legibilidad y reproducibilidad establecidos en la metodología MLOps.

4.2.3 Automatización del pipeline de modelado.

El flujo inicia con la ingesta del conjunto de datos *student_entry_performance.csv*, el cual se carga, versiona y almacena como artefacto dentro de W&B. Posteriormente, se ejecuta la limpieza de datos, donde se aplican transformaciones y normalizaciones que derivan en un conjunto depurado *(clean_data.artifact*), seguido de una etapa de validación que verifica la consistencia estructural y semántica de las variables. La segregación de datos divide el conjunto original en particiones de entrenamiento y prueba (*train_data.artifact y test_data.artifact*), garantizando un control de versiones que facilita la auditoría y la replicación de experimentos.

En las fases siguientes, el módulo de entrenamiento ejecuta el modelo en un entorno reproducible configurado con Hydra, *MLflow* y Optuna, generando artefactos de salida como el mapeo de etiquetas (*label_mapping.pkl*) y el modelo serializado (model.pkl). Finalmente, la etapa de prueba evalúa el desempeño del modelo y registra las métricas de rendimiento dentro del tablero de seguimiento de W&B, lo que permite comparar ejecuciones y analizar resultados de manera centralizada.

La integración de herramientas como *Jupyter*, W&B y *MLflow* conforma un ecosistema que combina el análisis exploratorio con la gestión automatizada de artefactos y modelos. Este enfoque garantiza coherencia, transparencia y control a lo largo del pipeline, además de sentar las bases para incorporar en fases posteriores mecanismos de despliegue continuo y monitoreo en producción que consoliden la madurez del sistema MLOps implementado.

4.2.4 Registro y gestión de experimentos.

Para gestionar los experimentos, usamos la plataforma Weights and Biases. Aprovechamos de su herramienta de línea de comandos ***wandb*** para ejecutar el modelo localmente desde una computadora.

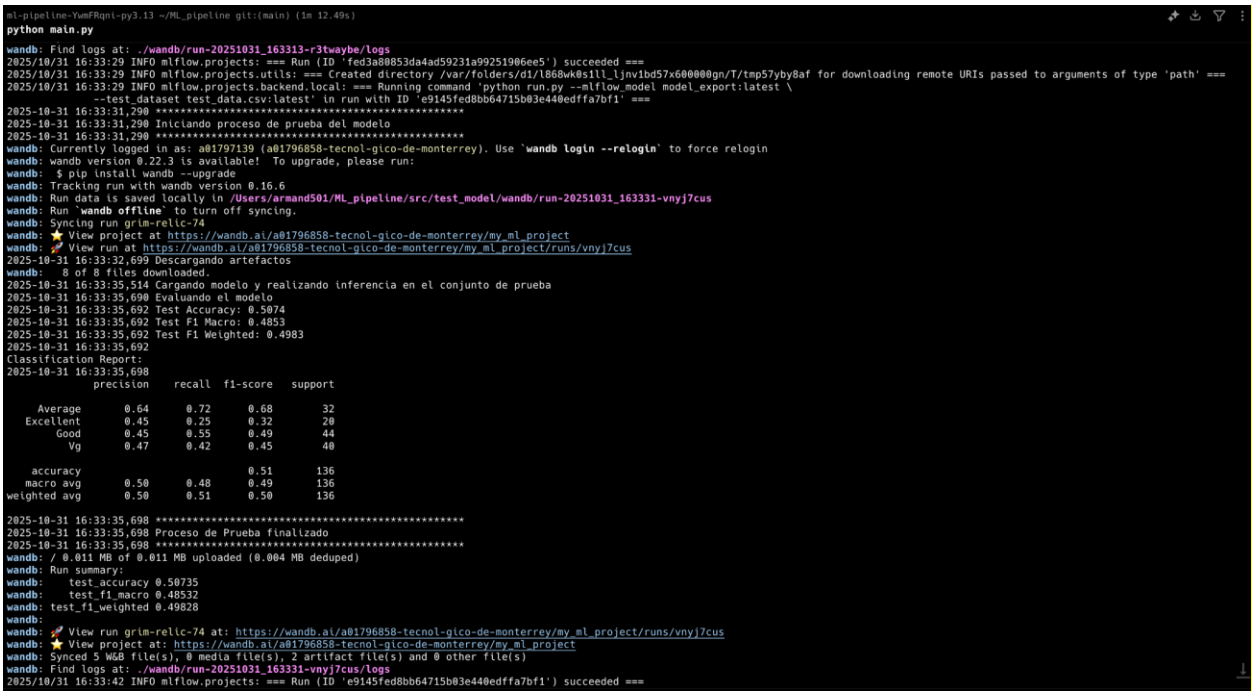

Ilustración 2 Herramienta wandb Para Ejecutar Modelo Localmente

W&B nos permite registrar los hiperparámetros, la configuración del experimento, versiones de datos, versiones de código, y revisarlos luego desde el dashboard. Por ejemplo, al iniciar un run con *wandb.init(...)* se puede asignar valores a wandb.config como *learning rate*, *batch size*, versión del dataset, nombre del modelo y estos quedan registrados.

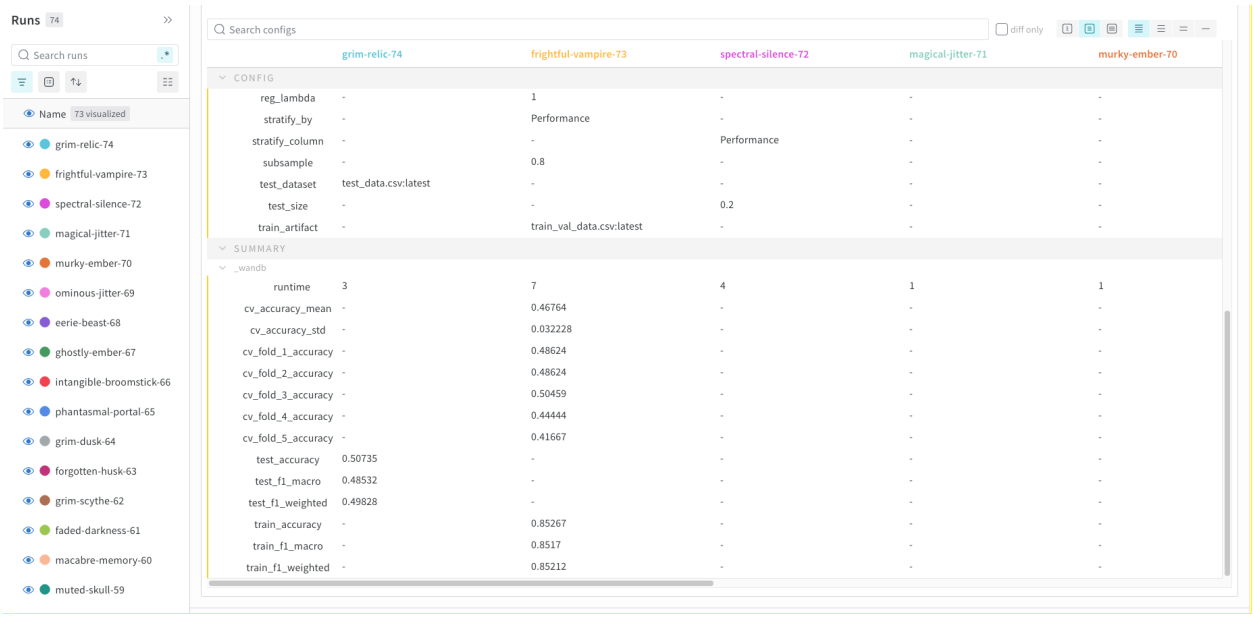

Ilustración 3 Dashboard de W&B

En el dashboard se ve la tabla de runs junto con sus configuraciones, de modo que puedes filtrar o agrupar según uno o varios parámetros. También se pueden guardar versiones de código fuente o vincular el commit Git, de modo que queda claro qué variante del código produjo los resultados vistos. Esta funcionalidad nos ayuda a responder preguntas como “¿qué parámetros se utilizó en esta ejecución que produjo ese resultado?”.

Se pueden visualizar gráficas interactivas de estas métricas a lo largo del tiempo, lo que permite ver la evolución de entrenamiento o de validación de test.

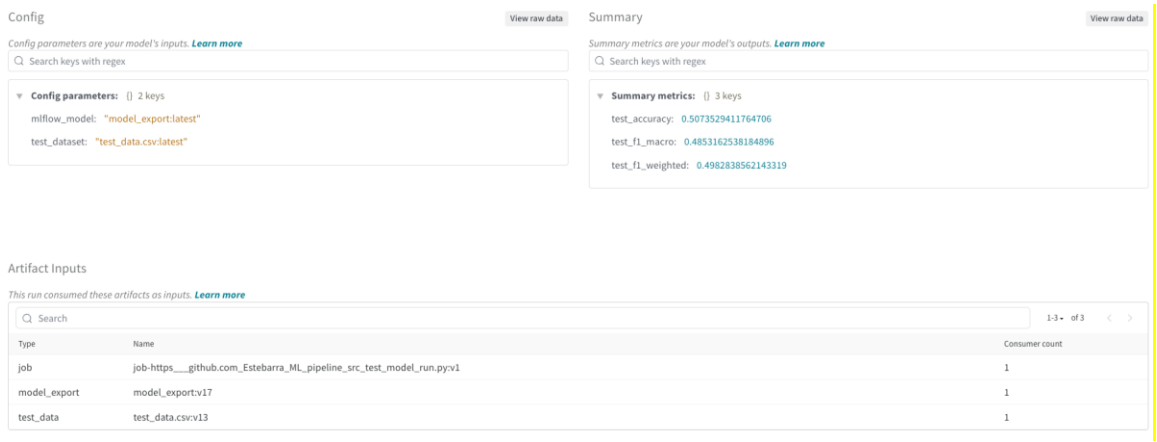

Ilustración 4 Dashboard de Configuración de W&B

W&B muestra una tabla dinámica de ejecuciones. Ahí podemos monitorear:

- Estado del run (completado, corriendo, detenido).
- Usuario y proyecto asociado.
- Tiempo de ejecución (*runtime*).
- *Dataset* y artefactos usados.
- Hiperparámetros configurados (*gamma, subsample, train_artifact, etc.*).
- Columnas personalizadas que se pueden añadir o filtrar (por ejemplo *test_accuracy, f1_macro*).

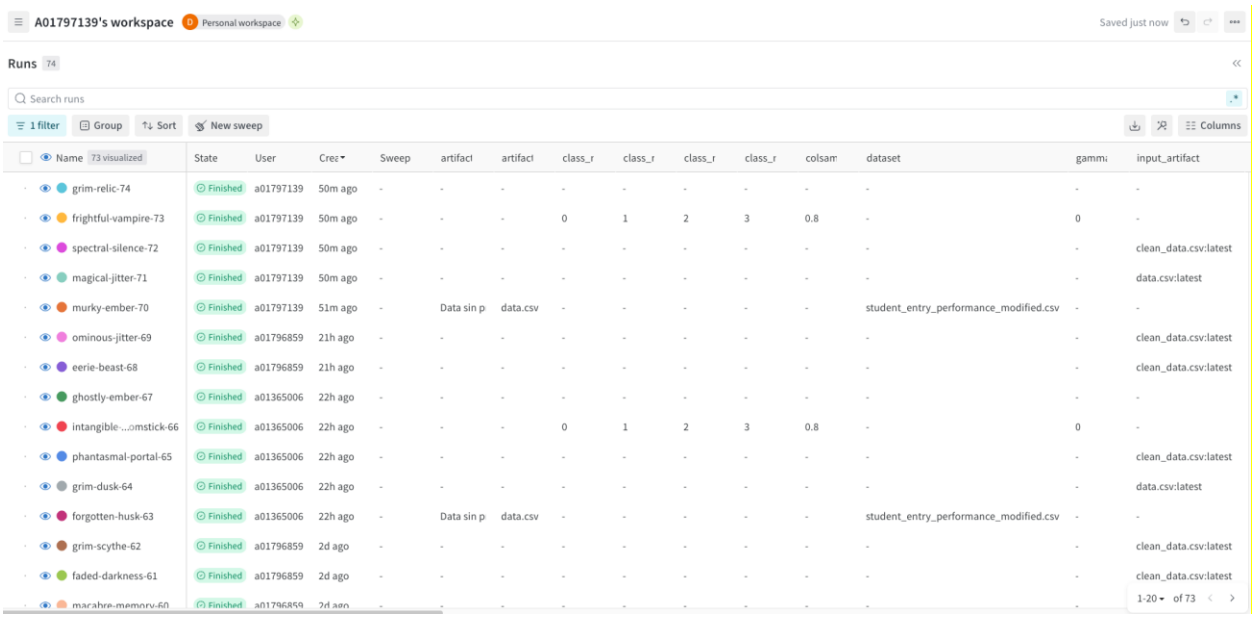

Ilustración 5 Dashboard Dinámico de Ejecuciones de W&B

Esto permite comparar ejecuciones de modelos diferentes o de una misma arquitectura bajo distintas condiciones, como diferentes *datasets* o parámetros.

W&B tiene un sistema de artefactos ("*Artifacts*") que permite versionar *datasets*, modelos exportados, salidas de la ejecución, etc. Esto es particularmente útil en un entorno de producción o de colaboración, donde varios miembros del equipo modifican datos y modelos y puedes llegar a necesitar saber exactamente "qué versión se utilizó".

Para nuestro caso, esta funcionalidad puede integrarse para asegurar trazabilidad completa.

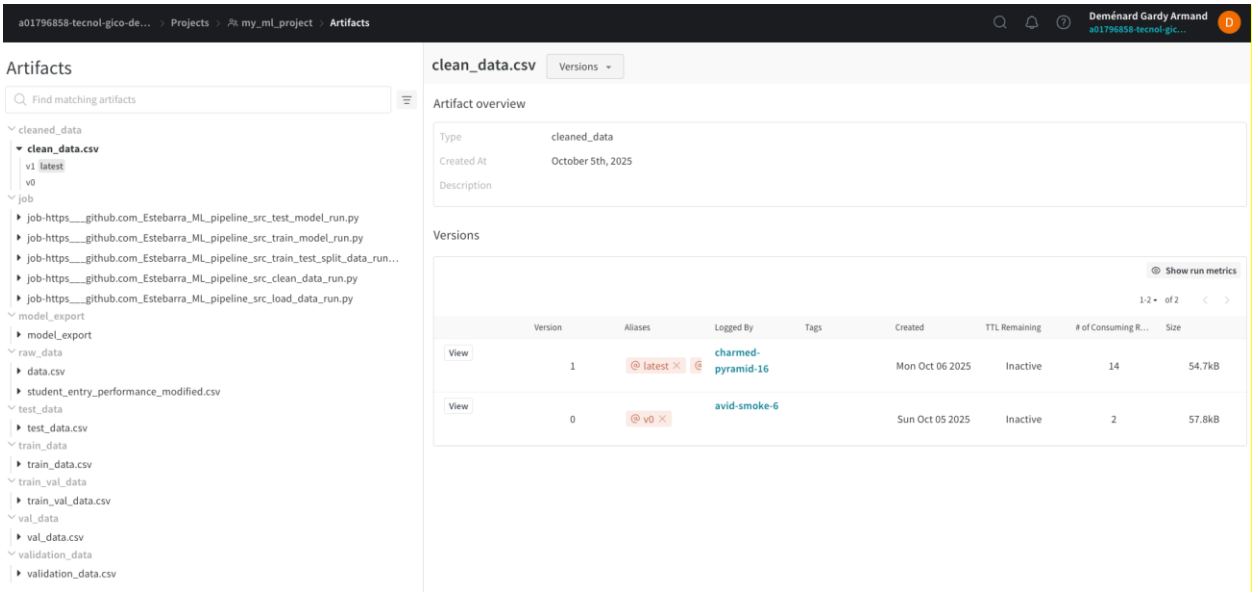

Ilustración 6 Dashboard de Artefactos de W&B

W&B no solo registra métricas de *Machine Learning* sino también información del entorno de ejecución: hardware, recursos usados, versiones de librerías, *CPU/GPU*, etc. Esto ayuda cuando los resultados podrían depender del hardware o del entorno. W&B automáticamente realiza un *log* de información del sistema (como número de *CPUs*, *GPU*, versión de Python) cuando iniciamos un run.

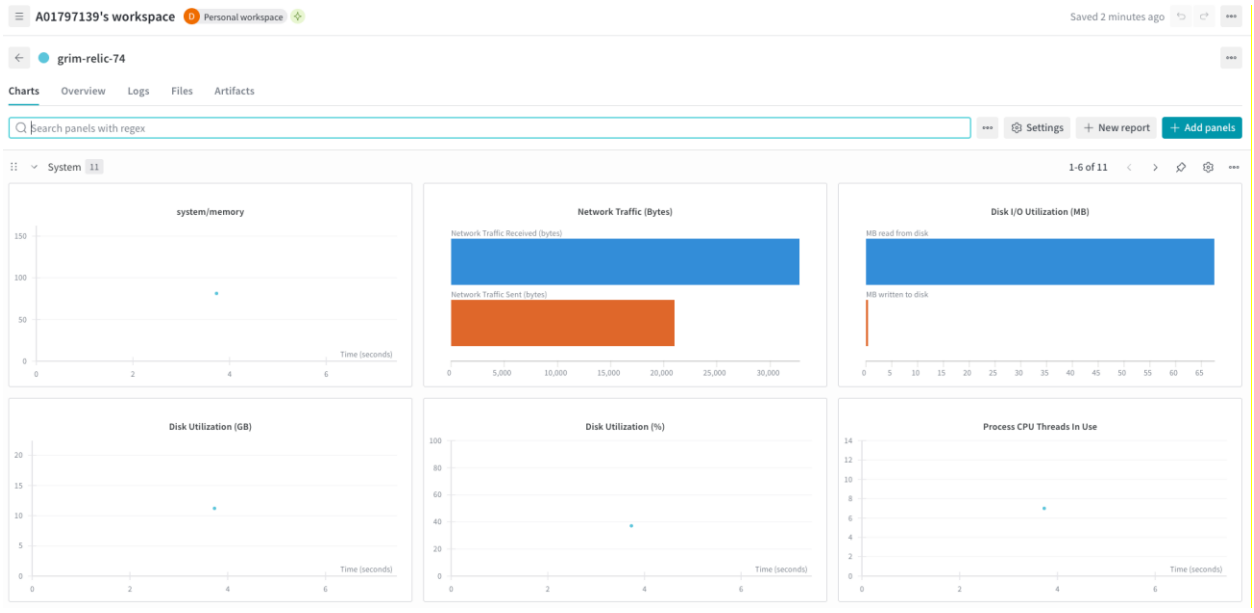

Ilustración 7 Dashboard de Información del Entorno de Ejecución

Capítulo 5: Resultados (Parcial)

Como punto inicial para la exposición de los resultados, se presenta nuevamente el ML Canvas formulado durante la Fase 1: Análisis y diseño del modelo predictivo, el cual resume los componentes esenciales del proyecto y su correspondencia con los objetivos definidos. Esta herramienta facilitó la organización integral de los requerimientos técnicos, las fuentes de datos, los indicadores de evaluación y la propuesta de valor del sistema desarrollado. A continuación, se muestra la tabla que sintetiza este análisis, donde se establecen las necesidades detectadas, los elementos metodológicos considerados y los beneficios proyectados de la aplicación del modelo predictivo.

Tabla 3 Requerimientos ML Canvas Parte 1

TAREA DE PREDICCIÓN	DECISIONES	PROPUESTA DE VALOR	RECOPILACIÓN DE DATOS	FUENTES DE DATOS
Entrada: Variables demográficas, académicas y socioeconómicas. **Salida (Target): Performance** (Excellent, Good, Average, Poor en Nivel de desempeño esperado). **Tipo de tarea:** Clasificación supervisada. **Algoritmos candidatos:** • Regresión Logística • SVM • Gradient Boosting (XGBoost, LightGBM) • Regresión Logística Ordinal	**Identificar factores sociales, educativos y personales que afectan en el desempeño de los estudiantes.** **Diseñar tutorías específicas de acuerdo a las necesidades de cada estudiante.** **La predicción del rendimiento también pueden ser:** • Apoyar políticas de becas o refuerzos educativos. • Ayudar a identificar candidatos en riesgo académico. • Guiar estrategias de admisión o intervención temprana.	**La propuesta de valor de este enfoque ofrece la posibilidad de generar información para diferentes actores del sistema educativo. Para las instituciones, el modelo permitirá identificar factores éxito académico y diseñar programas de apoyo. Para los estudiantes y sus familias, ofrecerá una visión más clara sobre el impacto real de invertir en programas de coaching, ayudando a una mejor toma de decisiones educativas. Finalmente, a nivel social, este análisis contribuirá a visibilizar posibles desigualdades en el acceso a recursos de formación, fomentando el diseño de políticas más justas y equitativas.**	**Los datos recolectados a partir de registros académicos y formularios de inscripción.** **Los datos incluyen variables demográficas (género, casta), educativas (porcentajes de clase X y XII, tipo de institución), y familiares (ocupación de los padres).**	**Los datos son obtenidos del dataset público "Student Performance on an Entrance Examination".** **Dataset recopilado por el Prof. Jiten Hazarika, que contiene información académica y socioeconómica de los aspirantes a universidades médicas de Assam.**

Tabla 4 Requerimientos ML Canvas Parte 2

SIMULACIÓN DE IMPACTO	HACIENDO PREDICCIONES	CONSTRUYENDO MODELOS	CARACTERÍSTICAS
Mayor precisión de predicción y de transparencia **Métricas de evaluación: Accuracy, Precision, Recall, F1-Score, Confusion Matrix**	**Las predicciones se hacen en bloque al finalizar las aplicaciones del examen de admisión.** **Se pueden hacer predicciones en tiempo real para calcular la probabilidad de aprobación de los estudiantes.**	**Construir un solo modelo de clasificación supervisada que prediga el nivel de rendimiento (Performance) de los estudiantes en categorías como *Excellent*, *Good*, Poor, Average.** • Entrenamiento inicial con el dataset histórico. • Actualización anual con nuevos datos de ingreso. • Frecuencia: una vez por ciclo académico (anual). • Tiempo estimado para el modelado: 1-2 semanas para limpieza, entrenamiento y validación.	**Entradas que alimentan el modelo:** • Gender • Caste • Coaching (ninguno, dentro o fuera de Assam) • Class_ten_education (tipo de institución) • Twelve_education (tipo de institución) • Medium (idioma de instrucción) • Class_X_Percentage • Class_XII_Percentage • Father_occupation • Mother_occupation

Tabla 5 Requerimientos ML Canvas Parte 3

	MONITOREO			
	Seguimiento de métricas técnicas (Accuracy, Precision, Recall, F1-Score, Confusion Matrix)	**Monitoreo de las diferencias de desempeño entre grupos(género, casta, desempeño previo, etc.)**	**Detectar si hay cambios en los patrones de los datos.**	

Durante la etapa de manipulación y preparación de datos, se ejecutó un proceso de limpieza integral orientado a eliminar inconsistencias, estandarizar valores y garantizar la uniformidad semántica del conjunto de datos. El script implementado realizó una serie de transformaciones columna por columna, corrigiendo errores tipográficos, variaciones de formato y discrepancias en el uso de mayúsculas y minúsculas.

En la variable Performance, se unificaron las distintas formas de escritura de las categorías, reemplazando variantes como *gOOD*, *aVERAGE*, *vG* y *eXCELLENT* por las etiquetas correctas Good, Average, Vg y Excellent respectivamente. De manera similar, se normalizó la variable *Gender* estandarizando *MALE* y *FEMALE* como *male* y *female*,

mientras que en *Caste* se corrigieron las abreviaciones y errores de capitalización para mantener la coherencia entre categorías (ST, OBC, General, SC).

El mismo procedimiento se aplicó a las variables *coaching, time, Class_ten_education, twelve_education, medium, Father_occupation y Mother_occupation*, homogeneizando las entradas para eliminar inconsistencias de escritura y mejorar la legibilidad de los datos. Asimismo, se aplicó un tratamiento de valores nulos mediante la imputación de la moda (valor más frecuente) en cada variable, con el fin de preservar la integridad de las observaciones sin introducir sesgos estadísticos.

Finalmente, se eliminó la columna *mixed_type_col*, identificada como irrelevante para el análisis por contener valores de tipo mixto que afectaban la consistencia del *dataset*. Tras estas transformaciones, el conjunto de datos resultante presentó una estructura más limpia, coherente y preparada para las etapas de modelado.

Todas las operaciones fueron registradas y versionadas a través de W&B, permitiendo conservar un historial completo de los cambios realizados sobre el *dataset* y asegurando su trazabilidad en el flujo MLOps.

Durante la etapa de exploración y preprocesamiento de datos, se llevó a cabo un examen detallado con el propósito de conocer la composición y las propiedades generales del conjunto de información. En primera instancia, se procedió a cargar el archivo de datos modificado, comprobando su correcta lectura y la integridad de los registros mediante la visualización inicial y final de las filas contenidas en el *dataset*. Posteriormente, se identificó la cantidad total de observaciones y variables, además de los tipos de datos correspondientes a cada una, diferenciando entre aquellas de carácter numérico y las de tipo categórico.

De forma complementaria, se efectuó un análisis descriptivo inicial destinado a determinar el número de valores únicos, las categorías más frecuentes y la presencia de posibles inconsistencias en la distribución de los atributos. Para cada variable se calcularon las frecuencias absolutas, lo que permitió identificar la proporción de estudiantes en los distintos niveles de desempeño y grupos sociodemográficos.

Después de aplicar la normalización de etiquetas en la variable *Performance*, los valores fueron unificados para eliminar inconsistencias tipográficas y garantizar uniformidad semántica. Tras el proceso de limpieza, las categorías finales quedaron consolidadas en *Excellent, Very Good (Vg), Good y Average*, con la siguiente distribución definitiva:

Tabla 6 Frecuencia de la Categoria de *Performance*

Categoría de rendimiento	Frecuencia	Porcentaje
Good	213	31.39 %
Very Good (Vg)	200	29.46 %
Average	158	23.27 %
Excellent	108	15.89 %
Total	679	100 %

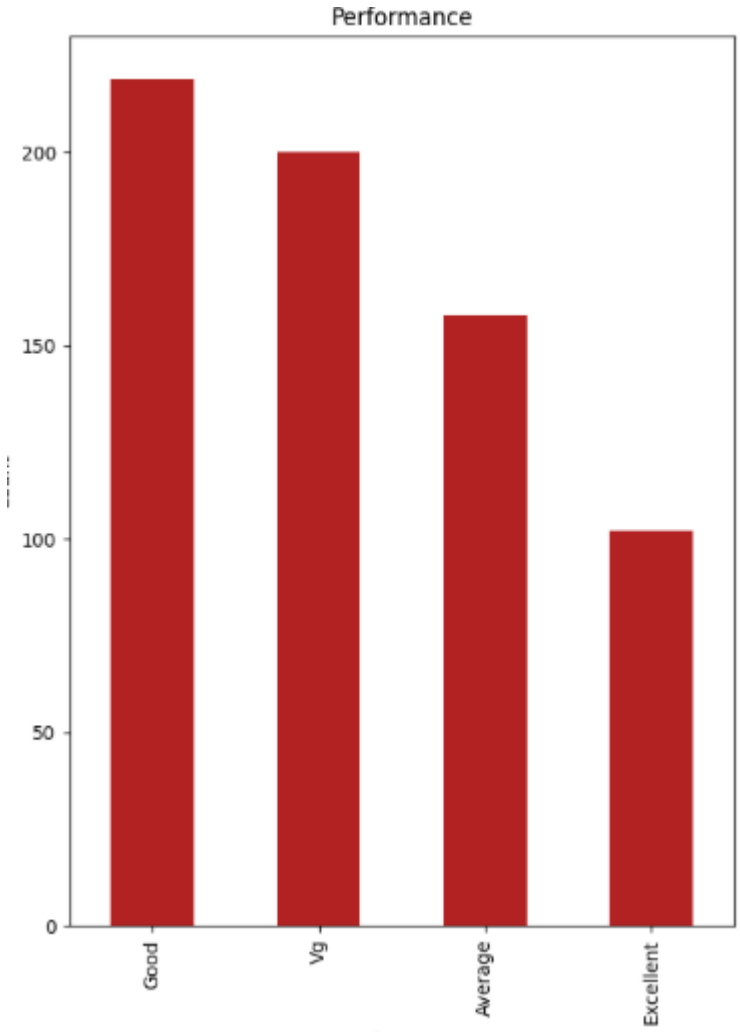

Ilustración 8 Grafico de Frecuencia de la Categoría de *Performance*

Después del proceso de corrección, el conjunto de datos adquirió una estructura uniforme y consistente, libre de discrepancias en la escritura y de valores duplicados que pudieran comprometer el desempeño del modelo predictivo. Se observa que la mayor proporción de estudiantes se agrupa en los niveles *Good y Very Good*, los cuales, en

conjunto, superan el 60% de la muestra analizada, reflejando una tendencia general hacia un rendimiento académico medio-alto.

El modelo XGBoost presentó un desempeño general moderado, alcanzando una *Accuracy* de 0.4741. El análisis por clase evidenció un comportamiento desigual entre categorías, donde la clase *Average* obtuvo los mejores resultados con una precisión de 0.86 y un *F1-Score* de 0.80, mientras que las categorías *Good, Very Good y Excellent* mostraron valores considerablemente menores, con *F1-Scores* de 0.47, 0.33 y 0.29, respectivamente. Esta tendencia sugiere que el modelo logró identificar con mayor efectividad a los estudiantes con desempeño promedio, pero tuvo dificultades para distinguir entre niveles de rendimiento superior, posiblemente debido al desbalance en la distribución de clases.

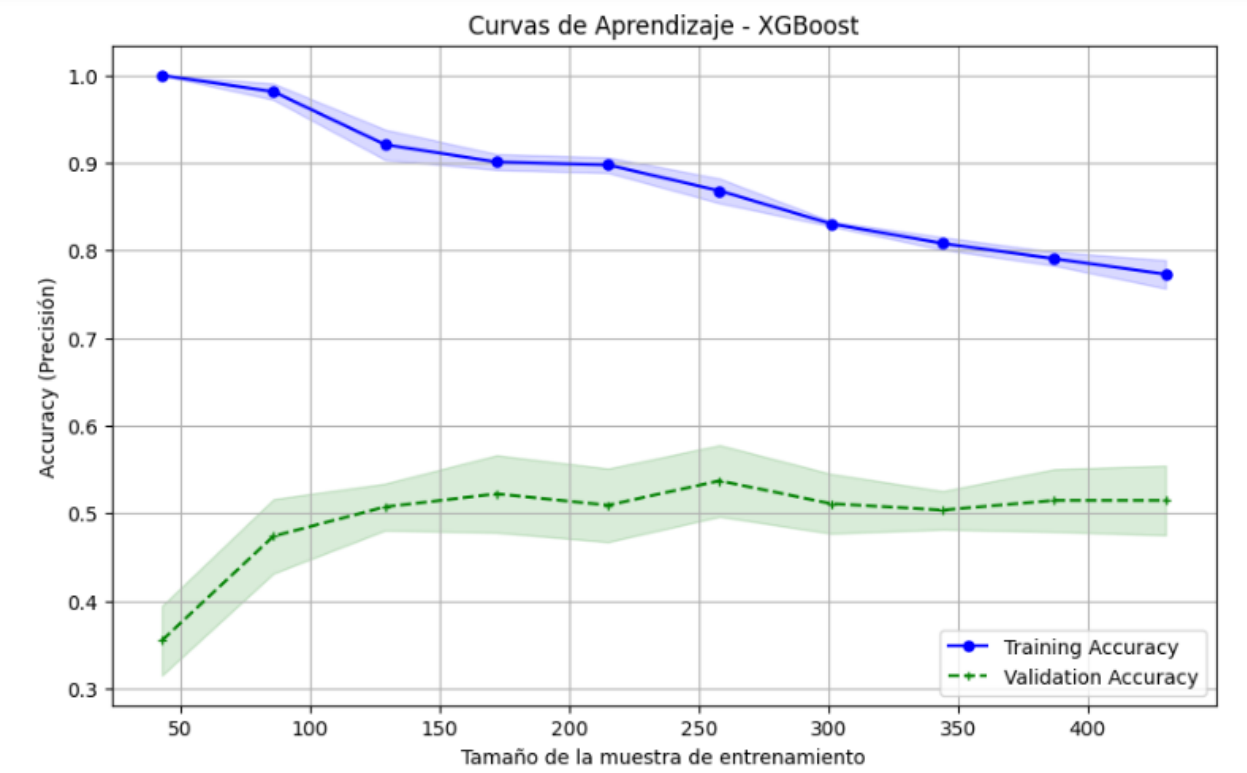

Ilustración 9 Curvas de Aprendizaje - XGBoost

En cuanto a las curvas de aprendizaje, se observa una clara brecha entre el desempeño de entrenamiento y el de validación, lo que indica un grado de sobreajuste (*overfitting*). El modelo alcanzó una alta precisión en el conjunto de entrenamiento, pero la *Accuracy* de validación permaneció cercana al 50%, sin mostrar convergencia significativa. Este comportamiento revela que, aunque XGBoost aprendió patrones específicos del conjunto de entrenamiento, su capacidad de generalización hacia nuevos datos fue limitada

El modelo de Regresión Logística mostró un rendimiento ligeramente superior al de XGBoost, alcanzando una *Accuracy* de 0.5185. En el análisis por clase, se observa que la categoría *Average* continúa siendo la mejor representada, con una precisión de 0.82 y un *F1-Score* de 0.77, mientras que las categorías *Good y Very Good* presentan un comportamiento intermedio, con *F1-Scores* de 0.51 y 0.47, respectivamente. Por su parte, la clase *Excellent* mantiene un bajo desempeño (*F1-Score* = 0.20), lo que refleja la dificultad del modelo para distinguir correctamente los casos de mayor rendimiento académico, probablemente debido al desbalance de la muestra y a la similitud entre los atributos que caracterizan a los niveles altos de desempeño.

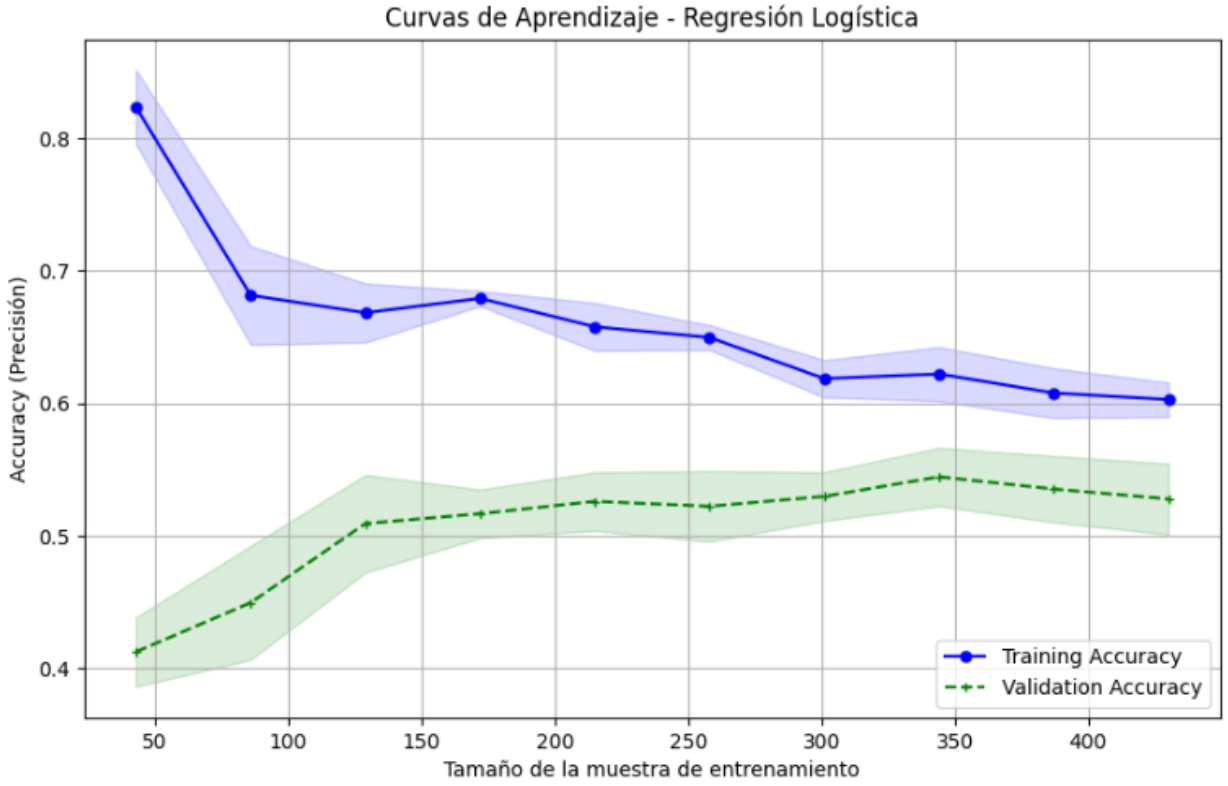

Ilustración 10 Curvas de Aprendizaje - Regresión Logística

Las curvas de aprendizaje evidencian un patrón de comportamiento consistente con un ligero sobreajuste. A medida que el tamaño del conjunto de entrenamiento aumenta, la precisión de entrenamiento desciende progresivamente, estabilizándose cerca de 0.6, mientras que la precisión de validación se mantiene alrededor de 0.5 sin converger plenamente. Esta diferencia indica que el modelo logra aprender patrones generales de los datos, pero aún enfrenta limitaciones para generalizar con mayor precisión.

El modelo SVM obtuvo una *Accuracy* de 0.4963, mostrando un desempeño similar al de los modelos previos, aunque con un comportamiento particular en las métricas por clase. La categoría *Average* nuevamente se posicionó como la mejor representada, con una

precisión de 0.95 y un F1-Score de 0.78, mientras que las clases Good y Very Good mantuvieron niveles moderados de desempeño, con *F1-Scores* de 0.48 y 0.45, respectivamente. Por el contrario, la clase *Excellent* mostró un *F1-Score* de apenas 0.10, lo que evidencia una baja capacidad del modelo para identificar correctamente los casos de mayor rendimiento.

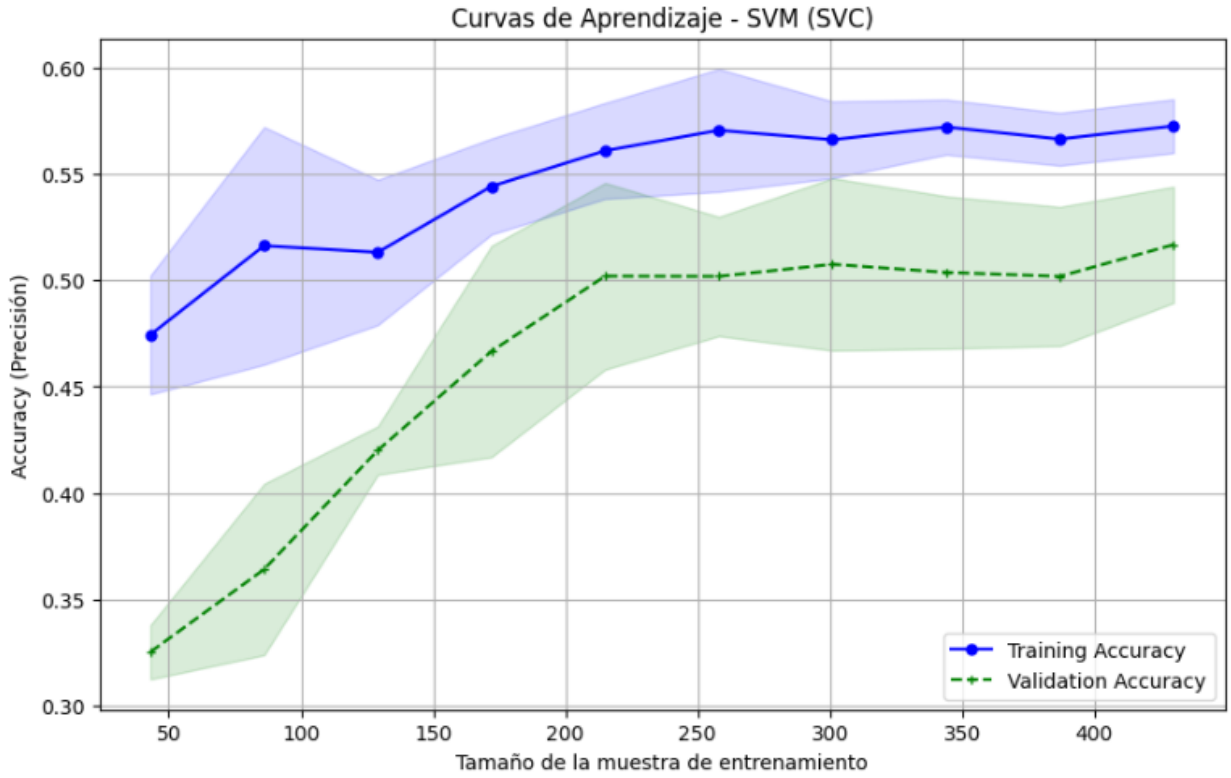

Ilustración 11 Curvas de Aprendizaje - SVM

En relación con las curvas de aprendizaje, se observa una tendencia ascendente tanto en la precisión de entrenamiento como en la de validación, lo que indica que el modelo continúa mejorando su ajuste conforme aumenta el tamaño de la muestra. Sin embargo, la brecha persistente entre ambas curvas refleja la existencia de ligero sobreajuste, aunque con una estabilidad superior respecto a los modelos anteriores. Este comportamiento sugiere que SVM presenta una buena capacidad de aprendizaje progresivo y una tendencia hacia la convergencia, lo que lo convierte en un modelo prometedor para etapas posteriores.

El modelo de Regresión Logística Ordinal obtuvo una *Accuracy* global de 0.5111, posicionándose como uno de los algoritmos con mejor equilibrio general entre las clases. La categoría *Average* alcanzó una precisión perfecta (1.00) y un *F1-Score* de 0.77, consolidándose como la más estable del conjunto, mientras que *Very Good* presentó un *F1-Score* de 0.52, evidenciando un avance en la identificación de niveles de desempeño medio-alto. En contraste, las categorías *Good y Excellent* registraron desempeños más

modestos (*F1-Scores* de 0.44 y 0.16, respectivamente), lo que indica que, si bien el modelo es capaz de capturar relaciones ordinales entre las clases, aún enfrenta limitaciones para diferenciar los extremos del espectro de rendimiento

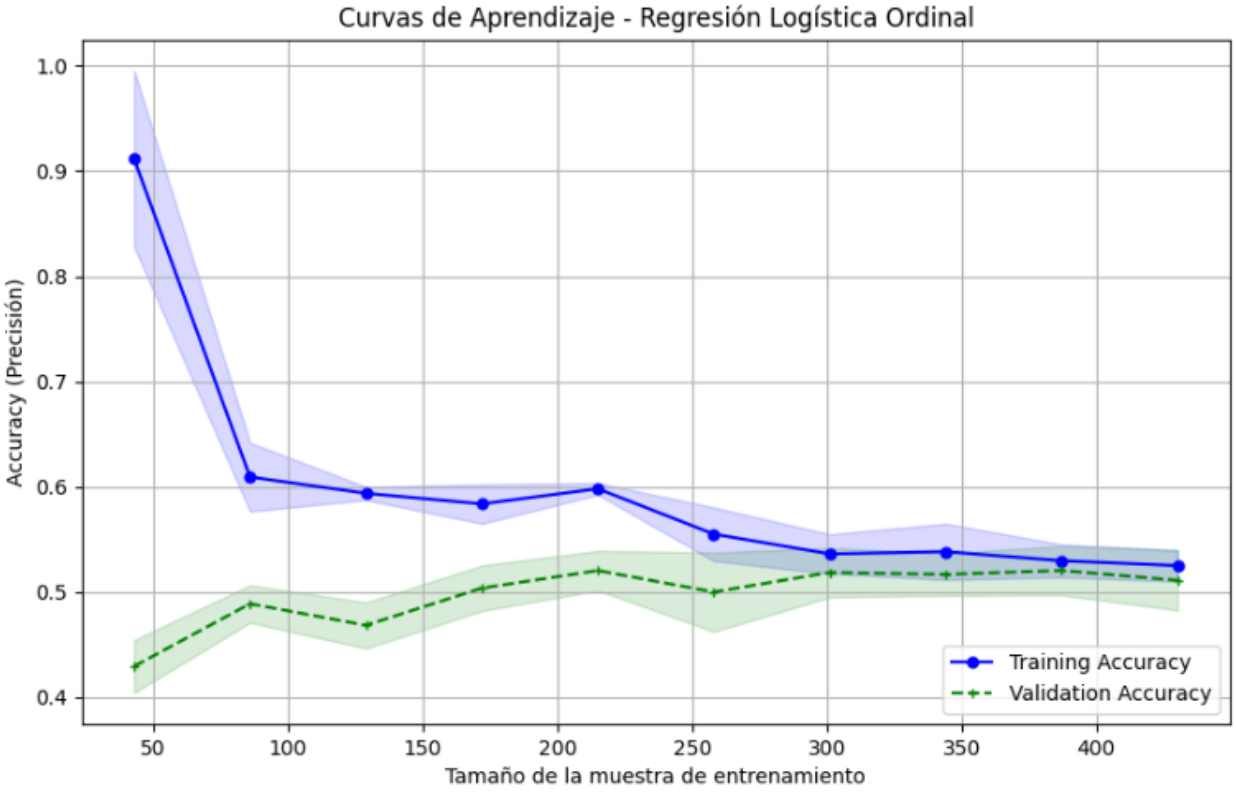

Ilustración 12 Curvas de Aprendizaje - Regresión Logística Ordinal

Las curvas de aprendizaje evidencian una tendencia progresiva hacia la estabilidad y la convergencia entre la precisión del entrenamiento y la de validación, lo que indica un comportamiento más equilibrado y con menor grado de sobreajuste en comparación con los modelos previos. Conforme se incrementa el tamaño de la muestra, la precisión de validación se mantiene constante en torno al 50%, mientras que la de entrenamiento desciende de manera gradual hasta aproximarse a ese mismo nivel, reflejando así una mejor capacidad de generalización y un ajuste más consistente del modelo frente a nuevos datos.

La creación del repositorio en GitHub permitió centralizar todos los elementos del proyecto y mantener un control eficiente sobre las versiones del código, los notebooks y los archivos de configuración. Esta herramienta facilitó la colaboración entre los integrantes del equipo, promoviendo un registro detallado de los cambios y una documentación continua de los avances durante cada etapa de la investigación. Su integración resultó fundamental para asegurar la reproducibilidad y coherencia del

desarrollo, consolidando un flujo de trabajo alineado con las buenas prácticas del ciclo de vida MLOps.

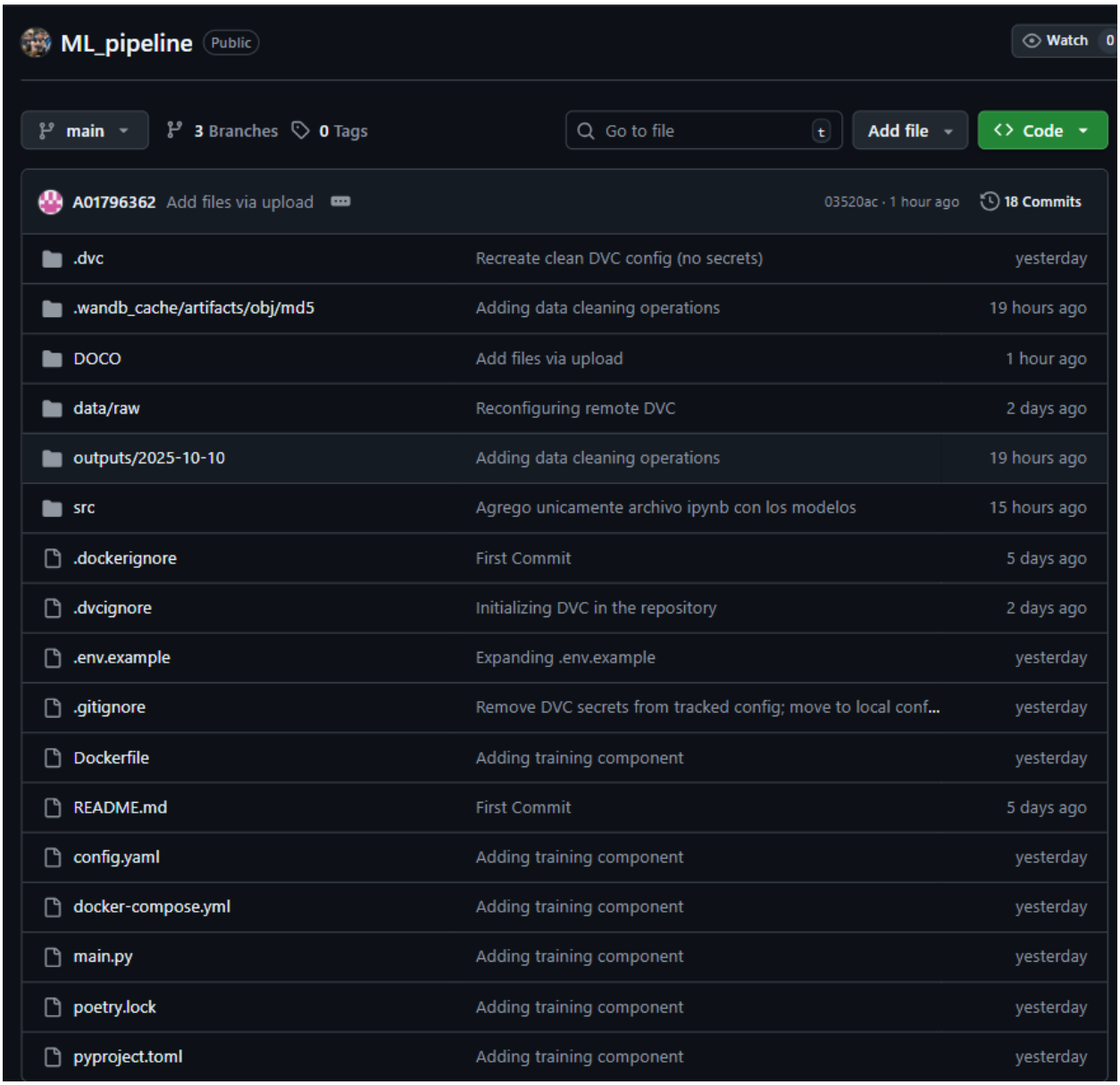

Ilustración 13 Repositorio de GitHub

La implementación del esquema *Cookiecutter* representa un paso clave en la consolidación de una estructura de trabajo organizada y coherente con las prácticas de ingeniería aplicadas al aprendizaje automático. El repositorio resultante muestra una arquitectura modular que distribuye de forma lógica los componentes del proyecto, incluyendo las carpetas data/, destinadas al almacenamiento de los conjuntos de datos; src/, donde se concentra el código fuente de los módulos principales; y outputs/, empleada para registrar los artefactos, métricas y resultados del pipeline. De igual forma,

se integran archivos de configuración como config.yaml y .env.example, así como dependencias gestionadas mediante Poetry a través de los archivos pyproject.toml y poetry.lock, lo que asegura la estabilidad del entorno virtual. La presencia de herramientas como Docker refuerza la reproducibilidad del sistema y permite ejecutar el proyecto en distintos entornos sin pérdida de consistencia, garantizando un control total sobre los datos, el código y los experimentos. En conjunto, esta estructura demuestra la correcta adopción del enfoque MLOps, al unificar orden, trazabilidad y escalabilidad dentro de un mismo esquema de desarrollo.

La plantilla implementada permitió estandarizar la creación de nuevos componentes dentro del pipeline de MLFlow, garantizando la consistencia estructural entre los distintos módulos del proyecto. Esta herramienta automatiza la generación de carpetas, scripts y archivos de configuración necesarios para cada etapa del flujo, solicitando al usuario parámetros como el nombre del paso, su descripción y los argumentos requeridos. Con ello, cada componente se crea bajo un esquema uniforme que incluye el script principal, el archivo MLproject y la configuración de entorno en conda.yml, listos para su integración en el pipeline general.

La plantilla se encuentra ubicada en el directorio /ML_pipeline/cookie-step/, donde se centralizan los archivos que permiten generar de forma automatizada nuevos componentes del pipeline de MLFlow. Dentro de este repositorio se incluyen tres elementos fundamentales: el archivo *cookiecutter.json*, que define las variables solicitadas al usuario como el nombre del paso, su descripción y los parámetros requeridos; la carpeta {{*cookiecutter.step_name*}}/, que actúa como estructura base para el nuevo módulo y contiene los archivos esenciales del flujo, entre ellos *MLproject* y el script principal en formato .py; y finalmente el archivo *README*.md, que proporciona una guía de uso detallada sobre la plantilla y su correcta implementación.

El funcionamiento de esta herramienta permite crear de manera automatizada la estructura básica de un nuevo paso del pipeline, ya sea para procesos de carga, limpieza, entrenamiento o evaluación, manteniendo coherencia con los estándares definidos por MLFlow. Cada ejecución genera un entorno listo para integrarse en el flujo principal, con un script preconfigurado, los archivos *MLproject* y *conda*.yml adaptables al nuevo componente, y un esquema de parámetros configurables desde la línea de comandos. Para iniciar la plantilla, el usuario ejecuta el comando *cookiecutter* [ruta_a_cookie-step] -o [directorio de destino], tras lo cual el sistema solicita la información necesaria y genera automáticamente una carpeta con la estructura completa del nuevo paso.

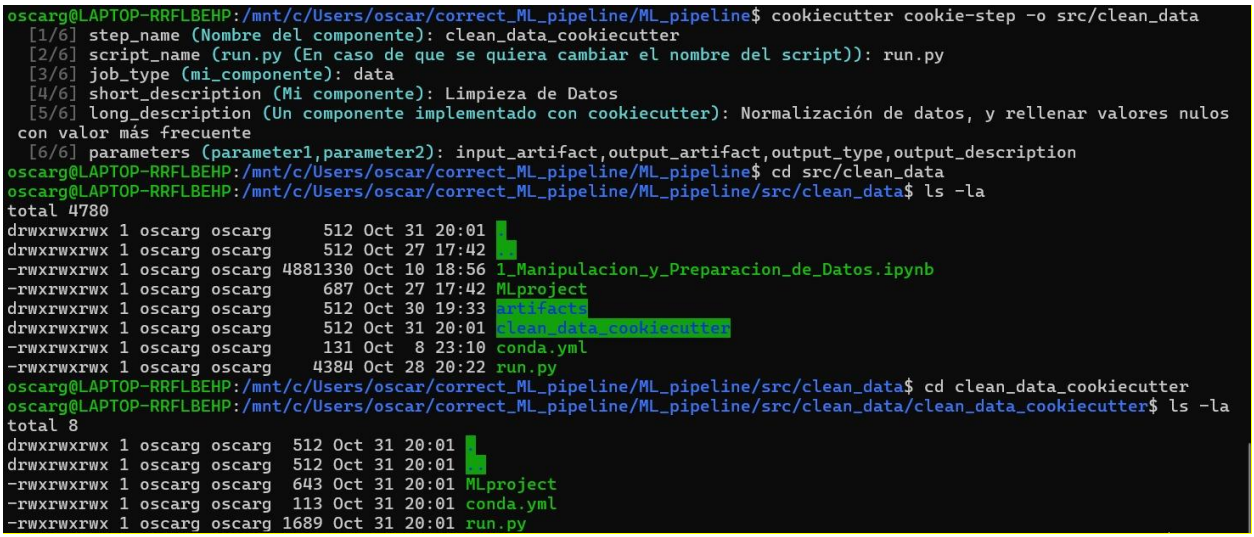

Ilustración 14 Ejecución de plantilla *Cookiecutter* para Experimentos

Esta plantilla no solo simplifica la incorporación de nuevos procesos como limpieza, validación o entrenamiento, sino que también favorece la mantenibilidad y la escalabilidad del sistema, al reducir la dependencia de configuraciones manuales y minimizar errores de estructura. En conjunto, el uso de *Cookiecutter* como generador de pasos modulares contribuye a consolidar una práctica de desarrollo reproducible y coherente con los estándares profesionales de ingeniería en entornos MLOps.

Los resultados de la refactorización evidencian una transición completa del trabajo exploratorio en notebooks hacia una estructura modular y ejecutable, diseñada para mejorar la trazabilidad y reproducibilidad del proyecto. Cada componente del pipeline se desarrolló como un script independiente, controlado desde la línea de comandos e instrumentado con un sistema de registro uniforme mediante *logging*. En esta nueva organización, el script load_data se encarga de cargar los conjuntos de datos locales en W&Bcomo artefactos versionados; clean_data realiza la normalización de categorías a través de un mapa centralizado de reemplazos, imputa los valores nulos por moda y elimina columnas irrelevantes antes de registrar el dataset limpio; train_split_data divide los datos en conjuntos de entrenamiento y prueba con estratificación y semillas fijas para asegurar consistencia; train_model integra un pipeline completo compuesto por el preprocesador y el modelo XGBoost, ejecuta validación cruzada estratificada y exporta el modelo en formato MLflow, incluyendo su firma, ejemplo de entrada y el mapeo de etiquetas; finalmente, test_data carga el modelo registrado y evalúa su rendimiento en el conjunto de prueba, generando métricas macro, ponderadas y un reporte de clasificación completo.

Durante la ejecución, cada módulo registra sus configuraciones y resultados directamente en W&B, lo que permite dar seguimiento al desempeño y reproducir los experimentos de manera controlada. En la etapa de entrenamiento, se documentan métricas de validación cruzada (*cv_accuracy_mean, cv_accuracy_std, cv_fold_scores*) junto con el rendimiento en entrenamiento (*train_accuracy, train_f1_macro, train_f1_weighted*), mientras que en la fase de prueba se almacenan las métricas finales (*test_accuracy, test_f1_macro, test_f1_weighted*) junto con el reporte de desempeño por clase. Todos los artefactos, *datasets* procesados, *splits*, modelo entrenado y mapeos de etiquetas, se versionan en W&B, consolidando un flujo de trabajo coherente y completamente trazable.

La refactorización fortalece la mantenibilidad del código, reduce la redundancia en los procesos de limpieza y entrenamiento, y eleva la calidad del pipeline mediante una estructura modular, reproducible y estandarizada. Además, se asegura la compatibilidad entre W&B y MLflow, lo que facilita la gestión del modelo y su futura implementación dentro de un entorno MLOps completo.

A continuación, se muestran una serie de ilustraciones donde puede apreciarse el código refactorizado, no obstante, solo mostraran unas partes como ejemplificación, los archivos run.py para cada etapa se pueden encontrar en el repositorio dentro de la carpeta src.

```python
#!/usr/bin/env python
"""
Realiza operaciones básicas de limpieza en el conjunto de datos y guarda los
resultados en Weights & Biases.
"""
import os
import argparse
import logging
import pandas as pd
import numpy as np

import wandb


logging.basicConfig(level=logging.INFO, format="%(asctime)-15s %(message)s")
logger = logging.getLogger(__name__)

def go(args):
    """
    Implementación de funciones básicas de limpieza
    """
    logger.info("*"*50)
    logger.info("Iniciando proceso de limpieza")
    logger.info("*"*50)

    run = wandb.init(
        job_type="basic_cleaning"
    )
    run.config.update(args)

    artifact = run.use_artifact(args.input_artifact)
    artifact_path = artifact.file()
    df = pd.read_csv(artifact_path)

    # Definir un mapa de limpieza centralizado
    # Agrupar reemplazos de valores por columna
    cleaning_map = {
        'Performance': {'gOOD': 'Good', 'aVERAGE': 'Average', 'vG': 'Vg', 'eXCELLENT': 'Excellent'},
        'Gender': {'MALE': 'male', 'FEMALE': 'female'},
        'Caste': {'st': 'ST', 'obc': 'OBC', 'gENERAL': 'General', 'sc': 'SC'},
        'coaching': {'wa': 'WA', 'no': 'NO', 'oa': 'OA'},
        'time': {'one': 'ONE', 'two': 'TWO', 'three': 'THREE'},
        'Class_ten_education': {'seba': 'SEBA', 'cbse': 'CBSE', 'others': 'OTHERS'},
        'twelve_education': {'cbse': 'CBSE', 'ahsec': 'AHSEC'},
        'medium': {'english': 'ENGLISH', 'others': 'OTHERS', 'assamese': 'ASSAMESE'},
        'Class_ X_Percentage': {'eXCELLENT': 'Excellent', 'vG': 'Vg', 'gOOD': 'Good'},
        'Class_XII_Percentage': {'eXCELLENT': 'Excellent', 'vG': 'Vg', 'gOOD': 'Good'},
        'Father_occupation': {
            'business': 'BUSINESS', 'bank_official': 'BANK_OFFICIAL',
            'college_teacher': 'COLLEGE_TEACHER', 'others': 'OTHERS',
            'school_teacher': 'SCHOOL_TEACHER', 'cultivator': 'CULTIVATOR',
            'engineer': 'ENGINEER', 'doctor': 'DOCTOR'
```

Ilustración 15 Ejemplo de de Refactorización de Código a Archivo Run.py 1

```
        # Aplicar limpieza en un solo bucle
        for col, replacements in cleaning_map.items():
            # Encadenar .str.strip() y .replace()
            df[col] = df[col].str.strip().replace(replacements)

        # Reemplazar valores nulos genéricos
        df.replace(['', ' ', 'NAN'], np.nan, inplace=True)

        # Obtener la lista de columnas que acabamos de limpiar
        cols_to_impute = [col for col in cleaning_map.keys() if col in df.columns]

        # Calcular todas las modas necesarias de una vez
        # Usar .mode().loc[0] para obtener el primer valor (la moda)
        imputation_values = {col: df[col].mode().loc[0] for col in cols_to_impute}

        # Aplicar fillna UNA SOLA VEZ con el diccionario de modas
        df.fillna(imputation_values, inplace=True)

        # Eliminar columna no deseada
        col_to_drop = 'mixed_type_col'
        df.drop(col_to_drop, axis=1, inplace=True)

        # Guardar y registrar el artefacto de salida (sin cambios)
        filename = args.output_artifact
        df.to_csv(filename, index=False)

        logger.info(f"Forma del dataframe resultante: {df.shape}")

        artifact = wandb.Artifact(
            filename,
            type=args.output_type,
            description=args.output_description,
        )
        artifact.add_file(filename)

        run.log_artifact(artifact)
        artifact.wait()

        # Limpieza local
        os.remove(filename)

        logger.info("*" * 50)
        logger.info("Proceso de limpieza finalizado")
        logger.info("*" * 50)
```

Ilustración 16 Ejemplo de Refactorización de Código a Archivo Run.py 2

```python
if __name__ == "__main__":

    parser = argparse.ArgumentParser(
            description="Este componente realiza una limpieza básica de datos"
        )

    parser.add_argument(
        "--input_artifact",
        type=str,
        help="Archivo objetivo",
        required=True
    )

    parser.add_argument(
        "--output_artifact",
        type=str,
        help="Archivo de salida",
        required=True
    )

    parser.add_argument(
        "--output_type",
        type=str,
        help="Tipo de salida",
        required=True
    )

    parser.add_argument(
        "--output_description",
        type=str,
        help="Archivo procesado por la eliminación de valores atípicos",
        required=True
    )

    args = parser.parse_args()

    go(args)
```

Ilustración 17 Ejemplo de Refactorización de Código a Archivo Run.py 3

El pipeline automatizado implementado permitió consolidar un flujo integral de modelado bajo principios MLOps, donde cada etapa del proceso se ejecuta de forma controlada y reproducible. La secuencia de módulos, carga, limpieza, segmentación, entrenamiento y prueba, opera sobre artefactos versionados que se registran automáticamente en W&B, garantizando la trazabilidad completa de los datos y modelos utilizados.

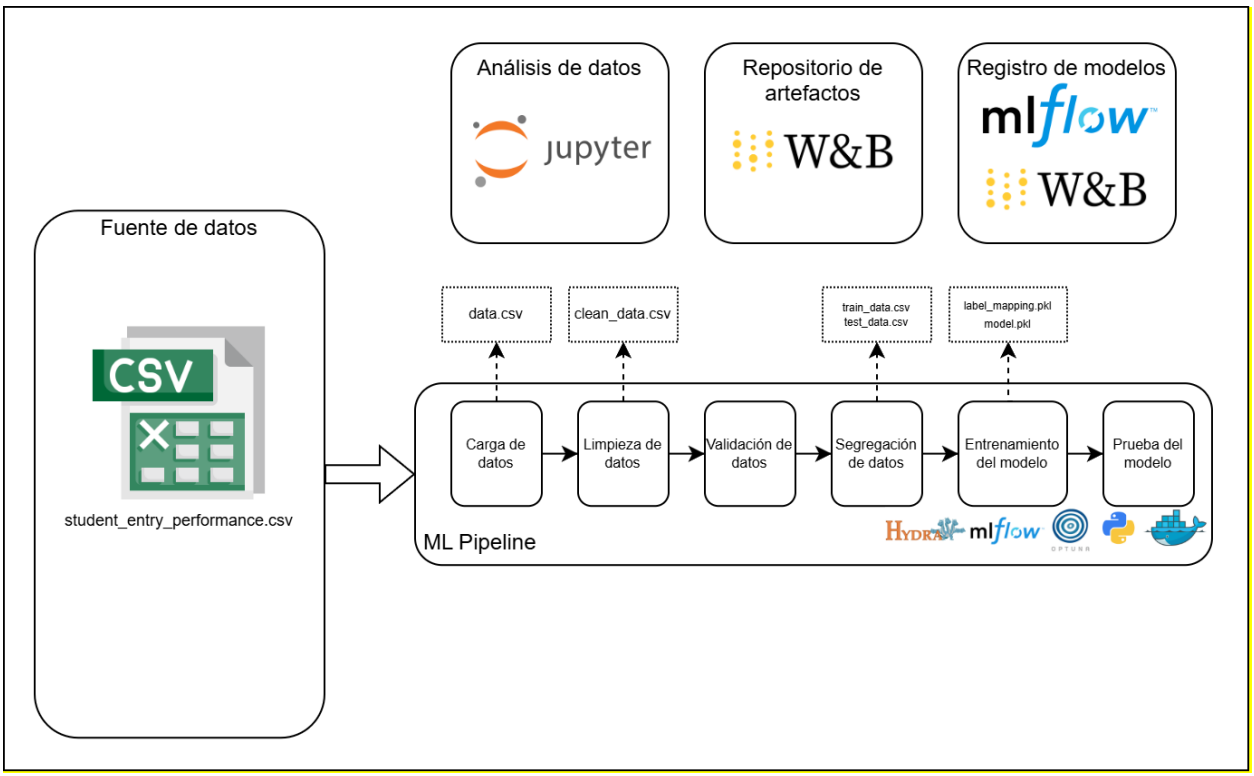

Ilustración 18 Definición de Pipeline

El proceso inicia con la carga del conjunto de datos original, almacenado como artefacto de tipo *raw_data* en W&B. Posteriormente, la etapa de limpieza normaliza categorías, corrige inconsistencias y elimina valores atípicos, generando un nuevo artefacto de tipo *clean_data*. La fase de segregación de datos produce las versiones *train_val_data* y *test_data*, que se registran de manera independiente para conservar su control histórico.

Durante el entrenamiento, el pipeline utiliza Hydra para administrar configuraciones dinámicas y MLflow para registrar el modelo entrenado junto con su firma y metadatos. El modelo y el mapeo de etiquetas se empaquetan y suben como artefacto *model_export* a W&B, lo que asegura su disponibilidad para posteriores evaluaciones o despliegues. Finalmente, la etapa de prueba descarga el modelo desde MLflow, evalúa su rendimiento con el conjunto *test_data* y registra las métricas finales (*Accuracy, F1-score y Precision*) en el tablero de seguimiento de W&B.

La integración entre W&B y MLflow permitió centralizar la gestión de experimentos, configuraciones y artefactos, mientras que el uso de Hydra proporcionó flexibilidad en la parametrización del pipeline. Este esquema eliminó la dependencia de archivos intermedios y posibilitó un flujo completamente trazable, reproducible y escalable. En conjunto, la arquitectura del pipeline refleja un nivel avanzado de automatización que cumple con las mejores prácticas de ingeniería en entornos MLOps.

El experimento analizado corresponde a una ejecución etiquetada como “grim-relic-74”, perteneciente al proyecto *my_ml_projec*t. El entorno de ejecución fue un sistema macOS 26.0.1 (arm64) con Python 3.13.1, operando de forma local en un entorno virtual gestionado por *poetry*. La ejecución se realizó mediante el script main.py, alojado en el repositorio público Estebarra/ML_pipeline.

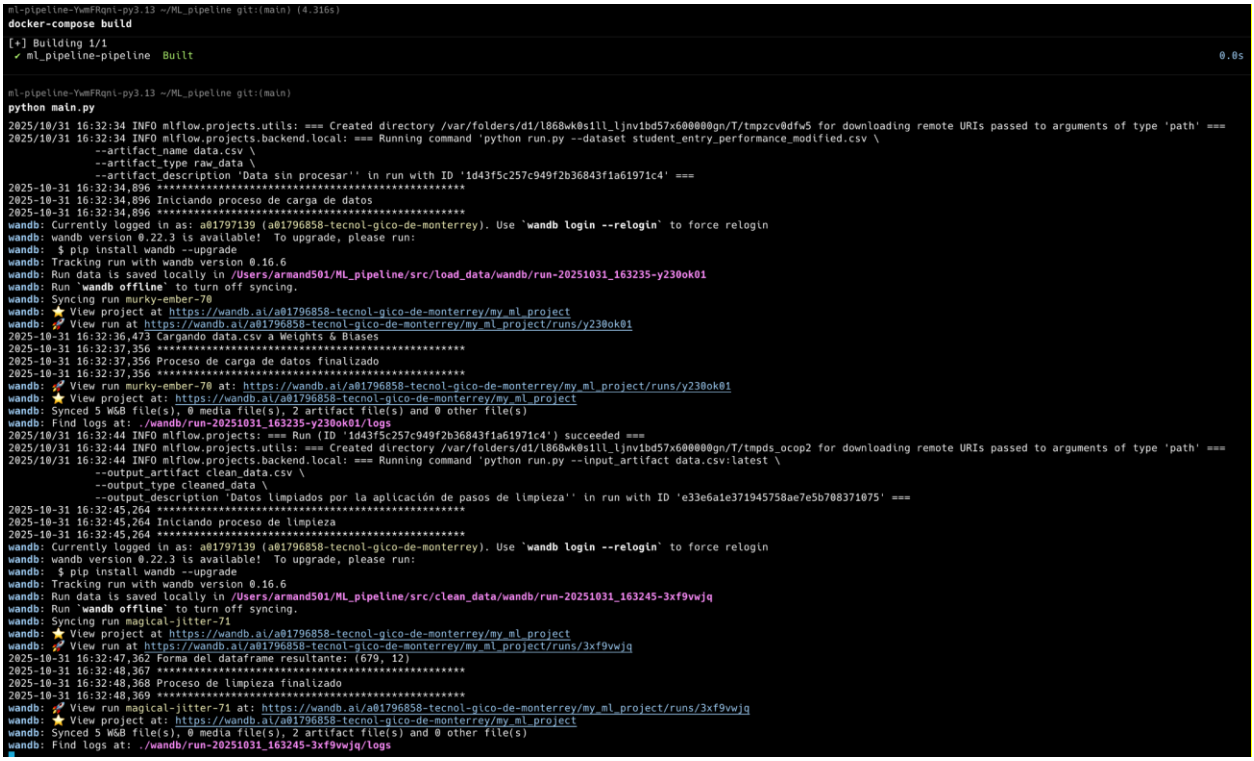

Ilustración 19 Experimento en Entorno Poetry

Dentro de la configuración se registraron dos parámetros clave: el modelo de referencia (*mlflow_model: model_export:latest*) y el conjunto de datos de prueba (*test_dataset: test_data.csv:latest*). W&B identificó además los artefactos consumidos por el proceso: un job proveniente del código fuente, el modelo exportado y el *dataset*

de prueba. Este registro automatizado garantiza trazabilidad completa sobre los elementos utilizados en cada fase de la inferencia, permitiendo reproducir el experimento con precisión.

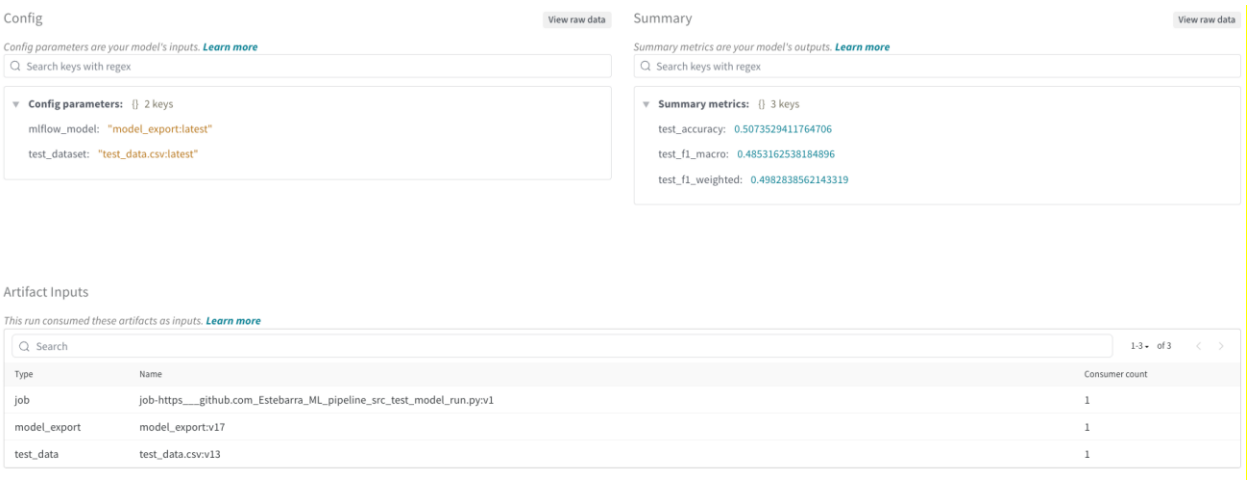

Ilustración 20 Resultados del Modelo en W&B

El modelo alcanzó una exactitud (*accuracy*) de 0.5074, un F1 macro de 0.4853 y un F1 ponderado (*weighted*) de 0.4983. Estas métricas, obtenidas mediante evaluación sobre el *dataset* de prueba, sugieren un desempeño apenas superior al azar. Al analizar el reporte de clasificación, se observa que las clases presentan comportamientos desiguales:

- La clase *Average* alcanza la mayor efectividad, con precisión de 0.64 y recall de 0.72.
- Las clases *Good y Vg* obtienen valores de F1 entre 0.42 y 0.49, reflejando una capacidad limitada de generalización.
- La clase *Excellent* muestra un F1 de apenas 0.32, indicando una subrepresentación o dificultad del modelo para identificarla correctamente.

Dado que el conjunto de datos parece contener varias clases balanceadas, los resultados apuntan a una falta de discriminación efectiva en las fronteras de decisión del modelo o a un proceso de entrenamiento insuficiente.

```
2025-10-31 22:33:35 2025-10-31 16:33:32,699 Descargando artefactos
2025-10-31 22:33:37 wandb:   8 of 8 files downloaded.
2025-10-31 22:33:37 2025-10-31 16:33:35,514 Cargando modelo y realizando inferencia en el conjunto de prueba
2025-10-31 22:33:37 2025-10-31 16:33:35,690 Evaluando el modelo
2025-10-31 22:33:37 2025-10-31 16:33:35,692 Test Accuracy: 0.5074
2025-10-31 22:33:37 2025-10-31 16:33:35,692 Test F1 Macro: 0.4853
2025-10-31 22:33:37 2025-10-31 16:33:35,692 Test F1 Weighted: 0.4983
2025-10-31 22:33:37 2025-10-31 16:33:35,692
2025-10-31 22:33:37 Classification Report:
2025-10-31 22:33:37 2025-10-31 16:33:35,698
2025-10-31 22:33:37               precision    recall  f1-score   support
2025-10-31 22:33:37      Average       0.64      0.72      0.68        32
2025-10-31 22:33:37    Excellent       0.45      0.25      0.32        20
2025-10-31 22:33:37         Good       0.45      0.55      0.49        44
2025-10-31 22:33:37           Vg       0.47      0.42      0.45        40
2025-10-31 22:33:37     accuracy                           0.51       136
2025-10-31 22:33:37    macro avg       0.50      0.48      0.49       136
2025-10-31 22:33:37 weighted avg       0.50      0.51      0.50       136
2025-10-31 22:33:37 2025-10-31 16:33:35,698 ************************************************
2025-10-31 22:33:37 2025-10-31 16:33:35,698 Proceso de Prueba finalizado
2025-10-31 22:33:37 2025-10-31 16:33:35,698 ************************************************
```

Ilustración 21 Métricas Analizadas en W&B

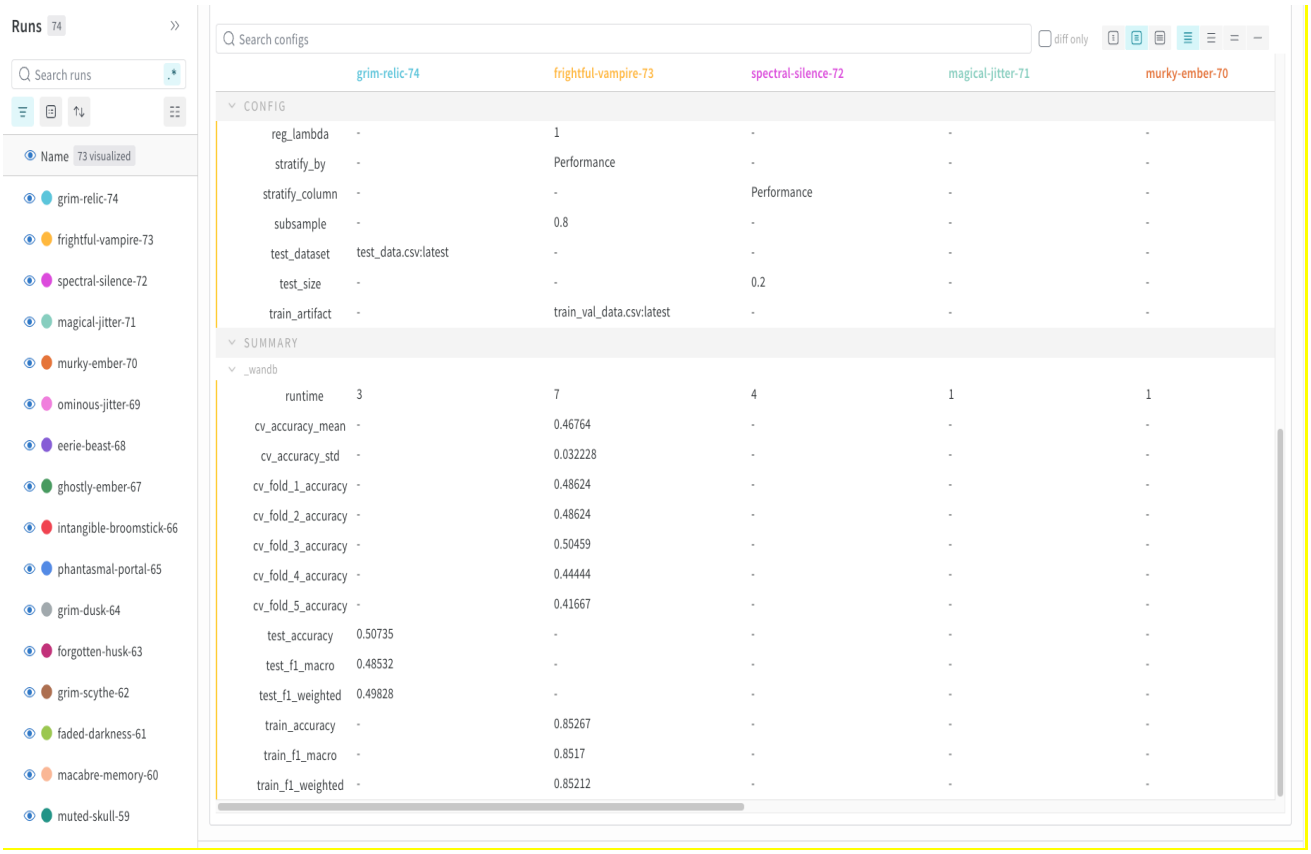

Ilustración 22 Ejecuciones de Experimentos en W&B

Los resultados obtenidos reflejan un modelo con bajo rendimiento en generalización, aunque con un pipeline bien instrumentado. El sistema de registro implementado mediante W&B demuestra su efectividad para documentar, comparar y diagnosticar ejecuciones en entornos de aprendizaje automático.

El principal desafío identificado es la eficiencia del modelo entrenado. Las métricas sugieren la necesidad de revisar la ingeniería de características, el balance de clases y los hiperparámetros utilizados. Sin embargo, la infraestructura de monitoreo ya establecida constituye una base sólida para la mejora iterativa y el despliegue controlado de nuevas versiones del modelo.

Capítulo 6. Conclusiones y Recomendaciones (parciales)

El desarrollo de la Fase 1: Análisis y diseño del modelo predictivo permitió establecer una base sólida para la construcción del sistema de clasificación supervisada orientado a predecir el rendimiento académico de los estudiantes. A través del proceso de limpieza y estandarización del conjunto de datos, se logró garantizar la coherencia interna del mismo, eliminando inconsistencias, errores tipográficos y duplicidades semánticas que afectaban la integridad de la información original.

El análisis exploratorio evidenció que la muestra cuenta con una distribución equilibrada entre las categorías de rendimiento, predominando los niveles *Good y Very Good*, que en conjunto representan más del 60% de los casos. Este hallazgo sugiere que el grupo estudiado presenta un desempeño académico medio-alto, con menor representación de los niveles *Average y Excellent*. Asimismo, se identificaron variables relevantes de tipo demográfico, educativo y familiar que servirán como insumo para el entrenamiento del modelo, permitiendo analizar de manera más precisa los factores asociados al éxito académico.

La incorporación del enfoque MLOps y el uso de herramientas como W&B y Python proporcionaron un marco metodológico reproducible, escalable y trazable, asegurando la transparencia de cada transformación aplicada. Esta integración facilitará el avance hacia las siguientes fases, donde se llevará a cabo el entrenamiento, ajuste y despliegue del modelo predictivo dentro de un entorno controlado y automatizado.

Entre los modelos evaluados, la Regresión Logística Ordinal se perfila como la opción más adecuada para integrarse en fases posteriores del proyecto. Aunque su *Accuracy* (0.5111) se mantiene en un rango similar al de los demás algoritmos, presenta un comportamiento más estable y equilibrado entre las métricas de entrenamiento y validación, además de una mejor capacidad de generalización. Su naturaleza ordinal le permite representar de manera coherente las relaciones jerárquicas entre las categorías de rendimiento académico (*Excellent, Very Good, Good, Average*), lo que otorga una ventaja conceptual frente a modelos puramente categóricos.

A diferencia de los demás enfoques, la Regresión Logística Ordinal mostró curvas de aprendizaje convergentes y una reducción visible del sobreajuste, lo que indica un modelo más consistente y menos dependiente del conjunto de entrenamiento. Por ello, se considera el punto de partida más apropiado para la siguiente fase del proyecto, donde se buscará optimizar su desempeño mediante técnicas de regularización, ajuste de hiperparámetros y potencial integración dentro del entorno automatizado MLOps

Aunque el proceso de limpieza y estandarización del conjunto de datos permitió alcanzar una estructura coherente y adecuada para el modelado, resulta conveniente seguir perfeccionando esta fase en etapas futuras del proyecto. En especial, se sugiere profundizar en la detección de valores atípicos, incorporando métodos estadísticos y representaciones visuales que faciliten la identificación de registros anómalos o inconsistentes dentro de las variables numéricas. Esta mejora contribuiría a fortalecer la calidad del conjunto de datos y asegurar una representación más precisa y confiable de la población estudiada.

Se considera necesario llevar a cabo un ajuste más profundo de los hiperparámetros del modelo, con el fin de potenciar su capacidad predictiva y elevar su desempeño a un nivel superior. Este proceso implicará la exploración sistemática de configuraciones mediante técnicas como *Grid Search* o *Random Search*, analizando diferentes valores de regularización, tipos de penalización y parámetros de optimización. Del mismo modo, será pertinente ampliar el alcance de la validación cruzada e implementar mecanismos de selección automática de variables, orientados a mejorar la representación de las características más influyentes. Con estas acciones, se espera incrementar la precisión y robustez del modelo.

La segunda fase marca un punto de consolidación dentro del proyecto, al transitar de un enfoque exploratorio hacia un entorno estructurado, modular y plenamente reproducible. La adopción del esquema *Cookiecutter* permitió organizar el flujo de trabajo bajo una jerarquía clara y coherente, lo que facilitó la trazabilidad de los componentes y el manejo independiente de los datos, el código y los modelos. Esta estructura se fortaleció con la refactorización del código, que transformó los notebooks iniciales en módulos ejecutables con funciones específicas, proporcionando una base técnica más estable, escalable y de fácil mantenimiento. En conjunto, estas mejoras incrementan la eficiencia operativa y consolidan la coherencia interna del proyecto a lo largo de sus diferentes etapas.

De igual forma, la automatización del pipeline y la integración de herramientas de seguimiento de experimentos como W&B y MLflow permitieron un control más riguroso de las métricas, configuraciones y artefactos generados durante el proceso de modelado. En particular, W&B se consolidó como una plataforma central para la supervisión de los experimentos, al ofrecer un portal visual interactivo donde se registran de manera automática los parámetros de ejecución, las curvas de entrenamiento, los resultados comparativos y las versiones de cada artefacto. Esta capacidad de registro y visualización en tiempo real facilita el análisis de desempeño entre modelos, permite identificar patrones de mejora y promueve la colaboración entre los diferentes roles del equipo de desarrollo.

El registro de modelos y el versionamiento de los datos fortalecieron la reproducibilidad de los resultados y establecieron un marco de gestión más transparente y trazable. Además, la interoperabilidad entre W&B y MLflow posibilitó mantener una relación directa entre los experimentos, los artefactos generados y los modelos exportados, garantizando la consistencia del flujo completo. Con la incorporación de principios de MLOps, esta fase consolida la madurez técnica del sistema y sienta las bases para su evolución hacia entornos de despliegue y monitoreo continuo, asegurando la estabilidad, la transparencia y la sostenibilidad del ciclo de vida del modelo.

Si bien esta fase logró consolidar un entorno estructurado y reproducible, es recomendable fortalecer la gestión de configuración mediante la centralización de parámetros y rutas en archivos YAML integrados al pipeline. Esta práctica permitiría disminuir la dependencia de argumentos definidos en consola y facilitar la automatización de los experimentos en diferentes entornos. También sería conveniente incorporar pruebas unitarias en los módulos principales, junto con un esquema básico de validación continua (CI) que asegure la integridad del código ante futuras modificaciones. Con ello se favorece la estabilidad del sistema y se optimizaría la colaboración entre los distintos roles técnicos involucrados.

De igual manera, resulta pertinente ampliar la integración de *Weights & Biases* como plataforma de seguimiento, aprovechando sus funciones avanzadas para la comparación automatizada de ejecuciones, la elaboración de reportes y la creación de tableros personalizados de métricas que fortalezcan la visualización del desempeño. Asimismo, profundizar en el uso del MLflow *Model Registry* permitiría establecer un flujo más formal para la aprobación, versionamiento y despliegue de modelos.

Referencias

[1] B. C. Fulano-Vargas and I. N. Meneses-Runza, “La evaluación escolar desde la perspectiva de los estudiantes,” *RECIE. Revista Caribeña de Investigación Educativa*, vol. 7, no. 1, pp. 163–182, 2023.

[2] M. A. L. O. Guizado, “Influencia de la evaluación del desempeño directivo y su efectividad en la gestión pública de la educación escolar,” *Ciencia Latina Revista Científica Multidisciplinar*, vol. 6, no. 5, pp. 874–892, 2022.

[3] G. Oyarzún Vargas and A. Falabella, “Indicadores de Desarrollo Personal y Social: La ilusión de la evaluación integral de la calidad,” *Psicoperspectivas*, vol. 21, no. 1, pp. 149–162, 2022.

[4] L. L. Blanco Figueredo and K. E. Arias Ortega, “Perspectiva decolonial de la evaluación escolar en contexto de diversidad social y cultural,” *Perfiles Educativos*, vol. 44, no. 177, pp. 168–182, 2022.

[5] A. Clément, É. Robinot, and L. Trespeuch, “The use of ESG scores in academic literature: A systematic literature review,” *Journal of Enterprising Communities: People and Places in the Global Economy*, vol. 19, no. 1, pp. 92–110, 2025.

[6] X. Zhong, B. Gallagher, S. Liu, B. Kailkhura, A. Hiszpanski, and T. Y. J. Han, “Explainable machine learning in materials science,” *npj Computational Materials*, vol. 8, no. 1, p. 204, 2022.

[7] G. Nicora, M. Rios, A. Abu-Hanna, and R. Bellazzi, “Evaluating pointwise reliability of machine learning prediction,” *Journal of Biomedical Informatics*, vol. 127, p. 103996, 2022.

[8] M. M. John, H. H. Olsson, and J. Bosch, “Towards MLOps: A framework and maturity model,” in *Proc. 2021 47th Euromicro Conf. on Software Engineering and Advanced Applications (SEAA)*, Palermo, Italy, Sept. 2021, pp. 1–8. IEEE.

[9] B. P. R. Rella, “MLOps and DataOps integration for scalable machine learning deployment,” *International Journal for Multidisciplinary Research*, vols. 1–3, 2022. [Online]. Available: https://www.researchgate.net/publication/390554912

[10] F. Lanubile, S. Martínez-Fernández, and L. Quaranta, “Teaching MLOps in higher education through project-based learning,” in *Proc. 2023 IEEE/ACM 45th Int. Conf. on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, Melbourne, Australia, May 2023, pp. 95–100. IEEE.

[11] S. Mohan, G. Gujrati, and N. Senthamarai, “Measuring and predicting student’s academic performance using ML,” in *Proc. Congress on Smart Computing Technologies*, Singapore: Springer Nature Singapore, Dec. 2022, pp. 387–403.

[12] C. F. E. Hoyos and J. C. A. Barilla, “El sistema educativo como sistema esencial para el desarrollo y la transformación social,” *Revista Oratores*, no. 14, pp. 144–156, 2021.

[13] L. M. Castro-Benavides, J. A. Tamayo-Arias, and D. Burgos, “Escenarios de la docencia frente a la transformación digital de las Instituciones de Educación Superior,” *Education in the Knowledge Society (EKS)*, vol. 23, pp. e27866–e27866, 2022.

[14] W. M. Romero Casalliglla, J. O. Chulca Abalco, G. G. Imbaquingo Guzmán, S. E. Pineda Anchaguano, E. C. Aules Aules, G. O. Tipán Sánchez, *et al.*, “Evaluación para el aprendizaje: más allá de las calificaciones,” *Revista InveCom*, vol. 5, no. 1, 2025.

[15] C. E. Mendoza Suárez and C. A. Bullón Romero, “Gestión del conocimiento en instituciones de educación superior: una revisión sistemática,” *Horizontes. Revista de Investigación en Ciencias de la Educación*, vol. 6, no. 26, pp. 1992–2003, 2022.

[16] H. Saravia Domínguez, P. Saavedra Villar, L. M. Felices Vizarreta, M. M. Campos Espinoza, and J. R. Janampa Urbano, “La aplicación del diseño curricular por competencias en la Educación Superior: Una revisión sistemática 2019–2023,” *Comuni@cción*, vol. 15, no. 1, pp. 92–104, 2024.

[17] Y. P. C. González, S. Z. J. Mora, and R. G. M. Morillo, “Tendencias y desafíos políticos y socioculturales de la educación superior contemporánea en Latinoamérica,” *Revista Boletín Redipe*, vol. 11, no. 1, pp. 71–91, 2022.

[18] E. J. C. Ortegón and R. R. Rey-González, “Una revisión crítica del modelo de admisión por estratos académicos en la Universidad Nacional de Colombia,” *Revista Educación Superior y Sociedad (ESS)*, vol. 34, no. 2, pp. 52–75, 2022.

[19] M. A. Rodríguez Medina, E. R. Poblano-Ojinaga, L. Alvarado Tarango, A. González Torres, and M. I. Rodríguez Borbón, “Validación por juicio de expertos de un instrumento de evaluación para evidencias de aprendizaje conceptual,” *RIDE. Revista Iberoamericana para la Investigación y el Desarrollo Educativo*, vol. 11, no. 22, 2021.

[20] J. I. Pincay-Ponce, A. De Giusti, J. J. Reyes-Cárdenas, A. G. Franco-Pico, A. V. Macías-Espinales, and P. A. Quiroz-Palma, “Analítica de datos de factores socioeconómicos que inciden en el rendimiento escolar: Revisión sistemática,” *Revista Ibérica de Sistemas e Tecnologias de Informação*, no. E54, pp. 531–545, 2022.

[21] J. N. M. Álava, L. A. M. Bermeo, J. Morales-Carrillo, and L. Cedeño-Valarezo, “Modelos de aprendizaje automático: Aplicación y eficiencia,” *Revista Científica de Informática ENCRIPTAR*, vol. 7, no. 14, pp. 87–114, 2024.

[22] O. E. D. Rodríguez and M. G. P. Hernández, “Minería de intenciones a partir de una base del conocimiento y aprendizaje automático supervisado,” *3C TIC: Cuadernos de Desarrollo Aplicados a las TIC*, vol. 10, no. 3, pp. 65–101, 2021.

[23] A. Almufarreh, K. M. Noaman, and M. N. Saeed, “Academic teaching quality framework and performance evaluation using machine learning,” *Applied Sciences*, vol. 13, no. 5, p. 3121, 2023.

[24] K. Fahd, S. Venkatraman, S. J. Miah, and K. Ahmed, “Application of machine learning in higher education to assess student academic performance, at-risk, and attrition: A meta-analysis of literature,” *Education and Information Technologies*, vol. 27, no. 3, pp. 3743–3775, 2022.

[25] A. Guillén Perales, F. Liébana-Cabanillas, J. Sánchez-Fernández, and L. J. Herrera, “Assessing university students’ perception of academic quality using machine learning,” *Applied Computing and Informatics*, vol. 20, no. 1/2, pp. 20–34, 2024.

[26] J. M. A. Acosta, M. L. Ramirez, and R. G. Cabrera, “Impacto del preprocesamiento en la clasificación automática de textos usando aprendizaje supervisado y Reuters 21578,” *Revista Colombiana de Tecnologías de Avanzada*, vol. 1, no. 43, pp. 110–118, 2024.

[27] B. P. Cordero-Torres, “Algoritmos de aprendizaje supervisado para proyección de ventas de camarón ecuatoriano con lenguaje de programación Python,” *Economía y Negocios*, vol. 13, no. 2, pp. 30–51, 2022.

[28] E. M. García, C. C. López, J. A. M. Rivas, and D. L. A. Capistran, “Evaluación de algoritmos de aprendizaje supervisado usando modelos binarios para clasificación de análisis de sentimiento,” *Tecnología Educativa Revista CONAIC*, vol. 11, no. 1, pp. 92–97, 2024.

[29] M. C. Mihaescu and P. S. Popescu, “Review on publicly available datasets for educational data mining,” *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 11, no. 3, p. e1403, 2021.

[30] J. Han, J. Pei, and H. Tong, *Data Mining: Concepts and Techniques*. Cambridge, MA: Morgan Kaufmann, 2022.

[31] S. K. Dubey, S. Gupta, S. K. Pandey, and S. Mittal, “Ethical considerations in data mining and database research,” in *Proc. 2024 11th Int. Conf. on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, Mar. 2024, pp. 1–4. IEEE.

[32] C. I. Loeza-Mejía, E. Sánchez-DelaCruz, and I. O. José-Guzmán, “Machine learning methodologies: history and challenges,” *Evolutionary Intelligence*, vol. 18, no. 3, p. 58, 2025.

[33] B. Eken, S. Pallewatta, N. Tran, A. Tosun, and M. A. Babar, “A multivocal review of MLOps practices, challenges and open issues,” *ACM Computing Surveys*, vol. 58, no. 2, pp. 1–35, 2025.

[34] S. Mei, C. Liu, Q. Wang, and H. Su, “Model provenance management in MLOps pipeline,” in *Proc. 2022 8th Int. Conf. on Computing and Data Engineering*, Tokyo, Japan, Jan. 2022, pp. 45–50.

[35] S. Moreschini, D. Hästbacka, and D. Taibi, “MLOps pipeline development: The OSSARA use case,” in *Proc. 2023 Int. Conf. on Research in Adaptive and Convergent Systems*, Barcelona, Spain, Aug. 2023, pp. 1–8.

[36] F. J. Iriarte, M. E. Ortiz, L. Unzueta, J. Martínez, J. Zaldivar, and I. Arganda-Carreras, “Multi-level XAI-driven MLOps pipeline for the adjustment of fruit and vegetable classifiers,” in *Proc. 2024 IEEE 12th Int. Conf. on Intelligent Systems (IS)*, Warsaw, Poland, Aug. 2024, pp. 1–6. IEEE.

[37] M. Grandini, E. Bagli, and G. Visani, “Metrics for Multi-Class Classification: an Overview,” *arXiv preprint* arXiv:2008.05756, 2020.

[38] A. Tharwat, “Classification Assessment Methods,” *Applied Computing and Informatics*, vol. 17, no. 1, pp. 168–192, 2021, doi: 10.1016/j.aci.2018.08.003.

[39] J. Opitz, “A Closer Look at Classification Evaluation Metrics and a Critical Reflection of Common Evaluation Practice,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 820–836, 2024, doi: 10.1162/tacl_a_00675.

[40] O. Rainio, P. Kumpulainen, and T. Hämäläinen, “Evaluation metrics and statistical tests for machine learning,” *Scientific Reports*, vol. 14, 2024, doi: 10.1038/s41598-024-56706-x.

[41] M. Ghanem *et al.*, “Limitations in Evaluating Machine Learning Models for Imbalanced Binary Outcome Classification in Spine Surgery: A Systematic Review,” *Brain Sciences*, vol. 13, no. 12, p. 1723, 2023, doi: 10.3390/brainsci13121723.

[42] Student Performance on an Entrance Examination [Dataset]. (2018). UCI Machine Learning Repository. https://doi.org/10.24432/C58D0H.

[43] DrivenData. *Cookiecutter Data Science Template*, 2021. Disponible en: https://drivendata.github.io/cookiecutter-data-science

[44] Fowler, M.: *Refactoring: Improving the Design of Existing Code*, 2nd ed. Addison-Wesley (2019).

[45] Pedregosa, F. et al.: *Scikit-learn: Machine Learning in Python*, *Journal of Machine Learning Research*, 12 (2011) 2825–2830.

[46] Google Cloud: *MLOps: Continuous Delivery and Automation Pipelines in Machine Learning*, 2020.

[47] Zaharia, M. et al.: *Accelerating the Machine Learning Lifecycle with MLflow*. *Databricks Technical Report* (2018).
[48] Biewald, L.: *Experiment Tracking with Weights and Biases*. *W&B Documentation* (2020).

[49] Pineau, J., Vincent-Lamarre, P., Sinha, K. et al.: *Improving Reproducibility in Machine Learning Research (A NeurIPS Paper Checklist)*. *Communications of the ACM*, 64(3), 2021.

[50] Gundersen, O.E., Kjensmo, S.: *State of the Art: Reproducibility in Artificial Intelligence*. *AI Communications*, 31(1), 2018.

[51] Valerio, A., Martin, F.: *Data Version Control for Machine Learning Workflows*. *SoftwareX*, 17, 2022.

[52] Ribeiro, M. T., Singh, S., Guestrin, C.: *"Why Should I Trust You?" Explaining the Predictions of Any Classifier*. *Proceedings of KDD*, 2016.