Proyecto Fase 2. Predicción del Riesgo Crediticio con Machine Learning

TC5044.10 – Operaciones de aprendizaje automático

Equipo 5

Data Engineer - Ingrid Pamela Ruiz Puga (A01021209)
Data Scientist – Fernando Emmanuel Pérez Escalante (A01796934)
Machine Learning Engineer - Sebastián Cabezón Rosas (A01840261)
Software Enginer – Quirec Angeles Martínez (A01745050)
DevOps – Daniel Ivan Benitez Fernandez de Jauregui (A01795860)

ESCUELA DE INGENIERÍA Y CIENCIAS

MAESTRÍA EN INTELIGENCIA ARTIFICIAL APLICADA

2 de Noviembre de 2025

INTRODUCCIÓN

Este documento corresponde a la Fase 2 del proyecto “Predicción del Riesgo Crediticio con Machine Learning”, dando continuidad al análisis, limpieza y modelado inicial realizado en la Fase 1.

En esta etapa se aplicaron principios de MLOps para estructurar el proyecto profesionalmente con **Cookiecutter**, refactorizar el código, automatizar procesos mediante **pipelines**, y gestionar experimentos, métricas y modelos utilizando **DVC, MLflow y Dagshub**.

El objetivo fue mejorar la organización, reproducibilidad y trazabilidad del proyecto, asegurando un desarrollo colaborativo y escalable.

RESUMEN DEL PROYECTO Y ML CANVAS

El dataset South German Credit contiene 1,000 ejemplos (700 créditos buenos, 300 malos) con 20 variables predictoras más la variable objetivo ('credit_risk': bueno/malo). Es un problema clásico de clasificación binaria orientado a predecir si un solicitante de crédito será buen o mal pagador, con aplicaciones directas en el ámbito financiero.

NATURALEZA DEL PROBLEMA

El objetivo es construir un modelo de clasificación binaria que prediga la probabilidad de impago de un solicitante. Esto permite a una entidad financiera tomar decisiones más informadas sobre aprobación o rechazo de crédito, reduciendo pérdidas por morosidad.

VARIABLES PRINCIPALES

El dataset contiene variables relacionadas con historial crediticio, empleo, ahorros, propiedades, estado civil, edad y más. No hay valores faltantes y las variables están discretizadas o codificadas.

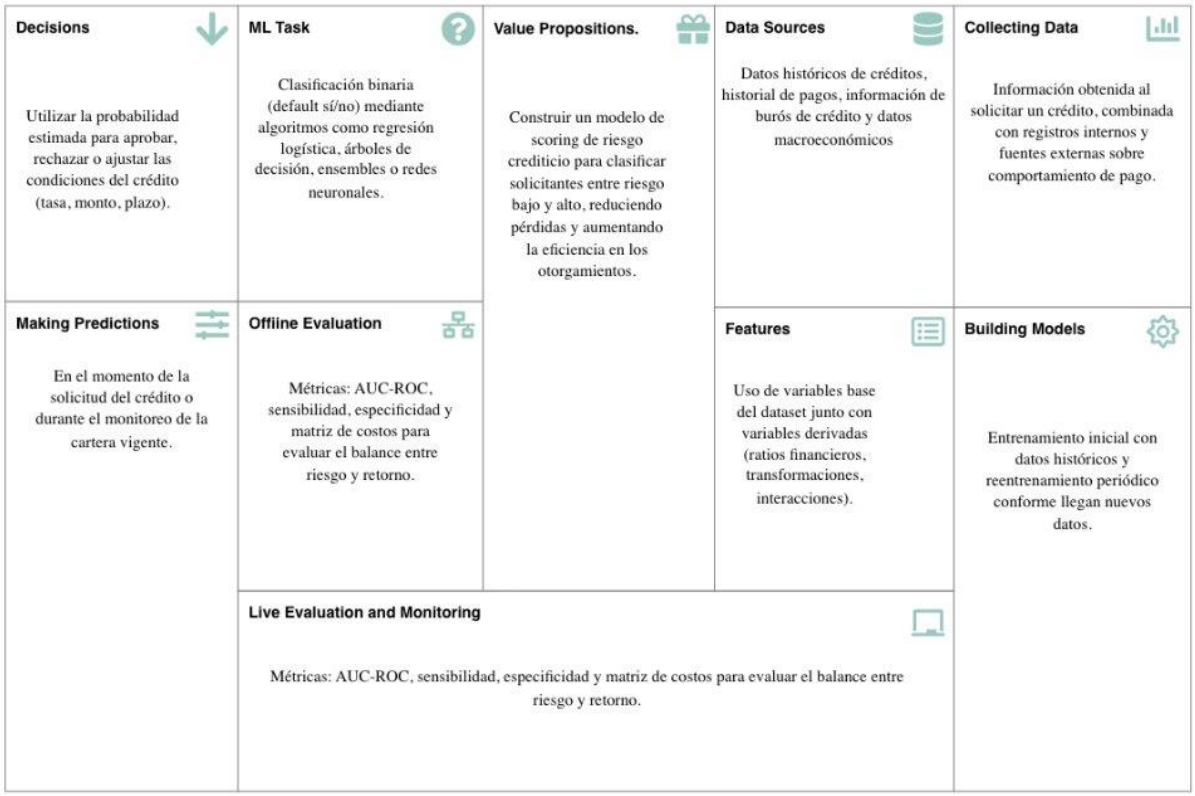
PRINCIPALES RETOS, SESGOS Y RIESGOS

- Desbalance entre clases reales y dataset.
- Posibles sesgos demográficos (e.g., trabajador extranjero).
- Datos antiguos (años 70s) que limitan la vigencia.
- Necesidad de explicabilidad en modelos financieros.
- Costos de error asimétricos (falsos negativos más costosos).
- Riesgo de multicolinealidad entre variables.

OPORTUNIDADES

El dataset es ideal para experimentación, prototipado y benchmarking en proyectos de scoring crediticio. Permite comparar distintos algoritmos y técnicas de ingeniería de características,

El ML Canvas describe de forma estructurada los elementos de un proyecto de machine learning: objetivo, fuentes de datos, características, predicción, decisiones y evaluación. A continuación, se presenta la versión adaptada para el dataset South German Credit.



Ver canvas en

https://github.com/secabezon/MLOps/blob/main/docs/MLCanvas_South_German_Credit.docx

Estructuración y Refactorización del Código

Organización del Proyecto (Cookiecutter)

Estructura propuesta para la elaboración del presente proyecto:

Durante esta parte de manera colaborativa se hicieron actualizaciones aplicando buenas prácticas de codificación con el objetivo de tener un código modular y reutilizable además de separarlo por clases y funciones. Esto con para permitir una mayor orden, legibilidad, mantenibilidad y escalabilidad de la solución.

Para la estructura general se siguió utilizando de Cookicutter como organización modular del proyecto además de hacerlo modular separándolo en y clases por separado.

La nueva estructura general del proyecto quedo de la siguiente manera.

```
MLOps-main/
|
├── data/                ← (a veces generado en ejecución)
├── docs/                ← documentación del proyecto
├── notebooks/           ← análisis y experimentos exploratorios
├── references/          ← información externa o documentación base
├── reports/             ← resultados, gráficas y métricas
├── scripts/             ← scripts auxiliares para tareas específicas
├── src/                 ← código fuente principal (core del modelo ML)
|   ├── data/            ← carga y procesamiento de datos
|   ├── features/        ← ingeniería de características
|   ├── models/          ← entrenamiento, evaluación y predicción
|   ├── visualization/   ← visualización de resultados
|   └── __init__.py
|
├── dvc.yaml / dvc.lock  ← control de versiones de datos con DVC
├── requirements.txt     ← dependencias del entorno
├── setup.py             ← instalación como paquete Python
├── Makefile             ← comandos automatizados (pipeline)
└── README.md            ← documentación principal
```

Esta estructura permite:

- **Reproducibilidad** completa del flujo de datos y entrenamiento.
- **Escalabilidad**, al poder integrar nuevas etapas sin afectar módulos existentes.
- **Trazabilidad**, al versionar datos, modelos y experimentos.
- **Colaboración** eficiente gracias a responsabilidades bien definidas.

  Ver repositorio

  https://github.com/secabezon/MLOps/blob/main/docs/MLCanvas_South_German_Credit.docx

Aplicación de Mejores Prácticas de Codificación en el Pipeline de Modelado

Durante esta fase, se aplicaron buenas prácticas de ingeniería de software para construir un pipeline robusto y reproducible utilizando `Scikit-learn`. El objetivo fue automatizar las etapas críticas del flujo de modelado y asegurar su mantenibilidad y reutilización en distintas configuraciones experimentales.

Diseño del Pipeline Automatizado

El pipeline se desarrolló como una secuencia lógica de transformaciones, utilizando la clase `Pipeline` de `Scikit-learn`. Las etapas incluyeron:

- **Limpieza de datos:** conversión de tipos, normalización de nombres y tratamiento de valores faltantes.
- **Imputación:** se empleó `KNNImputer` para columnas numéricas, aprovechando relaciones multivariadas.
- **Escalado y codificación:** mediante `StandardScaler` para variables numéricas y `OneHotEncoder` o `OrdinalEncoder` para categóricas.

- **Selección de características:** con `SelectKBest` basado en ANOVA F-test.
- **Modelado:** se integraron distintos clasificadores (`LogisticRegression`, `RandomForest`, `XGBoost`) como parte final del pipeline.

Mejores prácticas aplicadas

- **Modularización:** las funciones de preprocesamiento y entrenamiento se encapsularon en módulos independientes (`src/features/`, `src/models/`) para facilitar pruebas y reutilización.
- **Parámetros centralizados:** se utilizó el archivo `params.yaml` para controlar hiperparámetros, cantidad de features seleccionadas y configuraciones del pipeline, separando lógica de configuración.
- **Tipado estático y validación:** se utilizó `mypy` y `flake8` para asegurar consistencia del código y detección temprana de errores.
- **Pruebas:** se incluyeron pruebas unitarias con `pytest` para validar componentes individuales del pipeline.

Reproducibilidad y mantenimiento

El pipeline completo fue serializado mediante `joblib` y versionado con `DVC`, lo que permite su reutilización y trazabilidad en nuevas ejecuciones. Esta estructura facilita futuras mejoras, comparaciones de modelos y despliegue controlado.

Transición de notebooks a scripts modulares

La etapa inicial del proyecto se trabajó desde un notebook de EDA con código lineal y poca separación de responsabilidades. Como parte de esta fase, ese contenido fue refactorizado y encapsulado en clases y funciones organizadas por propósito, facilitando su reutilización y escalabilidad.

Se puede observar que en la versión anterior se utilizaba un Jupyter Notebook, donde se integraban diferentes etapas como análisis exploratorio, limpieza y manipulación de datos, lo que dificultaba la escalabilidad y carecía de una estructura que facilitara la realización de pruebas, el mantenimiento y la expansión del proyecto. Con los cambios realizados, se ha adoptado la programación orientada a objetos, organizando el código en clases y funciones. Esto permite crear bloques definidos y bien estructurados, facilitando su mantenimiento, escalabilidad y pruebas, además de posibilitar la integración de pipelines y la puesta en producción del código.

Comparativo entre EDA anterior y Actual.

LOGROS CLAVE DE ESTA FASE:

- **Refactorización del Código**
  Se reorganizó el código en módulos independientes, eliminando duplicaciones e incorporando ***docstrings***, anotaciones de tipo (***type hints***) y uso de ***logging***. Esta estructura mejora la claridad, facilita pruebas y documentación, y separa claramente etapas del pipeline (EDA, preprocesamiento, entrenamiento, etc.).
- **Automatización con Pipelines (Scikit-learn + DVC)**
  Se construyó un pipeline reproducible utilizando `scikit-learn` y `DVC`, que abarca preprocesamiento, entrenamiento y evaluación. Cada etapa está controlada mediante los archivos `dvc.yaml` y `params.yaml`, permitiendo su ejecución secuencial con `dvc repro` y el seguimiento de métricas con `dvc metrics show`.
- **Calidad de Código y Estilo (PEP8)**
  Se integraron herramientas de linting y formateo automático (`flake8`, `black`, `mypy`) junto con pruebas unitarias (`pytest`) para asegurar la calidad del código y prevenir errores en futuras iteraciones.

SEGUIMIENTO DE EXPERIMENTOS, VISUALIZACIÓN DE RESULTADOS Y GESTIÓN DE MODELOS

Durante esta fase, se integraron herramientas de gestión de experimentos y versionado con el objetivo de asegurar la trazabilidad completa del flujo de trabajo de Machine Learning. Se utilizaron **DVC**, **DagsHub** y **MLflow**, permitiendo controlar versiones de datos, visualizar métricas y registrar modelos.

VERSIONADO DE DATOS Y ARTEFACTOS (DVC + DAGSHUB)

El control de versiones de datos y modelos se implementó mediante **DVC**, conectando el repositorio con **DagsHub**como almacenamiento remoto. Esto permitió:

- Rastrear archivos grandes como datasets procesados y modelos entrenados sin almacenarlos directamente en Git.
- Sincronizar versiones de artefactos con el código fuente.
- Visualizar los cambios y versiones desde la interfaz de DagsHub.

Los comandos utilizados incluyeron `dvc add`, `dvc push` y `dvc pull`, asegurando que todo el equipo pudiera reproducir los resultados de forma consistente.

REGISTRO Y COMPARACIÓN DE EXPERIMENTOS (MLFLOW)

Para registrar y comparar experimentos, se incorporó **MLflow Tracking**, lo que permitió almacenar información clave de cada ejecución:

- Configuraciones de entrada: hiperparámetros, selección de variables, tipo de modelo.
- Métricas de desempeño: Accuracy, F1-score y ROC AUC.
- Artefactos generados: modelos serializados (`.joblib`), gráficos y logs.

Se configuró un servidor local de MLflow con tracking URI y rutas organizadas por experimento. La interfaz web de MLflow permitió comparar visualmente los experimentos, identificar las mejores configuraciones y tomar decisiones informadas sobre qué modelo conservar.

Registro de Modelos

Los modelos entrenados fueron registrados tanto con DVC (para versionado) como en el **Model Registry** de MLflow, documentando:

- Nombre y versión del modelo.
- Métricas alcanzadas en entrenamiento y validación.
- Hiperparámetros y pipeline asociado.

Esto garantiza que cada modelo pueda ser reproducido, evaluado y utilizado en etapas posteriores como despliegue o monitoreo.

Resultados y Comparativa de Modelos

Se siguieron utilizando los mismos 3 modelos que se habían mencionado en la primera parte y con los siguiente cofiguración:

- Random forest : `n_estimators=300`, `learning_rate=0.1`, `max_depth=6`. `subsample=0.8`, `colsample_bytree=0.8`
- Regresión Lógistica: `solver='lbfgs'`, `max_iter=100`
- XGboost: `n_estimators=100`, `n_jobs=-1`

Durante esta fase, se entrenaron y evaluaron tres modelos de clasificación supervisada utilizando el pipeline automatizado:

Modelo	Accuracy	F1 Weighted	ROC AUC
Random Forest	0.9208	0.9186	0.9422
Regresión Logística	0.9010	0.8995	0.9463
XGBoost	0.9257	0.9239	0.9195

Interpretación de resultados:

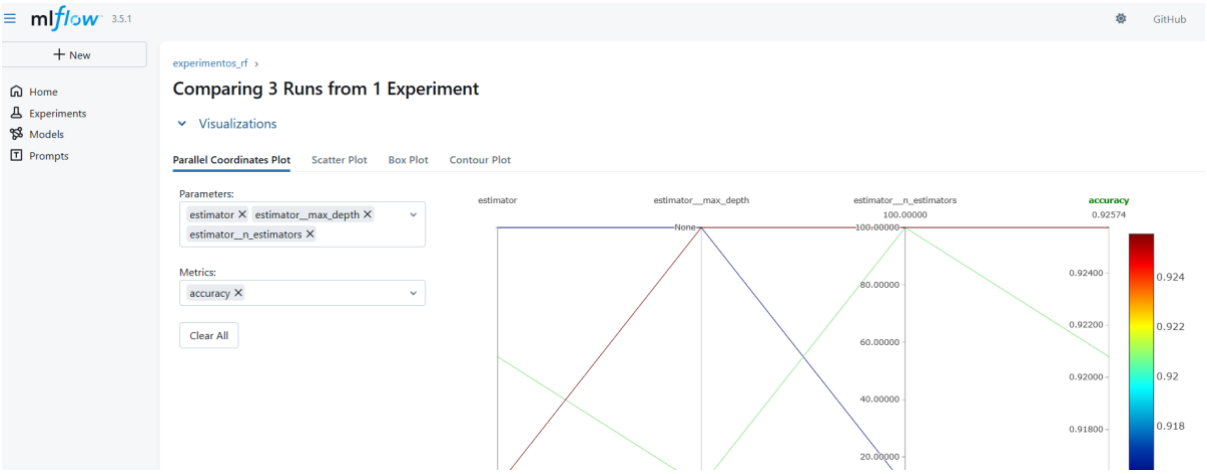
El modelo **XGBoost** obtuvo el mejor desempeño general en términos de ***accuracy*** y ***F1-score***, mostrando una alta capacidad de clasificación. Por su parte, la **Regresión Logística** presentó el mejor ***ROC AUC***, lo que indica una capacidad destacada para distinguir entre clases. Finalmente, **Random Forest** ofreció resultados sólidos, siendo una opción robusta y balanceada.

Capturas de MLflow Experiments:

Las ejecuciones fueron registradas mediante MLflow, permitiendo una visualización clara de los experimentos, métricas y parámetros utilizados. Las capturas de pantalla correspondientes se integran en el anexo de este documento.

En el siguiente ejemplo se pude observar un resumen de los 3 modelos que se están comparando además de cómo sus parámetros influyen en el valor del resultado accuracy. Analizando la visualización podemos observar que

El modelo 1 (XGBoost) y el 3 (random forest) tienen el mismo número de estimadores (`100.00`), pero resultados muy diferentes. La única diferencia visible en los parámetros seleccionados es el valor en el eje estimator , lo que sugiere que el tipo de modelo es el que más impacta en la diferencia de precisión entre estos dos.



Conclusiones y Reflexión Final

Durante esta segunda fase, consolidamos el trabajo iniciado en la etapa anterior incorporando herramientas y prácticas que garantizan la escalabilida d, trazabilidad y reproducibilidad del proyecto de machine learning.

Uno de los principales aprendizajes fue la **importancia de estructurar el proyecto** desde el inicio, siguiendo un esquema claro y profesional (Cookiecutter), lo cual facilitó la colaboración entre los integrantes y la modularización del código.

Además, se trabajó activamente en **ramas paralelas de desarrollo**, las cuales fueron consolidadas y versionadas mediante Git y DVC, permitiendo un seguimiento preciso de los cambios tanto en el código como en los datos y modelos. Esta práctica fortaleció el control de versiones y permitió un flujo de trabajo ordenado y confiable.

La **incorporación de MLflow** fue clave para el seguimiento de experimentos, ya que permitió registrar, visualizar y comparar los resultados obtenidos por distintos modelos, junto con sus hiperparámetros y métricas de evaluación. Esto mejoró la toma de decisiones y facilitó el análisis iterativo.

Finalmente, se aplicaron principios de **escalabilidad y mantenibilidad del código**, reorganizando scripts en funciones reutilizables y módulos independientes, lo cual sentó las bases para el futuro despliegue y mantenimiento del sistema.

Próximos pasos:

- Exponer el modelo como servicio API con **FastAPI**.
- Contenerizar el proyecto con **Docker** para facilitar el despliegue.
- Diseñar un plan de monitoreo básico para detectar **data drift** y asegurar la vigencia del modelo en producción.

EVIDENCIAS DEL PROYECTO

- ML canvas https://github.com/secabezon/MLOps/blob/main/docs/MLCanvas_South_German_Credit.docx

- Github https://github.com/secabezon/MLOps

- Dagshub https://dagshub.com/Pamela-ruiz9/MLOps

- EDA notebook https://github.com/secabezon/MLOps/blob/main/notebooks/EDA_german_credit.ipynb

- Presentación fase 1 https://tecmx-my.sharepoint.com/:p:/r/personal/a01840261_tec_mx/_layouts/15/Doc.aspx?sourcedoc=%7BD0B2759E-10D4-471D-B482-92EED3F483C4%7D&file=Presentaci%25u00f3n.pptx&action=edit&mobileredirect=true

- Presentación fase 2 https://tecmx-my.sharepoint.com/:p:/g/personal/a01796934_tec_mx/EVQPjSgVCTlFmX5l2IVCU14BKV0egOX7Q9KLBsdNoIiH_Q?e=UYehV5

- Video Fase 1 https://youtu.be/8-Dujrs5PNo

- Video Fase 2 https://youtu.be/2ZulCkdJ65o

- Log Github participantes

```
fernandoperezescalante@MacBook-Pro-de-Fernando MLOps % git log --pretty=format:"%h - %an <%ae> - %s" -20
5214bd0 - A01796934 <a01796934@tec.mx> - Add files via upload
0aa0c7e - Quirec <126831779+Quirec@users.noreply.github.com> - Add files via upload
137e7dd - Daniel Benitez <A01795860@tec.mx> - docs: prueba de autor correcto
fad99cc - A01796934 <a01796934@tec.mx> - Add files via upload
16f7f15 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Refresh DVC pointers for raw/processed files
8f294ad - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Agrego gitig
5f41114 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Elimina punteros DVC obsoletos
e921c0b - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - DVC: actualiza snapshot de models/ (nuevos .joblib)
07b74c7 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - carpetas acomodo
23df32d - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - DVC: actualiza carpeta models/ (nuevos artefactos)
58c56a8 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Track models/ with DVC
250db72 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Stop tracking models
ec83262 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Top features informativo
8c7b7b2 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Actualizacion  de archivos
0b5b55a - danieben <danieben@Daniels-MacBook-Pro.local> - Agrega documentación MLCanvas en carpeta docs
b4374e9 - sebastian Cabezon <secabezon21@gmail.com> - Se corrige subir data
1935dd7 - sebastian Cabezon <secabezon21@gmail.com> - Se agrega modelo y ajustes en codigo
efb5640 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Quitamos carpeta esordenada
bfe9da6 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Agregamos fase de modelado
494fa99 - Pamela Ruiz <pamela_ruiz@ciencias.unam.mx> - Agrego json de reglas de limpieza por variable
fernandoperezescalante@MacBook-Pro-de-Fernando MLOps %
```

- MLflow : https://dagshub.com/Pamela-ruiz9/MLOps.mlflow/#/experiments/3?searchFilter=&orderByKey=attributes.start_time&orderByAsc=false&startTime=ALL&lifecycleFilter=Active&modelVersionFilter=All+Runs&datasetsFilter=W10%3D

REFERENCIAS

- **Dua, D., & Graff, C.** (2019). ***UCI Machine Learning Repository: Statlog (German Credit Data)***. University of California, Irvine. Recuperado de: https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

- **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É.** (2011). ***Scikit-learn: Machine Learning in Python***. Journal of Machine Learning Research, 12, 2825–2830. Recuperado de: https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
- **DVC** (2023). ***Data Version Control Documentation***. Iterative.ai. Recuperado de: https://dvc.org
- **DagsHub** (2023). ***Collaborative Data Science and Versioning Platform***. Recuperado de: https://dagshub.com
- **MLflow** (2023). ***An Open Source Platform for the Machine Learning Lifecycle***. Recuperado de: https://mlflow.org